POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea in Ingegneria Informatica

PEDESTRIAN DETECTION AND TRACKING IN LOW QUALITY IMAGES WITH DEEP NEURAL NETWORKS

Relatore: Prof. Luigi PIRODDI

Correlatore: Prof. Marco TAGLIASACCHI

Tesi di Laurea di:

Federico MONTI

Matr. 817537

Anno Accademico 2014 / 2015

Contents

List of Figures

List of Tables

Abbreviations

ACF	**A**ggregated **C**hannel **F**eatures
BB	**B**ounding **B**ox
CNN	**C**onvolutional **N**eural **N**etwork
CVPR	**C**omputer **V**ision and **P**attern **R**ecognition Conference
DNN	**D**eep **N**eural **N**etwork
FOV	**F**ield **O**f **V**iew
FLOPS	**FL**oating point **O**perations **P**er **S**econd
GT	**G**round **T**ruth
HOG	**H**istogram of **O**riented **G**radients
ILSVRC	**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge
LDCF	**L**ocally **D**ecorrelated **C**hannel **F**eatures
NIN	**N**etwork **I**n **N**etwork
NN	**N**eural **N**etwork
PETS	**P**erformance **E**valuation of **T**racking and **S**urveillance
R-CNN	**R**egions with **C**onvolutional **N**eural **N**etwork **F**eatures
ReLU	**R**ectified **L**inear **U**nit
RMSPE	**R**oot **M**ean **S**quare **P**rediction **E**rror
SGD	**S**tochastic **G**radient **D**escent
SoC	**S**ystem **o**n a **C**hip
SVM	**S**upport **V**ector **M**achine
VOC	**V**isual **O**bject **C**lasses Challenge

Sommario

Negli ultimi anni sono stati fatti grandi miglioramenti nel campo della Computer Vision grazie a nuove e potenti architetture parallele in grado di processare grandi quantitativi di dati in poco tempo, permettendo di sviluppare e studiare nuove soluzioni altrimenti non analizzabili.

Uno dei problemi più tipici che in tale ambito si deve affrontare è rappresentato dalla cosiddetta Object Detection, ossia dalla necessità di identificare, nella maniera più robusta possibile, posizionamento e dimensione di oggetti interessanti all'interno di immagini o video.
Tale problema, benchè oggetto di moltissimi studi durante il corso degli anni, risulta ancora oggi essere soltanto parzialmente risolto a causa delle molteplici difficoltà e problematiche che esso presenta. Esempi tipici in tal senso sono dati da:

- Le molteplici condizioni ambientali in cui il sistema può dover operare, che affliggono aspetto e colore dei target.
- Le molteplici forme o strutture che istanze di una medesima classe di oggetti possono presentare.
- Le possibili occlusioni totali o parziali a cui i target possono essere sottoposti.

Allo stato dell'arte sono disponibili moltissime soluzioni per far fronte a questa situazione. In special modo, architetture basate su Deep Neural Network stanno suscitando grande interesse per i risultati che permettono di raggiungere. Queste reti neurali profonde infatti, grazie alla loro struttura vasta e complessa, sono in grado non soltanto di far fronte a tutte le problematiche appena esposte, ma anche di riconoscere adeguatamente molteplici classi di oggetti. Esse rappresentano in tal modo una soluzione flessibile, robusta ed unificata al problema presentato, anche se computazionalmente pesante a causa delle loro dimensioni.

Obiettivo di questo lavoro è quindi quello di realizzare una architettura di questo tipo basata su Deep Neural Network e mirata ad estenderne il funzionamento ad un contesto real-time, dove oltre all'accuratezza dei risultati prodotti anche la velocità delle soluzioni proposte gioca un ruolo fondamentale.

Abstract

In the last years a lot of improvements have been achieved in the Computer Vision field thanks to new and powerful parallel architectures, which can process huge amounts of data in short times, giving the possibility to develop and study new solutions otherwise not achievable.

One of the typical problems that must be solved in this research area is the so called Object Detection Problem, i.e. the need to identify, in the most robust way, position and size of objects in images and videos.
Despite the great efforts made by a lot of researchers to solve the detection problem, today, we still cannot consider it as completely solved because of the multiple issues and different conditions that may arise for each different image. Typical examples in this sense are:

- The various environmental conditions where the systems can work, which affect aspects and colors of the targets
- The different appearances that can be shown by distinct objects of the same class
- The possible occlusions that a target can suffer

At the state of the art a lot of different solutions are available that try to solve this problem. In particular, solutions based on Deep Neural Networks are arousing nowadays great interest for the results that they are able to achieve. These networks, thanks to their structure and complexity, are in fact not only able to face all the issues previously mentioned but also to carefully detect objects belonging to different classes, representing in this way a flexible, robust and unified solution to the detection problem. However, due to their wide and deep structure, these solutions typically show high computational loads, presenting in this way a reduced appeal for what can concern any real time application.

The goal of this thesis is the realization of a systems based on this kind of architectures and aimed at extending their typical offline application to real-time scenarios, where in addition to the accuracy of the information extracted also the speed of the implemented solutions plays an important role.

Acknowledgements

Foremost, I'd like to give a special thanks all the people that supported and helped me in the last months during the realization of this work, in particular my two professors Marco Tagliasacchi and Luigi Piroddi, which have been a reference point for comparisons and suggestions, whenever I needed their help. Luca Bondi and Luca Baraffio that supervised me step by step and Denis Tomè whom I shared the first part of this work.

I would like furthermore to express my gratitude to Enrico Busto (Addfor - CTO), who gave me all the tools I needed to realize the final architecture that will be explained in the following and all the Addfor staff I met during the realization of this thesis.

Besides my advisors, I would like to give a special thanks to my family that allowed me to reach this point in my life, pushing me to pursue my goals and ideas in every single moment.

Last but not least, an important thanks goes to all the friends that walked by my side in these years, without whom I'm sure all this would not have been possible.

Introduction

In the field of Computer Vision, object detection is a problem that has been faced by a lot of researchers in the years due to the complexity and variability this kind of task presents. Multiple solutions are so available in the literature that can be distinguished for both the accuracy and speed they are able to achieve.
However, whenever there is the need to deal with complex situations as can be the detection of pedestrians from moving cars, due to the reduced quality the acquired images can present, fast detector may turn out to be inaccurate and so much stronger and unfortunately slower detectors need to be implemented.

Giving a deeper look at the object detection problem, it turns out that several approaches have been presented in the last decade that are capable of facing a lot of different conditions, which can preclude the quality of the produced results.
Detectors based on Haar features and Boosted Decision Tree (as the Viola&Jones detector [1]) or on Histograms of Oriented Gradients and Support Vector Machines (HOG+SVM [2]) have been the reference detectors for several years thanks to the simple and powerful structures they implement.
Unfortunately, despite the good capabilities of these detectors to identify simple objects of a particular class in good environmental conditions (good light conditions, no occlusions...), this kind of solutions turns out to be inefficient if applied to more difficult situations where a lot of distortions are involved.
In order to face these more complicated tasks, other solutions based on multiple feature channels that present higher representational power have so been implemented.
In particular, the so-called ACF (Aggregated Channel Features Detector [3]) and its extension LDCF (Local Decorrelated Channel Features [4]) represent nowadays powerful solutions that deserve to be analyzed for the innovations they introduce.
These two detectors in fact, thanks to the approximation of features they apply, achieve

good multi-scale detection capabilities with sufficiently reduced computational times, representing a valid and cheap solution for several applications.

Anyway, the detection problem not necessarily concerns every time the identifications of objects belonging to a single class of interest. In a lot of different situations multiple types of objects may in fact be involved.
Think for example to surveillance systems, it could be of interest to identify if a car or a person is violating a fixed security policy.
In such situations single class detectors, as the ones presented so far, would not be capable at realizing an effective detection, simply because they cannot distinguish the different classes where the objects belong to. Therefore, more powerful and complex solutions need to be taken into account for facing such complicated scenarios.

In this sense, a solution that appears promising at present is provided by the so-called DNNs (i.e. Deep Neural Networks), which achieved in the last years incredibly high performances in image classification and detection tasks, outperforming the previous state of the art even in single-class detection processes [5].
This kind of networks, thanks to their really wide and large structure, can in fact extract reduced and representative sets of features that allow to identify in a very robust way the class each single object belongs to, simply recurring to linear Support Vector Machine (SVM) [6]. However, this comes at a cost of a high computational burden, which hinders the application of this kind of solutions to real time applications.

Following this considerations, the present work is aimed at illustrating a possible way to realize robust and efficient DNN detectors, introducing:

- A refined learning process for training and exploiting such strong and complex classifiers.
- A possible architecture capable at combining multiple detection solutions recurring to a suitable tracking system, in order to produce accurate results at more than 10 fps over the selected HW architecture.

Chapter 1 provides a presentation of the pedestrian detection problem that will be the reference task of this work, the main measures exploited to compare different solutions and the available datasets that can be used to generate good and robust detectors.

Chapter 2 recaps the state of the art, outlining pros&cons of each analyzed solution and the associated computational loads.

Chapter 3 presents the learning method explored to train a Deep Neural Network capable at achieving high-level performance.

Chapter 4 presents the proposed architecture, highlighting the innovations introduced and the results achieved.

Finally, in chapter 5 some conclusions are drawn and possible extensions plus further works are presented.

Chapter 1

Introductive Steps

1.1 Problem Introduction

As already introduced in the previous Chapter, object detection is a really large and challenging task which can present really different drawbacks depending on the context in which it may be applied.
Objects in fact may radically change both in term of colors, shapes, scales and in general appearances; imposing the necessity to have strong and robust solutions capable at dealing with all the different aspects and dimensions the targets may present.

In order to deal with these particular situations, researchers looked in the years for more and more accurate ways for recognizing the presence and locations of desired objects in a given image.
Multiple solutions have so been presented in this sense but all based on a common approach of 3 steps:

1. Promising regions extraction (where a set of regions that can potentially contain objects of interest are retrieved).
2. Features extraction (where a set of features, capable at representing in a similar way object belonging to the same class, are fetched from all the regions identified).
3. Regions Classification (where a suitable trained classifier is applied for discriminating regions of interest from background ones).

The main idea is to extract a set of possible regions that can contain objects of interest, retrieve a simple and compact high level representation of such patches and finally apply a suitable classifier capable at differentiating background regions from desired ones.

Among all the possible scenarios however, one of the most interesting and difficult context that have been faced in the literature is represented by the so-called Pedestrian Detection task.
Detecting pedestrians, in fact, can be a really though challenge due to:

- The multiple aspects and poses people can assume while walking.
- The bad conditions in which the acquisition process can be realized (think for example to images acquired from a moving car)

Various detectors have been realized in the years that allows to achieve appreciable results also in this context, however, the pedestrian detection task appears so various and heterogeneous that even nowadays it still remains an open research area, which deserves to be analyzed.

For the difficulties this scenario presents and the amount of researches available in the literature, we decided to focus our attention over this particular problem.
All the considerations that will be realized in the following, however, should not be considered as strictly related to the selected class of targets, but can be easily extended to any other possible scenario simply retraining the implemented classifiers.

1.2 Reference Measures

Independently of the algorithm that one may decide to implement and the context of application, in order to check and compare the performance produced by different detectors, suitable measures capable at representing the detection capabilities of the selected solutions need to be taken into account. A common and well-defined option, which is widely used in the literature, is represented in this case by the so-called Receiver Operating Characteristic (ROC).

The ROC basically illustrates the performance of a detection system, plotting the miss rates (i.e. the fractions of missed pedestrians) produced at different confidence levels against the associated false positives per image (fppi).

In order to discriminate interesting objects from background in fact, detectors assign to every analyzed region a score that is proportional to the probability that something of interest is situated in that particular location.
Thresholding at different levels these scores and checking the obtained results, the performance produced by the selected detectors can then be compared, highlighting how each single solution is capable at discriminating background regions from desired targets with scores of different magnitude (e.g. figure 1.1).

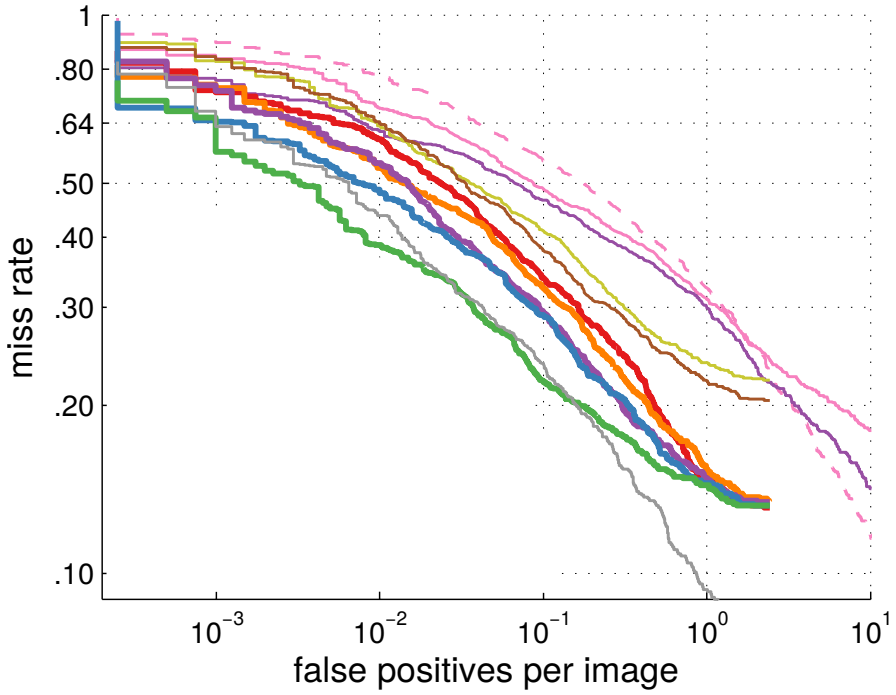

FIGURE 1.1: ROC for various algorithms and associated LAMRs.

Curves, however, do not represent immediate and clear indications of the performance presented by different solutions. Therefore, in order to compare in a really direct and simple way the detection capabilities produced by different architectures, valuable scalar values are also typically introduced.

Following this reasoning, the Receiver Operating Characteristic is usually coupled with another suitable measure, the so-called Log Average Miss Rate (LAMR, eq. 1.1).

$$LAMR = e^{\sum_{i=1}^{N} log(miss\ rate_i)} \tag{1.1}$$

where miss rates$_i$ is the i-th miss rate extracted from the associated ROC at the i-th fppi.

The main idea of the LAMR is to represent the mean capability to detect all the existing pedestrians in every analyzed frame basically sampling the ROC at various fppi and averaging the returned miss rates (figure 1.2). The measure produced in this way represents a compact indication of the performance produced by the selected detectors, giving the possibility to discriminate the various architectures in a very simple and straightforward way.

Typical fppi from which the required miss rates can be extracted are:

$$fppi = [0.0100\ 0.0178\ 0.0316\ 0.0562\ 0.1000\ 0.1778\ 0.3162\ 0.5623\ 1.0000] \tag{1.2}$$

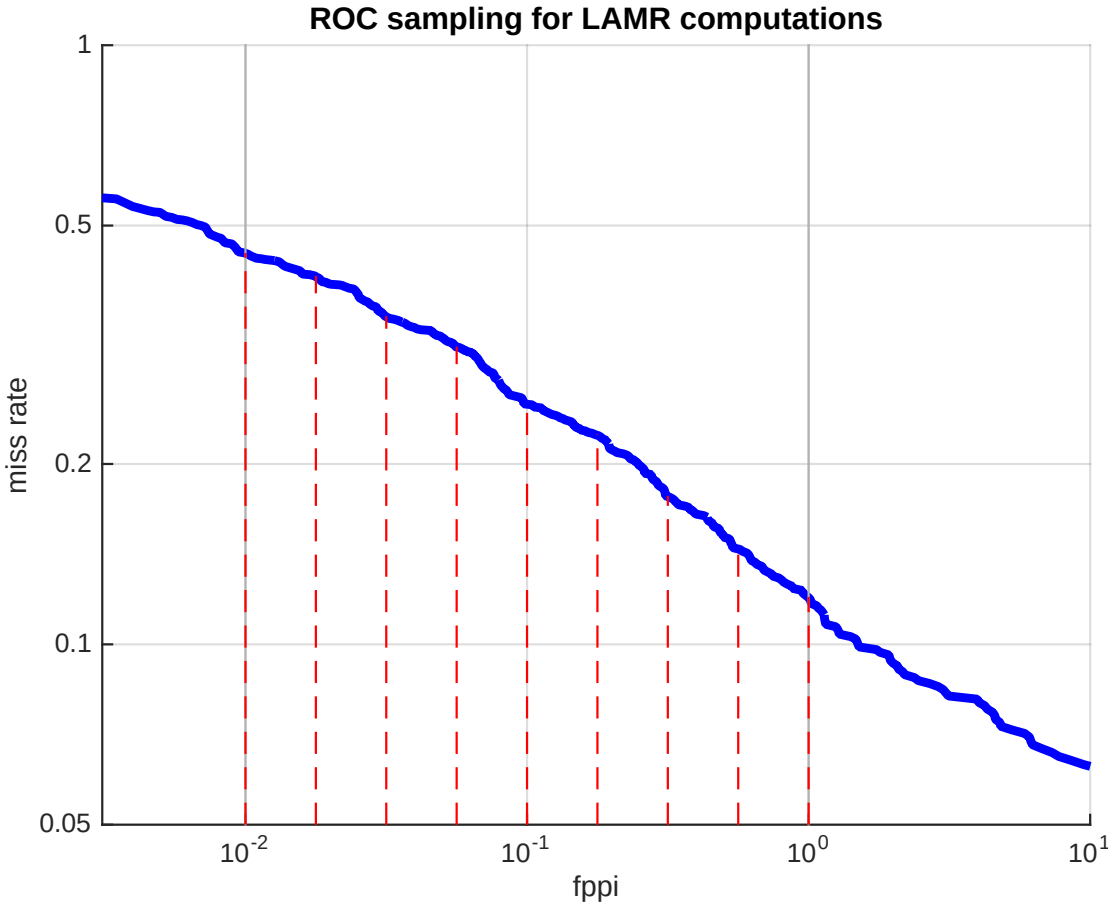

FIGURE 1.2: Extraction of the miss rates required for computing the LAMR from the associated ROC.

1.3 The Considered Datasets

Any solution that can be introduced for identifying the location of any objects of interest needs a reference dataset over which learn the aspects of the desired targets.
Therefore, before introducing any kind of state of the art solution that allows detecting pedestrians, a dedicated section related to the dataset available in the literature, which give the possibility to build and test the selected detectors, needs to be realized.

A lot of possibilities are available in this sense thanks to the numerous challenges that every year take place.
Typical examples are the famous ImageNet Dataset and Caltech Pedestrian Dataset, which are particularly suitable to train large and complex structures thanks to the huge and various content they present.
In particular, the ImageNet Dataset represents up-to-now one of the reference dataset for what concerns multi-class object detection and image classification. It is built by two main subset:

- The Object Detection Dataset.
- The Image Classification Dataset

Which respectively contain sets of images that have been carefully annotated for giving a solid ground truth for both the detection and classification tasks.

In the Object Detection Dataset, 200 different fully-annotated object categories have been carefully chosen (figure 1.3) considering a set of different factors. Such as:

- Object scale
- Level of image clutterness
- Average number of object instances
- ...

While in the Image Classification Dataset, 15 million high-resolution images belonging to roughly 22,000 categories have been collected and labeled recurring to the Amazon's Mechanical Turk crowd-sourcing tool.

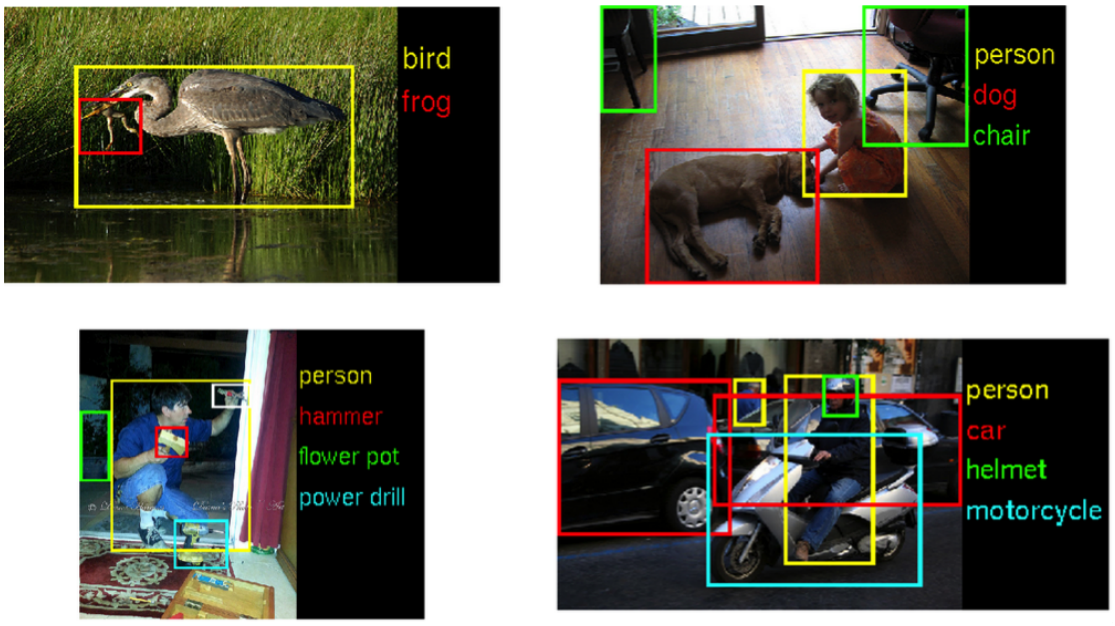

FIGURE 1.3: Examples of bounding boxes and associated classes in the ImageNet Dataset.

Independently by the particular subset in which one may be interested into, thanks to the variability and the amount of information available in the ImageNet dataset, up-to-now it represents one of the reference dataset for both Object Detection and Image Classification tasks.

However, as we have presented few sections ago, the main focus of this work concerns pedestrian detection. One of the most famous and exploited dataset in this sense is the so-called Caltech Pedestrian Dataset.
This particular set of images basically consists of approximately 10 hours of 30Hz video ($\sim 10^6$ frames) taken from a vehicle driven through regular traffic in an urban environment. The video resolution is 640×480, and, not unexpectedly, the overall image quality is lower than that of other images of comparable resolution.
The driver was independent from the authors of the dataset and had instructions to drive normally through neighbourhoods in the Los Angeles metropolitan area, which was chosen for its relatively high concentration of pedestrians.
250,000 frames (in 137 approximately minute long segments extracted from the 10 hours of video) have been collected and annotated for a total of 350,000 bounding boxes around 2,300 unique pedestrians.
In order to make "the movement" of the bounding boxes (BBs) smooth frame by frame, a cubic interpolation of the coordinates has been applied.

For every frame in which a given pedestrian is visible, annotators mark two different BBs:

- A BB that indicates the full extent of the entire pedestrian (for occluded pedestrians this involves estimating the location of hidden parts).
- A BB that delineates only the visible region.

During an occlusion event, the estimated full BB stays relatively constant while the visible BB may change rapidly (see figure 1.4).

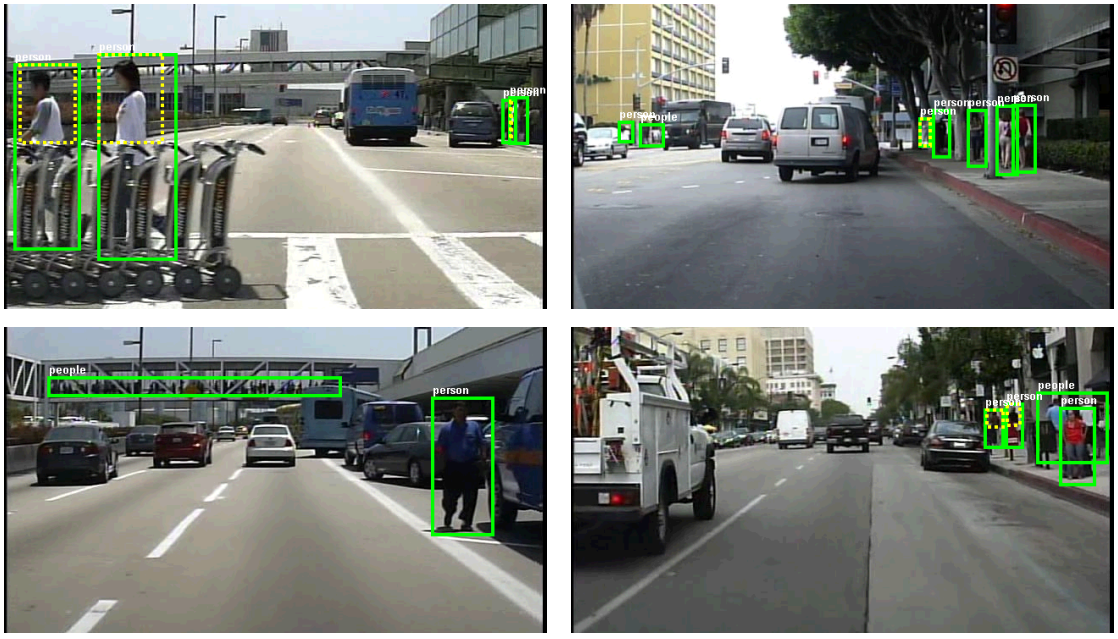

FIGURE 1.4: Examples of full bounding boxes and visible bounding boxes in the Caltech Dataset.

For each sequence of BBs belonging to a single object:

- Individual pedestrians were labelled as 'Person' (~1,900 instances).
- Large groups for which it would have been tedious or impossible to label individuals were delineated using a single BB and labelled as 'People' (~300).
- Ambiguous or easily mistaken instances were labelled as 'Person?' (~110).

This classification is of fundamental importance for the realization of any kind of detector, since it allows to discard samples that may alter the final results due to their strange classification nature (i.e. the 'Person?' ones).

The Caltech Pedestrian Dataset plays a fundamental role in the literature for the number of solutions that have been tested and compared over these sets of images and for this, it will represent the reference dataset also in this particular work.

Chapter 2

Object Detectors: a Review

2.1 Introduction

Once we have introduced the fundamental datasets and indices available in the literature (i.e. LAMR and ROC), the following step of our analysis concerns a revision of the main state of art detectors that could be applied in our situation.
Since it's not possible to analyze every single available architecture presented in the literature, in the following, only a description of the main available detectors that have excelled in the years for accuracy or efficiency will be carried on.

Our revision starts with the famous Viola&Jones algorithm, which is useful to introduce how the pedestrian detection problem has been initially faced, reviewing the concepts of feature representation and of Boosted-classifier (which is a fundamental part for many new state of the art detectors).
It proceeds with the HOG+SVM detector, which allows to introduce the multiple pose and aspect problem that may jeopardize the detection performance if not suitably managed.
Then, two of the main state of the art detectors (ACF and LDCF) will be presented, reviewing how they allow to solve the problems presented by HOG+SVM recurring to feature approximation and Boosted-classifier.
Finally, an illustration of the main part of this work will be realized, reviewing some of the most famous DNN architectures currently available in the literature and explaining the rationale behind them.

2.2 Viola&Jones Detector

One of the most famous detectors available in the literature is the so-called Viola&Jones detector [1]. This detector, proposed in 2001 by Paul Viola and Michael Jones, has been the first object detection framework to provide competitive object detection rates in real-time, introducing a new and easy way to approach this kind of task (which represents nowadays a consolidated baseline for any new possible proposal).
The main idea of the Viola&Jones detector is to apply a boosted binary classifier, made by a cascade of simple binary classifiers, over a set of features that represents the overall image behavior.
A feature is a mathematical compact representation of a particular image behavior that can be exploited to correctly classify regions' proposal, even if subjected to particular variations (e.g. magnitude of the gradients or histogram of oriented gradients are independent by changes in the average luminance). The choice of the best features to use, is strictly problem dependent.

There are many motivations for using features rather than the pixel values directly. The most common reason is that features can encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. In the Viola&Jones framework there is also a second critical motivation for features: a feature-based system operates much faster than a pixel-based one.

The cascade structure of the overall classifier is meant to realize a complex and powerful classifier recurring to a combination of simple binary classifiers (aka weak/stump classifier), which are slightly better then a toss of coin.
The application of simple classifiers in different layers allows in fact to check different properties of the input image at different levels, producing a much more robust and accurate classifier than the one which could be realized with only one stage.
In particular, the cascaded architecture has interesting implications for the performance of the overall classifier. Analyzing in fact the false positive rate (i.e. the rate with which we classify a negative sample as positive), it turns out that, since the activation of each simple classifier depends entirely on the behavior of its predecessor, the false positive rate for an entire cascade is:

$$F = \prod_{i=1}^{K} f_i \tag{2.1}$$

where F is the overall false positive rate, f_i is the false positive rate of the i-th classifier and K is the overall number of stage in the cascade.

and similarly the detection rate (i.e. the rate with which we classify a positive sample as positive) for a cascaded architecture can be computed as:

$$D = \prod_{i=1}^{K} d_i \tag{2.2}$$

where D is the overall detection rate, d_i is the detection rate of the i-th classifier and K is the overall number of stage in the cascade.

Thus, to match the false positive rates typically achieved by other detectors, each classifier can get away with having surprisingly poor performance. As example, for a 32-stage cascade to achieve a false positive rate of 10^{-6}, each classifier needs only to achieve a false positive rate of about 65%.
At the same time however, each elementary classifier needs to have a very high recall[1] in order to produce an overall detector capable at achieving satisfactory detection rates. For example, to produce a global detection rate of about 90%, each classifier in the aforementioned cascade needs to present a detection rate of approximately 99.7%.
Ideally, each stump classifier should discard only a reduced set of samples that clearly do not belong to the desired regions, producing in this way a high detection rate and leaving to the subsequent detectors the due to discard any other possible negative region remaining in the set.

Anyway, in order to build the cascade of simple detectors here presented, a suitable learning algorithm requires to be exploited. To efficiently solve this particular requirement, a new solution based on a variation of the AdaBoost meta-algorithm has been introduced by Viola and Jones in [1] .

AdaBoost [7], short for "Adaptive Boosting", is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, which can be used in conjunction with many other types of learning algorithms to improve their performance. The main

[1]Recall (also known as sensitivity or true positive rate) is the fraction of all relevant instances retrieved.

idea is that several weak classifiers trained with different learning algorithms can be combined into a weighted sum that represents the final output of the boosted classifier. As long as the individual classifiers are slightly better than random guessing (i.e., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner.

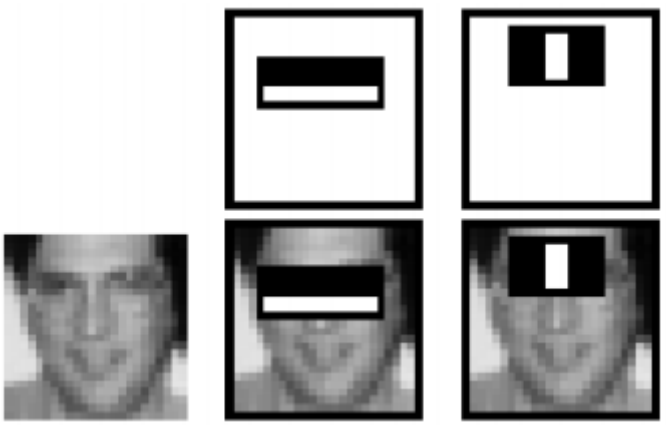

FIGURE 2.1: Example of two features selected by the Viola&Jones learning algorithm

Following this idea, the variation introduced by Viola&Jones constructs a "strong" classifier as a linear combination of specific weighted simple "weak" classifiers (eq. 2.3), aimed at checking only some particular features which are selected during the training process (see figure 2.1).

$$h(\boldsymbol{x}) = sign(\sum_{j=1}^{M} \alpha_i h_j(\boldsymbol{x})) \tag{2.3}$$

where $\boldsymbol{x}$ is the input patch, $h(\boldsymbol{x})$ is the final output of the strong-classifier, $h_j(\boldsymbol{x})$ is the output of the j-th weak classifier and α_i its associated weight (which determines its relevance).

Each weak classifier is a threshold function based on the feature f_j (eq. 2.4), where the threshold value θ_j and the polarity $s_j \in \{1, -1\}$ are determined in the training, as well as the coefficients α_j.

$$h_j(\boldsymbol{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases} \tag{2.4}$$

where f_j is the j-th feature extracted from the input patch $\boldsymbol{x}$, θ_j is a threshold applied over the value of the feature in order to establish the value of the the output (equal in absolute value to the polarity s_j).

To simplify the explanation, a simplified version of the learning algorithm is here presented.

Given a Set of N positive and negative training images with their labels $(\boldsymbol{x_i}, y_i)$, for each stage of the cascade that we want to produce the following procedure is applied:

1. A weight $w_1^i = \frac{1}{N}$ is initially assigned to each image i.
2. For each required feature j with $j = 1..M$ (where M is the overall number of features we want to test in a stage):
   (a) The images weights w_j^i are renormalized such that they sum to one.
   (b) For each possible feature k, the optimal threshold and polarity θ_k, s_k of the related weak classifier are computed[2] and a weighted error E_k, representing the goodness of this latter, determined:

$$E_k = \sum_{i=1}^{N} w_j^i \epsilon_k^i \tag{2.5}$$

$$\epsilon_k^i = \begin{cases} 0 & \text{if } y^i = h_k(\boldsymbol{x}^i, \theta_k, s_k) \\ 1 & \text{otherwise} \end{cases}$$

   where ϵ_k^i is the classification error produced by the weak classifier k over image i.

   (c) The classifier that produces the minimum value of E is chosen and a weight α_j that is inversely proportional to the error rate is assigned.
   Since each classifier is related to a specific feature, a selection of the feature to test is basically accomplished at this step.
   (d) The weights for the next iteration (i.e. w_{j+1}^i) are finally reduced for the images that were correctly classified[3] and the algorithm returned to point 2a.
3. The strong classifier of the current stage is set equal to:

$$h(\boldsymbol{x}) = sign(\sum_{j=1}^{M} \alpha_j h_j(\boldsymbol{x})) \tag{2.6}$$

[2]The computation of the classifier's parameters depends only on the training algorithm that has been selected.

[3]Images with wrong classifications must appear more important in the following steps, in order to improve the discriminability the overall strong-classifier.

4. A new training set is constructed containing only the samples that passed the previous steps and the overall learning procedure repeated over this new set.

Analyzing the proposed algorithm however, it turns out that a large portion of computational effort is not spent in the classification phase, but instead in the computation of the features that represents the image behavior. In order to achieve real-time performance therefore, a suitable set of features, fast to compute, must then be considered.
In order to face this problem, Viola&Jones introduced in their work the so-called Haar-Like features.
Haar features are simple local representations that consider adjacent rectangular regions in a detection window, summing up the pixel intensities in each region and calculating the differences between these sums.

Despite their simplicity, the amount of features that may be required to be computed in a given region can be really large. So, in order to speed-up the overall process, Viola&Jones introduced in their work a further improvement characterized by the so-called Integral Images.

Integral images can be defined as two-dimensional lookup tables in the form of matrices with the same size of the original images, where each element contains the sum of all pixels located on the top-left part of the figures themself.
Thanks to this alternative representation, the sum of rectangular areas in the image can now be computed only recurring to four lookups (see figure 2.2), exponentially reducing the cost of the overall feature extraction.

Despite the goodness and simplicity of this solution, the Viola&Jones detector presents some crucial disadvantages, which make it hardly applicable in the detection of complex and heterogeneous objects.
In fact, while it is extremely fast, scale invariant and rotation invariant, it is mostly effective only with frontal images and can hardly cope with 45° rotation both around the vertical and horizontal axis. It is moreover sensitive to lighting and environmental conditions, making the detection harder every time there are changes in the operative conditions.
In order to achieve a robust detection even in more difficult situations, more powerful and complex solutions have been introduced in the literature.

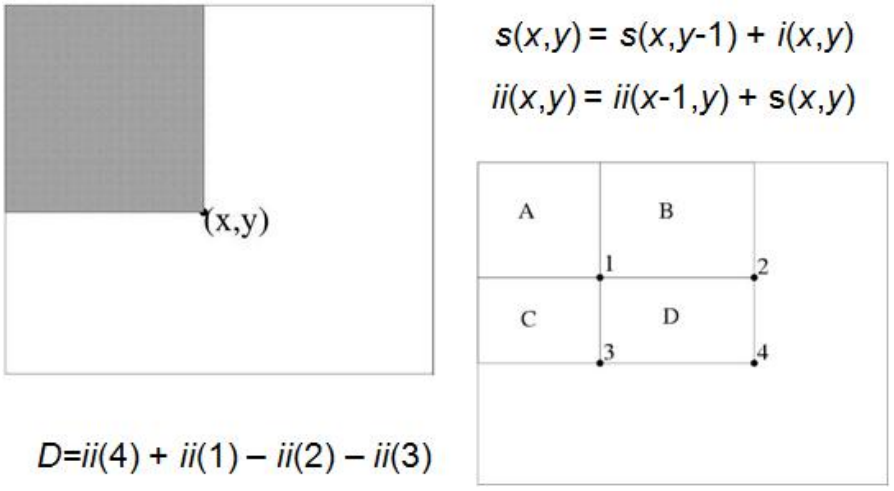

FIGURE 2.2: Example of integral image, where $s(x, y)$ represents the sum of the previous column-pixel and $ii(x, y)$ represents the sum of the all the pixels situated among the origin and the point (x, y)

2.3 HOG+SVM

Continuing with our illustration of the most famous and powerful detectors available in the literature, one of the solutions that for sure deserves to be reviewed in our analysis is the so-called HOG+SVM detector [2] (i.e. Histogram of Oriented Gradients + Support Vector Machine), introduced in 2005 by Navneet Dalal and Bill Triggs for the identification of pedestrians in static images.

In detail, the rationale behind the histogram of oriented gradients descriptor is that local object appearances and shapes can be described by the distribution of intensity gradients (i.e. edge directions).
Basically, dividing a promising region into small connected regions (i.e. cells) and computing an histogram of gradient directions for each of them (see figure 2.3), a global descriptor of a proposed region can be obtained simply concatenating the bins extracted from each retrieved local representation and vectorizing the overall representation.

The feature vector produced in this way gives a compact representation of the overall region content that can be exploited for detecting the presence or absence of interesting objects, simply applying a suitable classifier as a SVM.

To give a better understanding of the behavior the detector presents, an example of pedestrian detection is here considered, which explains the typical pipeline applied with the HOG+SVM detector.

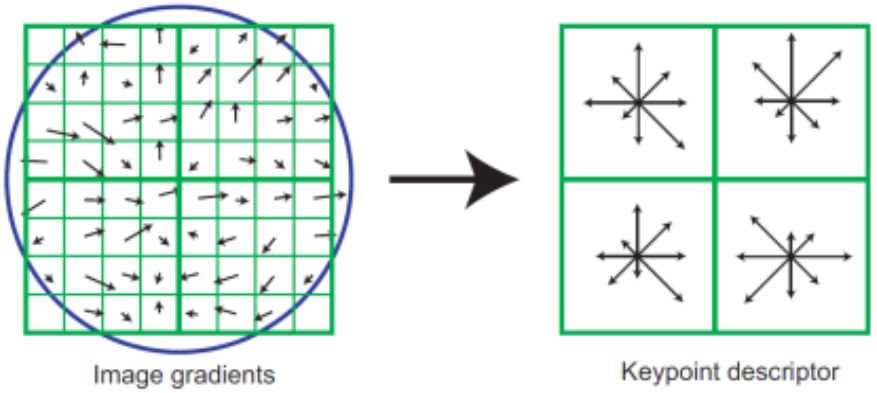

FIGURE 2.3: Example of gradient histograms extracted from 4 subregions

Given an input image where to detect the occurrences of possible pedestrians, all the available 64×128 subregions are extracted in a sliding window manner (see figure 2.4). Each proposal is divided in subsequent 8×8 cells and for each cell a local histogram of oriented gradients with 9 possible orientations is computed (see figure 2.5).

FIGURE 2.4: Example of one of the possible 8x8 cells that can be extracted from one of the 64×128 windows extracted from the image (luckily, one that contains a pedestrian).

For each gradient vector, its contribution to the histogram is given by the magnitude of the vector (stronger gradients have a bigger impact on the histogram) and is split

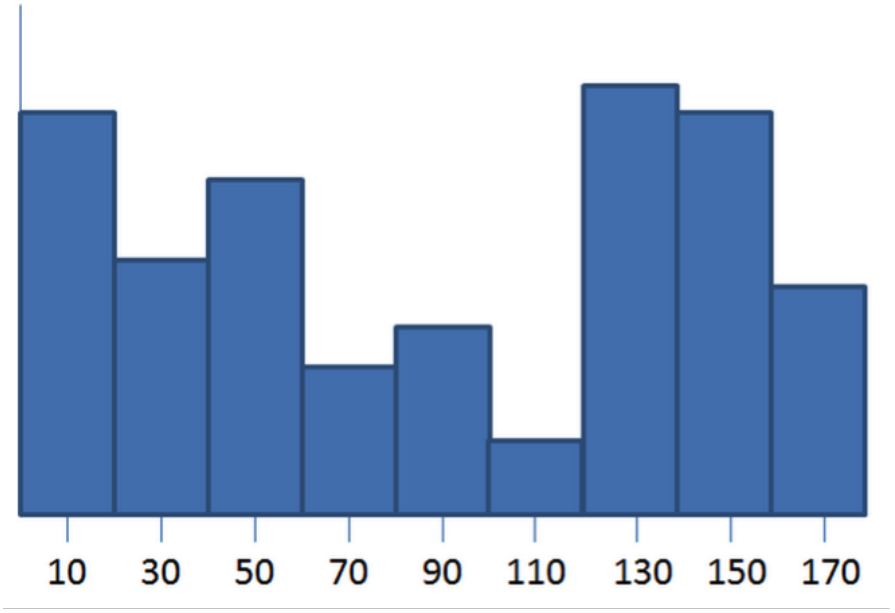

FIGURE 2.5: Example of a possible orientations histogram extracted from an 8×8 cell. Dalal and Triggs used "unsigned gradients" in their original version, such that the orientations only ranged from 0 to 180 degrees instead of 0 to 360.

between the two closest bins.
So for example, if a gradient vector has an angle of 85 degrees, then 1/4 of the magnitude is added to the bin centered at 70 degrees, and 3/4 to the bin centered at 90.

The various extracted histograms are then normalized in-group, in order to make the description more robust to possible variations (such as multiplicative noise).
The cells are in fact aggregated in blocks of 2×2 cells, with 50% of overlaps (see figure 2.6), concatenated in a unique vector of $9 \times 4 = 36$ bins and then normalized, dividing this overall vector by its magnitude.
Thanks to the block overlapping, each cell appears multiple times in the final descriptor and is normalized by different sets of neighboring cells. This allows to take into account all the multiple variations that may affect the proposed region in a neighborhood of the analyzed point.

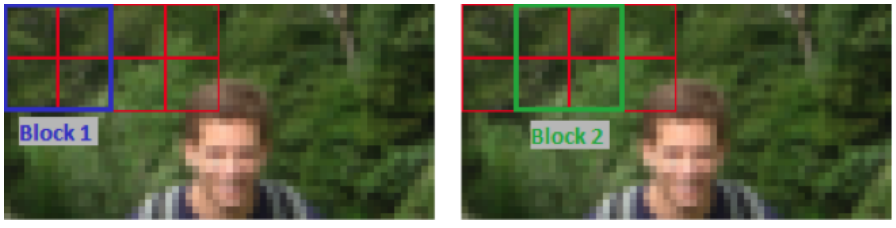

FIGURE 2.6: Example of cells aggregation

The overall description of a proposal is finally produced and built by a set of subsets of histograms. Each subset represents the local histogram of an overlapping block extracted from the image.
Therefore, for a 64×128 image, 7×15 overlapping 8×8 blocks are extracted, each one built by 2×2 cells and described by 4 local histograms, for a total of: $7 \times 15 \times 4 \times 9 = 3780$ bins.
A SVM is finally applied on these data to obtain the predicted class of the analyzed proposal.

The HOG+SVM detector shows sufficiently good results for pedestrian detection in good quality images, reducing false positive rates by more than one order of magnitude compared to the best Haar wavelet based detector.
For this reason, this solution became really popular in time, spurring many developers to produce highly optimized implementations of this algorithm, such as the one proposed in the OpenCV library, which can process an image at different scales (applying different image resizing) in 80 ms on a compact and embedded board such as the Nvidia Jetson TK1.

Despite the power of the HOG descriptor, the presented detector exposes some limitations that reduce the overall performance of the detection task, especially in low quality images.
A single bounding box of a pedestrian for example, can vary its aspect considerably wrt to the pose that a person can present, reproducing a considerable variation of the HOG descriptor that makes the application of the classifier less accurate.
For this reason, different and more powerful detectors based on a set of multiple features have been studied during the years.

2.4 ACF

The Aggregated Channel Features detector (aka ACF [3]) is one of the most famous detectors available nowadays. It combines good single-class object detection and contained computational times, recurring to approximations of the features values at multiple scales.
This type of detector, thanks to the utilization of multiple features channels (i.e. multiple representations of the input image), is capable of outperforming the performance

of HOG+SVM, even distinguishing pedestrians in multiple poses and aspects without incurring in high computational loads.

The main idea behind ACF is to compute a rich representation of an image made by a set of 10 features channels:

- Normalized Gradient Magnitudes (1 channel).
- Histogram of Oriented Gradients (6 channels).
- LUV Color Channels[4] (3 channels).

paying at the same time only a reduced computational price.

The key insight is that natural images have fractal statistics [3] (each part of the image has the same statistical properties as the whole image). Therefore, it's possible to reliably predict the image structure across different scales.

Dollar et al. demonstrates in [3] that the behavior of shift-invariant features (features that present the same behavior even if the image is shifted) at multiple scales is governed by a power law.
Basically, letting $\phi(I)$ denoting an arbitrary (scalar) image statistic and $E[\cdot]$ the expected value over an ensemble of natural images, Ruderman and Bialek made the fundamental discovery that the ratio of $E[\phi(I_{s1})]$ to $E[\phi(I_{s2})]$, computed over two sets of natural images at scales s1 and s2, is equal to:

$$E[\phi(I_{s1})]/E[\phi(I_{s2})] = (\frac{s1}{s2})^{-\lambda_\phi} \tag{2.7}$$

Where every statistic has its own corresponding λ_ϕ.

The presented equation gives unfortunately only the behavior of a statistic at different scales computed over an ensemble of images, but does not define the value of features computed for a single image at multiple scales.

However, since a single image can be decomposed into a set of K single patches (i.e. sub-images) such that $I = [I^1 \dots I^K]$ and since we are considering shift-invariant features, then the value of features computed in a single region, ignoring boundary effects, is approximately independent from the value of pixels in another region.

[4]For reference see the CIE 1976 (L*, u*, v*) color space.

A features map Ω (i.e. a set of channels representing the output of a shift-invariant transformation applied over the original image), from which a final shift-invariant feature f_Ω can be computed, can in fact be approximated as:

$$\Omega(I) = \Omega([I^1 \dots I^K]) \approx [\Omega(I^1)...\Omega(I^K)] \tag{2.8}$$

exactly as if we had K different images instead of only one.

Considering then that a single global feature (or a local feature determined in the same way but only on a subregion of the overall image) defined as:

$$f_\Omega(I_s) \equiv \frac{1}{h_s w_s k} \sum_{i,j,k} C_s(i,j,k) \text{ where } C_s = \Omega(I_s) \tag{2.9}$$

where I_s the image I at the scale s, h_s and w_s its height and width, k the number of layers of the channel map.

can be computed as:

$$f_\Omega(I) \approx \frac{1}{K} \sum_{k=1}^{K} f_\Omega(I^k) \tag{2.10}$$

where the approximation is due to the boundary effects introduced in the the channel maps.

if all the K regions have similar size. It is possible to write a relation among features extracted at multiple scales of a same image combining eq. 2.10 and 2.7:

$$\frac{f_\Omega(I_{s1})}{f_\Omega(I_{s2})} = (\frac{s1}{s2})^{-\lambda_\Omega} + \Sigma \tag{2.11}$$

where we use Σ to denote the deviation from the power law for a given image.

However, in order to apply such equation, the value of the parameter λ needs first to be determined.

Since λ changes from feature channel to feature channel (but not from class of objects to class of objects[5]), its value can be calculated using a simple measure μ_s defined as:

$$\mu_s = \frac{1}{K} \sum_{i=1}^{K} \frac{f_\Omega(I_s^i)}{f_\Omega(I_1^i)} \tag{2.12}$$

that according to equation 2.11, must be equal to:

$$\mu_s = s^{-\lambda_\Omega} + E[\Sigma] \tag{2.13}$$

Putting therefore together eq. 2.13 and 2.12 the value of λ can be easily computed simply reversing the formulae.

Once the value of λ is determined, the last step that remains to be faced is how to apply in practice what we have theoretically explored so far.

Typically, in order to compute features maps at different scales, the standard approach requires to compute:

$$C_s = \Omega(R(I, s)) \tag{2.14}$$

where $R(I, s)$ is the resizing of the image to scale s.

ignoring the information contained in $C = \Omega(I)$.
What Dollar et al. propose instead is to apply the following approximation:

$$C_s \approx R(C, s) \cdot s^{-\lambda_\Omega} \tag{2.15}$$

and aggregate the values of different pixels smoothing and downsampling the estimated feature space, in order to improve the robustness of the overall representation.

This approximation allows to build a features pyramid of an input image (which is a multi-scale representation where channels $C_s = \Omega(I_s)$ are computed at every scale s) determining the value of features at one scale per octave through image resizing (an

[5]Analyzing in fact the values of μ_s computed at different scales ($s = 2^{-1/8}, ..., 2^{-24/8}$) for both pedestrian and natural images, it is possible to observe as the value of λ is basically the same for pedestrian objects and also natural images.

octave is an interval between one scale and another with half or double its value) and approximating all the other feature channels in the middle.
Since the cost of approximating C_s is typically 1/3 of computing C_s at the original scale, the computational burden of the overall process is in this way incredibly reduced.

In order to robustly detect the presence or absence of pedestrians in the given regions, Dollar et al. finally suggest to apply a cascade classifier made of 2048 depth-2 boosted decision trees over the features pyramid determined as just explained.
Due to the simplicity of such classification process however, the presented detector doesn't exploit all the power of the available extracted features set, since it applies only orthogonal splits over a set of features virtually correlated one to each other.

2.5 LDCF

The locally decorrelated channel features detector (aka LDCF [4]) is an extension of the ACF-detector introduced by Dollar et al. in 2014 to solve the limits exposed by their previous solution.

Analyzing the behavior of the boosted decision tree applied by ACF over the feature channels in fact, it is possible to observe as the split applied by each stump classifier (each node of a depth-2 decision tree) does not follow necessarily the data distribution, but is orthogonal to the channels extracted (figure 2.7).
Due to this sub-optimal classification, the quality of the final detections is therefore limited and more complex solutions needs to be explored, in order to further improve the performance achieved.

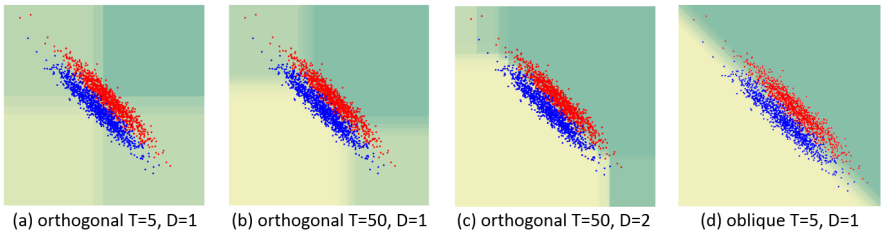

FIGURE 2.7: Comparison of orthogonal vs oblique boosted decision tree at variations of the number of trees in each stage (T) and depth of the overall boosted tree(D). Regions with different intensities represents areas assigned to different classes (e.g. the lighter the area, the stronger the assignment to the blue class). As it is possible to see, even with a small amount of decision trees in a stage, the oblique boosted decision tree outperforms all the boosted orthogonal decision trees here presented.

A possible solution in this sense could be the realization of a tree where the various splits are not realized only on a single feature, but over a linear combination of them, improving the performance of the overall process but increasing at the same time the associated computational burden.

However, due to the large load and high complexity in the training procedure such kind of classifiers presents, Dollar et al. decided to avoid the application of oblique decision trees, introducing in their new detector a local decorrelation of the feature channels extracted from each proposal (which gave the opportunity to apply the well-known orthogonal decision trees in a much powerful way).

Basically, after the computation of the feature pyramid associated to an acquired image, a $m \times m$ correlation matrix Σ (representing the correlation of all the features situated in a generic $m \times m$ patch p) is computed.
From the Σ matrix, a Singular Value Decomposition (SVD) is applied, obtaining the matrices Q and Λ:

$$\Sigma = Q\Lambda Q^T \tag{2.16}$$

and a final decorrelation achieved, simply multiplying each single patch p to the extracted Q matrix:

$$\bar{p} = Q^T p \tag{2.17}$$

where $\bar{p}$ is the decorrelated patch p.

Unfortunately this kind of approach, despite its correctness and validity, produces in the end an overcomplete representation of the initial image, since for each feature channel $m \times m$ sub-features are extracted by the presented decorrelation.

In order to reduce the computational burden introduced by this operation, two optimizations are so proposed by Dollar:

- First, the authors suggests to apply an approximation of Q, taking only the first k eigenvectors associated to the highest eigenvalues.
  This allows to cut the original set of $m \times m \times 10$ features to a more contained set of $k \times 10$.

To further reduce the dimension of the final representation and improve the robustness of the overall process, a final downsampling over the features extracted is also recomended at this step.

- Second, instead of decorrelating the extracted patches with a matrix multiplication (i.e. $Q^T p$), an application of k convolutional filters over the feature channels[6] is advised in [4].
  This gives in fact the possibility to reduce the overall computational load simply recurring to highly optimized convolutional functions, which can be applied (if necessary) in the Fourier Domain.

Experimental evidence finally showed how good values for an application of this decomposition are $k = 4$ and $m = 5$, which produce sufficiently good local representation of the image, without incurring in excessive computational burdens.

Thanks to the innovations introduced, the LDCF detector outperforms many other state of the art solutions, achieving an accurate and robust multi-scale single-class object detection with sufficiently contained computational burdens.

2.6 AlexNet

Up to now a lof of different approaches have been presented, which concern the application of a suitable classifier over a set of feature channels that are defined and fixed a priori (as a result of multiple attempts and decades of human research).

In the last years however, thanks to the innovations introduced by new powerful parallel architectures, massive black box approaches came back in fashion, giving the possibility to apply such particular kind of solutions even in situations previously not affordable.
In the Computer Vision field this return to the past has been materialized into the so-called Deep Neural Networks (aka DNNs) and especially into their subclass named: Deep Convolutional Neural Networks (aka CNNs).

CNNs are basically very large and deep neural networks (i.e. made by a lot of layers) where each neuron is applied over set of features coming from the previous layer in a convolutional way, replicating typical image processing operations.

[6] Which is equivalent to the previous multiplication since the multiplication of Q^T per p is basically the convolution of each column vector of Q per p

The main idea in this case is to try to learn the best possible features (which are strictly related to the case that must be faced) and then apply a simple linear classifier over the final feature set. The power of this kind of models is so concentrated in the overall structure responsible of extracting good representations of the given input image.

Despite the multitude of models presented that could be applied in a lot of different situations and with different goals (e.g. CIFAR10, MNIST, GoogleNet, NIN ...), one of the solutions that has shown really good performance, both in generic image classification and object detection tasks, is the so-called AlexNet [6].

AlexNet is a 7-layers convolutional neural network, which takes in input a $224 \times 224 \times 3$ (then extended in a later version to $227 \times 227 \times 3$) proposal and produces as output 4096 features representing the content of the given region (see figure 2.8).

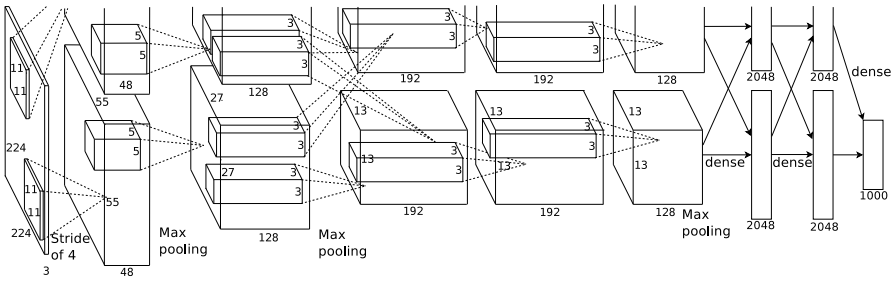

FIGURE 2.8: An illustration that represents the AlexNet structure.

In order to analyze its particular architecture however, two main components, which are the basic blocks of this kind of network, need to be previously explained. Namely: the ReLU Layer and the Pooling Layer.

Typically, in every Neural Network (NN) each node applies a linear transformation over a set of input data and then a non-linear activation function, in order to decide if activating the neuron or not.
Good examples of activation functions are in this case: binary activation functions, linear activation functions, hyperbolic tangent or sigmoidal activation functions.

It turns out however, that saturating non-linearities (non-linearities which are asymptotically limited by a fixed value) are much slower in terms of training times than the non-saturating ones.
A good example of non-saturating non-linearity is illustrated in eq. 2.18 (introduced by Nair and Hinton in [8]), which allows to achieve same training results wrt other saturating activation functions, but with much less iterations over the training set (fig. 2.9).

Any neuron that implements this activation function is typically referred to as: Rectified Linear Unit (ReLU).

This particular function plays a fundamental role in our revision, since it corresponds to the only activation function, which has been exploited in AlexNet for costructing the overall model.

$$f(x) = max(0, x) \tag{2.18}$$

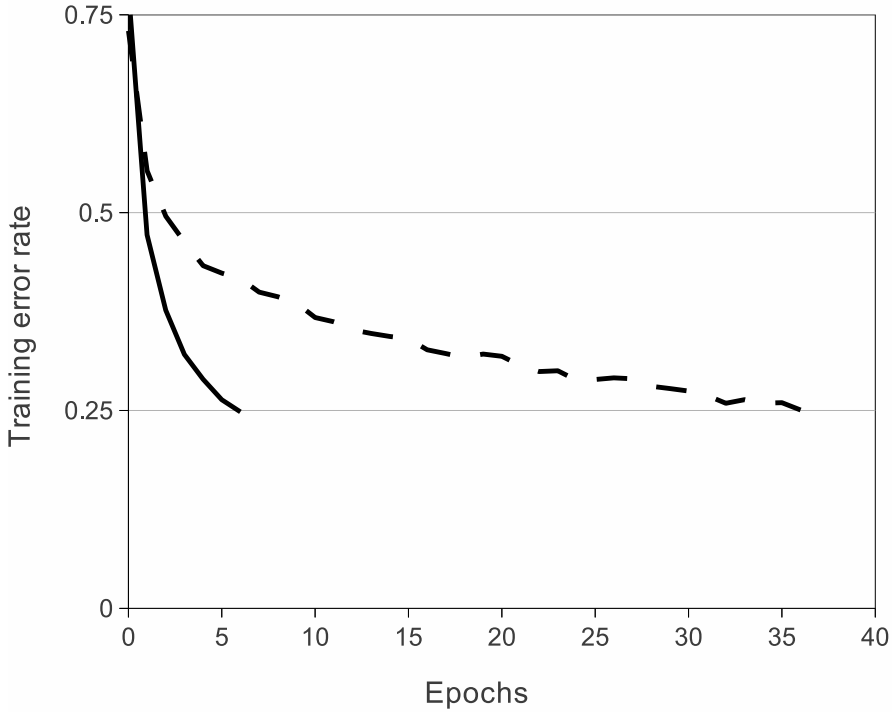

FIGURE 2.9: Comparison of training errors at various epochs over the training set, produced applying the Cifar Net with ReLU (the solid line) and tanh layer(the dashed one). The ReLU implementations reaches a 25% training error rate six times faster than its equivalent with tanh neurons.

As previously explained however, the architecture that is here considered concerns the application of a multitude of convolutional neurons over the feature space generated by the previous layer. In this case due to the large size of these architectures, if neurons are free to learn independently one by each other the relative kernels they should implement, the overall NN may overfit in the end the training data, reducing the performance produced at test time.
In order to improve robustness and generalization of the proposed solution, the various neurons available in this large and depth NN are typically split in different sets (named kernel maps), which implement different functions one to each other.

Thanks to this grouping in fact, the number of degrees of freedom presented by the net can be strictly reduced and so the possibility to overfit the data.

Unfortunately, even applying the illustrated solutions, the overall network may present a poor generalization if not trained with really large amounts of data, so, in order to increase robustness and performance of the trained model, a pooling layer aimed at reducing the amount of data that flows from one layer to another is typically introduced.

The main idea is to generalize the local behavior of an image, extracting every s points the maximum response produced by all the neurons situated in a $z \times z$ neighborhood. This allows to reduce the amount of features retrieved at each layer, since only the strongest behaviors are actually retrieved step by step.
If $s = z$ a traditional local pooling is obtained, if instead $s < z$ an overlapping pooling is produced.
This last case is the one implemented by Alex et al. in their architecture, where a 3×3 overlapping pooling is applied with stride s equal to 2 ($z = 3$ and $s = 2$).

Now that all the required basic blocks have been introduced, the main structure of AlexNet can finally be illustrated.
7 main layers in the following order build this architecture:

- A convolutional layer made by 96 kernels of dimensions $11 \times 11 \times 3$, applied with a stride of 4 pixels over the initial input image and followed by a suitable ReLU layer plus a max-pool layer with $z = 3$ and $s = 2$.
- A convolutional layer made by 256 kernels of dimensions $5 \times 5 \times 48$, applied without any stride over the initial input image and followed by a suitable ReLU layer plus a max-pool layer with $z = 3$ and $s = 2$.
- A convolutional layer made by 384 kernels of dimensions $3 \times 3 \times 256$ and followed by its ReLU layer, applied without any stride or max-pooling layer.
- A convolutional layer made by 384 kernels of dimensions $3 \times 3 \times 192$ and followed by its ReLU layer, applied without any stride or max-pooling layer.
- A convolutional layer made by 256 kernels of dimensions $3 \times 3 \times 192$ and followed by its ReLU layer, applied without any stride or max-pooling layer.
- Two fully connected layer (producing each one 4096 features in output) directly connected one to each other, without any other layer in the middle.

Each layer is aimed at extracting a higher level representation of the input image, which could be used to properly classify any patch taken into account.
At the top of the net a SOFTMAX classifier is typically applied, in order to retrieve the strength of the various predicted classes associated to the current analyzed proposal.

Thanks to the flexibility this particular network presents, AlexNet has become one of the most famous CNN available in the literature, spurring frameworks, such as the Berkeley Caffe framework (`http://caffe.berkeleyvision.org/`), to propose their ILSVRC trained version[7] as the reference net for many classification and detection tasks.

It turns out in fact that an already trained version of AlexNet can be easily finetuned (i.e. trained) for recognizing not only objects contained in the original training set, but also targets it has never seen, simply recurring to a set of stochastic gradient iterations applied over the new training data.

The main idea is to exploit the detection capabilities already learnt by the network (which could be significant, given the multitude of objects already processed) and adapt them to a complete different situation, guiding in some way the training process towards good configuration points, which otherwise could be hardly achievable with any training from scratch.

One of the main contributions of this work is a study aimed at finding a reasonable and valid training process for finetuning this kind of networks to pedestrian detection tasks, highlighting suggestions and best practices that can be followed, in order to produce satisfactory results.

2.7 R-CNN

One of the latest solutions available in the literature that concerns the object detection with CNN is the so-called: Regions with Convolutional Neural Network Features (aka R-CNN [9]). One of the main strengths of this detector wrt the previous ones is the capability to recognize with accuracy objects belonging to different classes, giving the possibility to extract much more information from a single image, without the necessity to apply multiple detectors.

The main idea of R-CNN is to extract all the possible regions (which may contain interesting objects) independently of their scale, stretch them in order to fit a fixed

[7] Over which the original model has been trained by Alex et al.

dimension (a process usually referred to as warping) and finally apply a complex CNN, in order to retrieve the classes the various objects belong to (fig. 2.10).
3 main algorithms have so been selected for facing the 3 main tasks involved, namely:

- Selective Search for the extraction of promising regions (an algorithm capable at retrieving hierarchies of regions, applying continuously regional aggregations guided by local similarity measures, see 2.7.1.)
- AlexNet for feature extraction
- Linear SVM for object classification

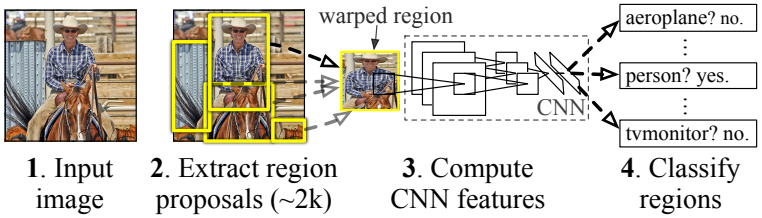

FIGURE 2.10: RCNN pipeline

Using this kind of approach and finetuning the initial AlexNet model trained on the ILSVRC-2013 dataset with the PASCAL VOC dataset, Girshick et al. achieved in their work state of the art results, outperforming all the competitors that participated at the ILSVRC-2013 (fig. 2.11).

Following these results, the R-CNN detector has been analyzed in this work and compared with other state of the art object detectors, highlighting weaknesses and limits of the presented approach, which make this kind of solution hardly applicable in particular situations as the one here considered.

2.7.1 Selective Search

Selective Search [10] is a hierarchical segmentation algorithm aimed at extracting all the possible regions situated at all the available scales in a given image.
The main idea behind Selective Search is to apply an initial segmentation of the image recurring to a graph-based algorithm and then aggregate, in multiple iterations, the various regions that are the most similar among the extracted ones.

The algorithms proceed exactly in this way:

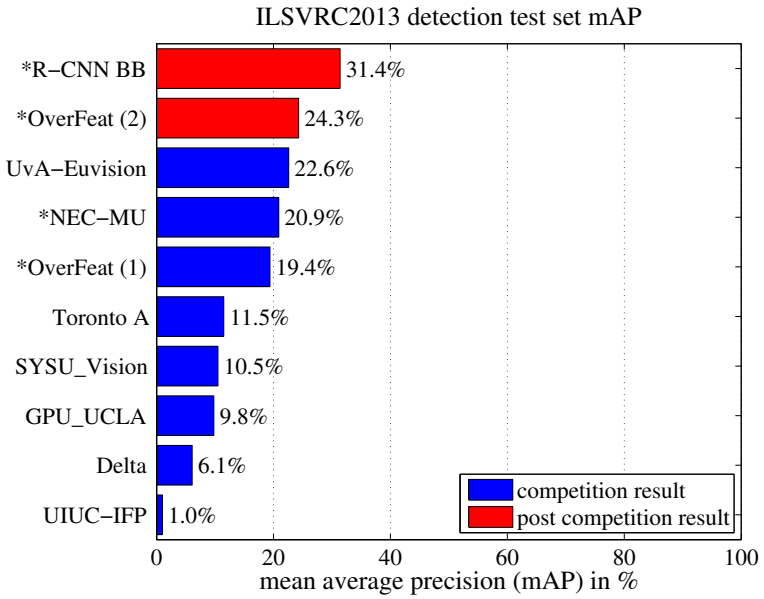

FIGURE 2.11: Results obtained by R-CNN over the ILSVRC-2013 datasets. The results of R-CNN and Overfeat are highlighted in red since they have been realized and tested only after the competition has taken place. Algorithms marked with an * use data outside the ILSVRC-2013 competition.

1. First, an Efficient Graph-Based Image Segmentation [11] is applied over the input image, in order to produce a good set of initial regions:

   (a) An undirected graph $G = (V, E)$ is created [8].

   (b) An initial segmentation S^0 is fixed equal to V, according to the result produced at the previous step.

   (c) All the edges contained in E are sorted in a non-decreasing order wrt their weights.

   (d) Iterating over the arcs of the ordered set E (for $q = 1 \dots |E|$) and denoting with:

   - v_i and v_j the vertices connected by the q-th edge (i.e. $o_q = (v_i, v_j)$).
   - $w(o_q)$ the weight of the edge o_q.
   - C_i^{q-1} the component of S^{q-1} containing v_i.
   - C_j^{q-1} the component of S^{q-1} containing v_j.
   - $MInt(C_1, C_2)$ the minimum internal difference of $C_1 \cup C_2$ (eq. 2.19).

[8] Where the set of vertices V represents all the available pixels and the set of edge E represents the dissimilarities situated between each couple of neighbors (e.g. the difference in intensity or color)

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{2.19}$$

$$Int(C) = \max_{e \in MST(C,E)} w(e) \tag{2.20}$$

$$\tau(C) = \frac{k}{|C|} \tag{2.21}$$

where $MST(C, E)$ is the Minimum Spanning Tree that connects all the element of C.

the segmentation S^q is constructed from the segmentation S^{q-1} considering that:

- If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leqslant MInt(C_i^{q-1}, C_j^{q-1})$, S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}.
- Otherwise $S^q = S^{q-1}$.

The parameter k is of fundamental importance in the presented approach, since it tunes the grain of the aggregation process.
Small values of k produce in fact a lot of regions, while large values of k bring the algorithm to aggregate a lot of different components.

2. After the segmentation is realized and the initial regions $R = \{r_1, \ldots, r_n\}$ are obtained, the Hierarchical Grouping responsible of creating the desired hierarchy of regions is finally applied (Algorithm 1).

As it is possible to see, the algorithm:

- requires the computation of a similarity among each region and all the others available in the same set.
- assignes a priority to each region, representing the likelihood to actually have something of interest in that particular location.

Priorities are established in a reverse order wrt the step in which the two regions that compose the final location are aggregated[9].
The main idea is that, if two regions are aggregated in the last iterations of the algorithm, they probably represent distinct objects with really different behaviors. Therefore, they should be processed first wrt the other proposals.

[9]Plus a certain randomness, in order to avoid pushing only large regions wrt smaller but more promising ones.

Algorithm 1: Hierarchical Grouping Algorithm

Initialize the similarity set $S = \emptyset$;
Initialize the priority set $P = \emptyset$;
foreach *Neighboring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$;
    $S = S \bigcup s(r_i, r_j)$
i = 1;
while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = max(S)$;
    Merge corresponding regions $r_t = r_i \bigcup r_j$;
    Remove similarities regarding $r_i : S = S\ s(r_i, r_*)$;
    Remove similarities regarding $r_j : S = S\ s(r_j, r_*)$;
    Calculate similarity set S_t between r_t and its neighbours;
    $S = S \bigcup S_t$;
    $R = R \bigcup r_t$;
    $P = P \bigcup i * rnd()$;
    $i = i + 1$;
Extract object location boxes L from all regions R;
Sort the extracted boxes L in descending order wrt P;

Similarities instead are computed recurring to the sum of 4 different normalized measures:

$$s(r_i, r_j) = s_{color}(r_i, r_j) + s_{texture}(r_i, r_j) + s_{size}(r_i, r_j) + s_fill(r_i, r_j) \tag{2.22}$$

which represent complementary affinities that sould be considered, in order to produce a general and meaningful result.
The 4 measures that are employed in this case are:

- $s_{color}(r_i, r_j)$, which measures color similarity.
  For each region, a color histogram of 25 bins is computed for each channel and a global histogram is retrieved vectorizing and normalizing[10] the various computed bins.
  The similarity is measured recurring to an histogram intersection:

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k) \tag{2.23}$$

where c_i^k is the k-th bin of the color histogram of image i and the same states also for c_j^k.

[10]The normalization process is accomplished recurring to the L_1-norm of the entire histogram.

- $s_{texture}(r_i, r_j)$, which measures texture similarity.
  For each region, 8 gaussian derivatives are computed in 8 different orientations for each color channels and an histogram made by 10 bins per orientation is finally computed. As in the previous case, a global histogram is obtained vectorizing and normalizing[10] the retrieved bins.
  The similarity measure is computed recurring again to an histogram intersection:

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k) \tag{2.24}$$

where t_i^k is the k-th bin of the texture histogram of image i and the same states also for t_j^k.

- $s_{size}(r_i, r_j)$, which encourages regions to have similar size throughout the grouping.
  This is a desirable property since avoids to create few large regions that glob all the others, ensuring to have object locations at all scales in every part of the input image.
  It is computed as:

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(I)} \tag{2.25}$$

where $size(I)$ is the size of the overall image and $size(r_i)$, $size(r_j)$ the corresponding ones of regions r_i and r_j.

- $s_{fill}(r_i, r_j)$, which measures how well region r_i and r_j fit into each other.
  It retrieves the fraction of image not belonging to neither the two components, but contained in the tight bounding box that globs both. The main idea is to fill gaps produced during the grouping.
  If region r_i is contained in region r_j in fact, it is reasonable to merge them. On the other hand, if the two regions are really far one from each other, the two components do not represent good candidates and so, they should not be merged together.

The similarity measure is computed as:

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{i,j}) + size(r_i) + size(r_j)}{size(I)} \tag{2.26}$$

where $size(BB_{i,j})$ is the size of the tight bounding box that contains both region i and region j. $size(r_i)$, $size(r_j)$, $size(I)$ are instead defined as in $s_{size}(r_i, r_j)$.

Thanks to the multiple considered conditions and the good segmentation initially produced by the graph-based algorithm, Selective Search showed really high performance over the PASCAL VOC 2007 test set, outperforming all the other competitors selected by the authors (figure 2.12).
The algorithm also achieved a recall equal to 0.99 if around the top-10,000 retrieved locations are considered for classification, appearing in this way as a suitable candidate whenever multi-class object detections need to be realized.

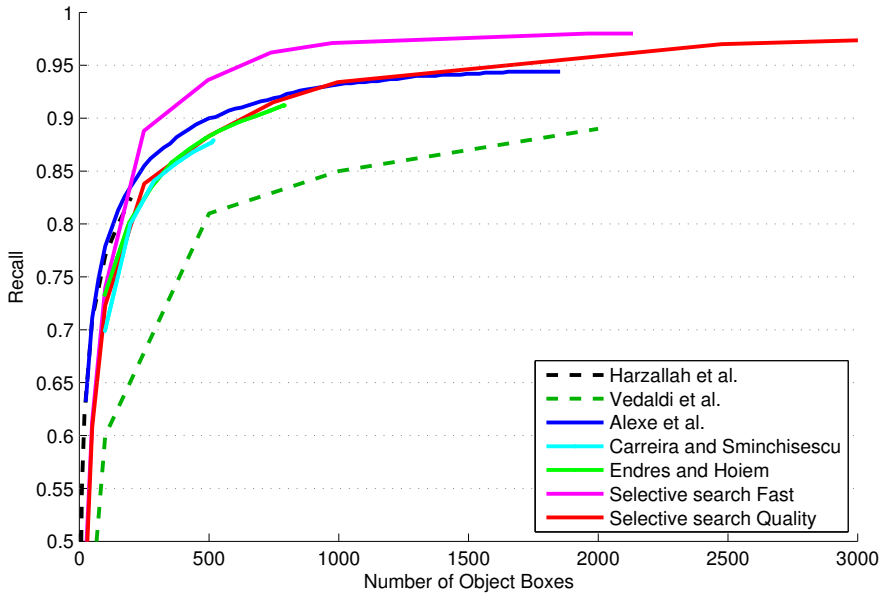

FIGURE 2.12: Trade-off between the number of objects retrieved and the Pascal Recall criterion (i.e. the recall is the percentage of objects in the given image for which there is a proposed region with an overlap larger than 50% of the object size)

2.8 ACF+AlexNet

As presented in the previous section, the pipeline proposed by Girshick et al. in [9] represents nowadays a good baseline for object detection tasks, since gives the opportunity to apply CNNs in a really flexible and powerful way.
Inspired by these results, Hosang et al. further explored the presented approach, introducing a new solution capable at outperforming the performance of the best previous convnet pedestrian detector by more than 10% over the Caltech Pedestrian Dataset, representing at publication time one of the strongest solutions ever evaluated.

The structure of the architecture is basically the same of the R-CNN proposal and is composed by 3 main blocks:

1. A region proposal method.
2. A convolutional neural network.
3. A linear classifier.

The main innovation introduced is given in this case by the change applied in the region proposal algorithm.
Rather than using the Selective Search method, the authors suggest in their publication to use a simple and sufficiently accurate detector, which produces a reduced set of regions that are then analyzed by a more complex and powerful classifier as a CNN+SVM.
The idea is to use a baseline detector with a high true positive rate and let the CNN to decide which region is of interest and which not.

The proposal architecture is built in this case by:

1. An ACF detector.
2. A finetuned AlexNet.
3. A suitable Linear SVM.

The strongest net trained by the authors with the method they present achieves a Log Average Miss Rate equal to 23.3% on the Caltech test set, outperforming in this way the performance produced by all the state of the art detectors here illustrated (fig. 2.13).

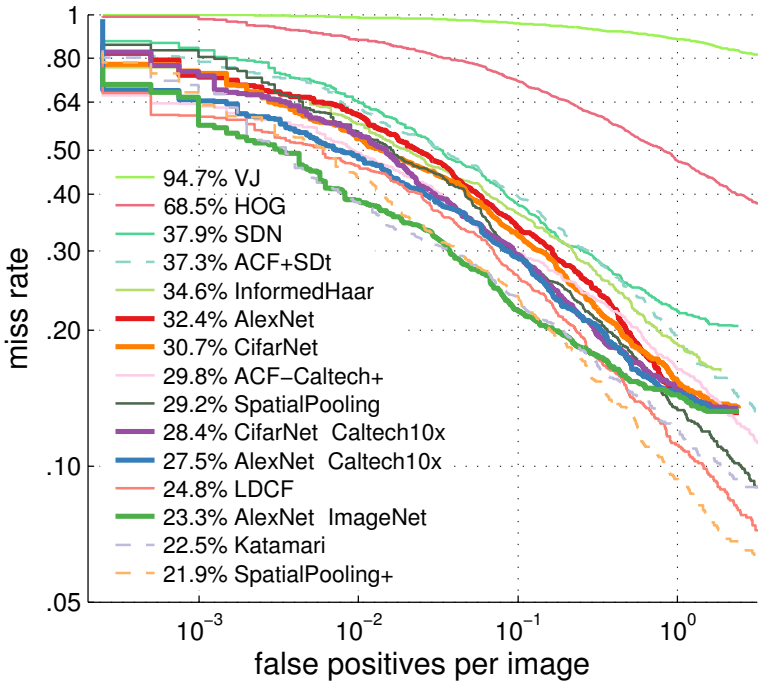

FIGURE 2.13: Results obtained applying various detector over the Caltech test set

In order to achieve these results, however, a suitable finetuning of AlexNet must be applied.

In their publication Hosang et al. provided some general guidelines, which may be useful for achieving state of the art performance.

First of all, they highlighted the necessity to have a huge amount of data to train the selected CNN.
Assuming to use in fact a very huge and deep network, a large dataset helps to prevent overfitting at training time, improving the quality of the final predictions retrieved.

An extension of the Caltech training set has so been introduced in [5], namely: the Caltech10x dataset.
The main idea in this case was to augment the amount of data available, simply taking 1 frame over 3 instead of 1 over 30 (which is the typical choice made in the construction of the Caltech dataset, fig. 2.14).

A second main contribution introduced by Hosang et al. concerns instead how actually building the training set that should be used to finetune the selected NN.
The finetuning of a large architecture is in fact a really difficult process, due to the high probability to produce models which overfit the training samples.

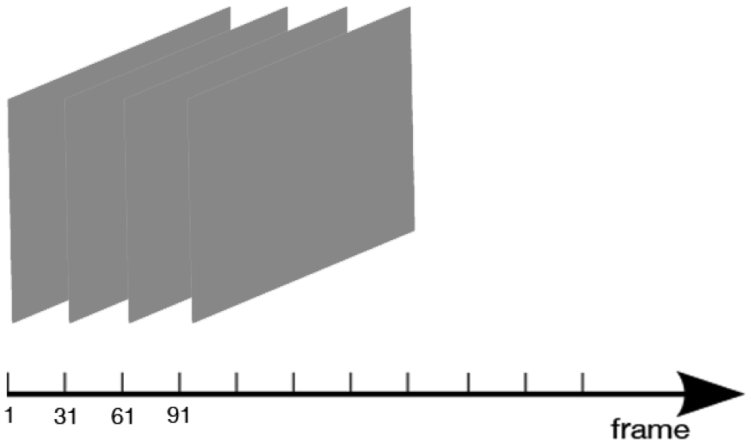

FIGURE 2.14: Typical extraction applied over the Caltech dataset, where only one frame per second (1 every 30, since the video is recorded at 30 FPS) is taken.

In order to adapt the already trained AlexNet over this new situation, a specific study aimed at finding a suitable procedure to apply for producing the best possible results has been realized by the authors.

After various training attempts, Hosang et al. found out that the following operations significantly improve the performance of the finetuned model over the Caltech Dataset:

1. Train originally the net over the ImageNet dataset and only then proceed with the finetuning operation.
2. Use positive regions that are produced by the ground truth (GT) annotations and not by the proposal detector (i.e. we extract from each image the patches that contain perfectly the pedestrians we want to detect, instead of the best regions produced by the region proposal algorithm).
3. Use a fixed positive to negative ratio in the training samples equal to 1:5 (i.e. for every positive sample, 5 negative samples are taken).

The results obtained following these steps are shown in tables 2.1, 2.2 and 2.3. [11]

[11] Note that the results obtained in tables 2.1 and 2.2 have been extracted training the lighter CifarNet network over the Caltech training set and have been then extended to AlexNet. This allowed to test multiple configurations in a reduced amount of time, still producing good performance in the end.

Positives	Negatives	MR
GT	Random	83.1%
GT	IoU<0.5	37.1%
GT	IoU<0.3	37.2%
GT, IoU>0.5	IoU<0.5	42.1%
GT, IoU>0.5	IoU<0.3	41.3%
GT,IoU>0.75	IoU<0.5	39.9%

TABLE 2.1: Effects of different positive-negative selections over the final log-average miss rate (MR). Best performance appears for positive samples taken from the ground truth annotations (GT) and negative samples with an Intersection over Union (IoU) smaller than 0.5 wrt any available pedestrian.

Ratio	MR
None	41.4%
1:10	40.6%
1:5	39.9%
1:1	39.8%

TABLE 2.2: Effects of different positive:negative ratios over the final log-average miss rate (MR).

AlexNet initial training	Finetuning	SVM training	Test MR
Random	none	Caltech1x	86.7%
ImageNet	none	Caltech1x	39.8%
P+Imagenet	Caltech1x	Caltech1x	30.1%
P:Places			27.0%
ImageNet			25.9%
ImageNet	Positives10x	Positives10x	23.8%
	Caltech10x	Caltech10x	23.3%
Caltech1x	-	Caltech1x	32.4%
	-	Caltech10x	32.2%
Caltech10x	-	Caltech1x	27.4%
	-	Caltech10x	27.5%

TABLE 2.3: Results obtained executing multiple training and computing the log-average miss rate of the final model over the Caltech test set (set05-set10). Best performances are obtained pre-training AlexNet with the ImageNet dataset and finally finetuning the proposal over the Caltech10x dataset. Training the SVM over the Caltech10x instead of Caltech1x improves the performance, even not significantly ($\sim 0.2\%$).

However, as in any learning procedure, a set of parameters need to be fixed in order to obtain satisfactory results.

The values imposed in this case for the training process are the following ones and

corresponds to the same applied by Girshick et al. in [9] for training their best model for the ILSVRC-2013 [12]:

- test_iter: 100
- test_interval: 1000
- base_lr: 0.001
- gamma: 0.1
- stepsize: 20000
- display: 20
- max_iter: 100000
- momentum: 0.9
- weight_decay: 0.0005
- snapshot: 10000

Anyway, as the CNN requires a suitable training procedure, also the SVM calls for a specific analysis aimed at establishing the best parameter setup that should be applied.

In order to achieve the best possible performance, Hosang et al. proposed in their paper to apply a simple grid search aimed at checking the results produced by multiple combinations of the SVM regularization factor C and the maximum IoU (Intersection over Union, see eq. 2.27) associated to a negative sample.

$$IoU_{i,j} = \frac{Overlap(i,j)}{area_i + area_j} \tag{2.27}$$

Intersection over union among BB i and j.

The C regularization factor controls in fact the trade-off between maximizing the margin of the separating hyperplane and minimizing the amount of slack required to accept all the training samples in non-separable cases (which is a typical situation), determining for this the quality of the final retrieved hyperplane.
The maximum IoU controls instead the trade-off between which extracted subregions

[12] For a specific explaination of the meanings, refer to section 3.2.

should be considered as negatives and which as positives, affecting the content of the final training dataset and so the performance of the overall architecture.

Following this procedure the authors showed that good parameters for training the SVM are:

- $C = 10^{-3}$
- $IoU = 0.5$

which, by the way, correspond to the parameters we will use in our experiments.

Chapter 3

Finetuning of a Deep Neural Network and Region Proposal Analysis

3.1 Introduction

As reviewed in Chapter 1, in order to exploit at best the power of any kind of classifiers one wants to implement, multiple solutions based on a common pipeline of 3 main steps (see figure 3.1) have been introduced in the literature.

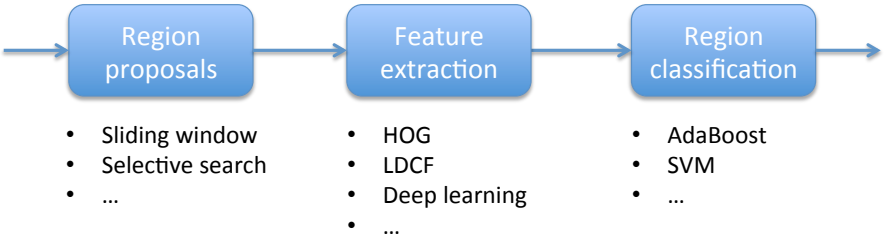

FIGURE 3.1: A common pipeline for pedestrians detection.

In particular, Hosang et al. presented in [5] a valid analysis that explains how applying ACF as region proposal algorithm and then classifying the retrieved regions with a finetuned AlexNet it is possible to obtain state of the art results for pedestrian detection, outperforming the performance presented by many other proposals.
Multiple suggestions on how achieve a satisfactory finetuning have also been illustrated.

Despite the quality of this work, the authors unfortunately decided to not publicly release neither the code capable at achieving the performance illustrated, nor the finetuned model.
Therefore, in order to realize one of the most advanced DNN detector available at the state of the art, a suitable study of the architecture presented by Hosang in [5] will be here illustrated, highlighting which solutions could be suitable as proposal algorithms over the Caltech Pedestrian Dataset and introducing a set of further suggestions on how producing a valid finetuning of the original AlexNet.

To carry on a satisfying analysis, this chapter will be divided in several sections, each one concerning every single possible problem that must be faced or tool that can be exploited for realizing the selected architecture:

1. First, an introduction of the main framework employed for finetuning the initial model and the reference toolbox exploited for retrieving the final results, will be presented.

2. Then, the main proposal algorithms that can be used for realizing the architecture illustrated in [5] will be discussed, further highlighting which solutions could be suitable for embedded prototypes.

3. Finally, a deeper revision of the guidelines defined by Hosang will be illustrated, introducing a set of new tips and suggestions, which could be followed in order to produce better results.

3.2 The CAFFE framework

One of the most popular deep learning framework available over the web for realizing training and finetuning of DNN is represented nowadays by the so-called Berkley Caffe.

Caffe is a deep learning framework realized with speed and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
It operates in a Linux or OS X console and requires a bit of familiarity with typical shell commands to be mastered.
Special wrappers are also defined in order to allow an easy development of applications in MATLAB and Python based on this underlying framework.

The main idea behind Caffe is to give a simple and textual layer-oriented description of the network that one would like to train. Each layer is defined with one of the multiple types the framework provides and connected in a cascade fashion with the other layers only with a simple named reference.

As an example, the definition of AlexNet used in this work is reported in the Appendix: 3.14 of this chapter.

Every layer has a name, receives as input a specific named object coming from the previous one, produces as output a specific named object for the next one and is annotated with parameters that define:

- Its structure.
- The initialization process
- The training process.

For initializing the net, Caffe requires in fact a set of parameters for each layer that are grouped in a section named "weight_filler", which is of fundamental importance whenever a particular layer is not found in the initialization model given in input to the framework.
Indeed, at initialization time each layer of the network is initialized with the corresponding weights contained in the input .caffemodel file (a file that contains the pre-trained model) and associated to a layer with the same structure and name. If a particular layer is not found, the missing weights are randomly initialized exploiting the parameter setup contained in the "weight filler" section (which contains a set of mandatory parameters required for setting a related p.d.f.).

As for the initialization process, also for training Caffe needs the definition of a set of parameters that guides the learning process. All the information required for completing this task are in this case collected in the so-called "param" section.
Before analyzing this section however, an introduction to the learning process the framework realizes is strictly necessary to understand its content.

In order to train a network in Caffe, a fundamental step is the definition of a solver in an apposite file (as reference we report here in the appendix 3.15 an example of the solver used to train our models).

The solver orchestrates model optimization by coordinating the network's forward inference and backward gradients to form parameter updates, which attempt to improve the loss.
The responsibilities of learning are so divided between the solver itself (for overseeing the optimization and generating parameter updates) and the Net (for yielding loss and gradients).

Three are the main types of solvers that can be used in Caffe in order to realize the training of a network:

- Stochastic Gradient Descent (SGD).
- Adaptive Gradient (ADAGRAD).
- Nesterov's Accelerated Gradient (NESTEROV).

In the following, any kind of training that will be explained should be considered executed with SGD, which is typically the most common choice.

At each iteration of the learning method Caffe computes the outputs of a set of samples (called batch), compute the gradients in a stochastic way and determine the updating factor for each layer wrt:

- Its gradients.
- A general learning rate defined in the solver.
- A specific learning rate defined in the net for that specific layer.

In particular, the global loss function $L(W)$ defined for the overall set of weights W (eq. 3.1) is approximated during the training by its corresponding stochastic function (eq. 3.2) over which the final gradients are computed.

Differently from a typical stochastic gradient descent approach, the approximation of the global loss is not given by the loss realized by a single sample but by a set of samples, which improves stability and convergence.
In order to reduce the overfitting of the network, a weight-decay approach based on a specific regularization parameter λ is applied. The main idea is to avoid overfitting, increasing the loss if the values of the weights explode (which is exactly a typical overfitting situation).

$$L(W) = \frac{1}{|D|} \sum_{i}^{|D|} f_W(X^i) + \lambda r(W) \tag{3.1}$$

Where:

- D is the set of sample
- W is the set of net weights
- X^i is the $i-th$ sample of the set D
- $f_W(\cdot)$ is a loss function computed over a given input sample
- $r(W)$ is a measure that summarizes the weights (such as an Euclidean Norm of the vector of weights W)

$$L(W) \approx \frac{1}{N} \sum_{i}^{N} f_W(X^{(i)}) + \lambda r(W) \tag{3.2}$$

Stochastic approximation of the loss function introduced in Caffe over a subset of N samples.

To further increase the performance, a second regularization factor is typically introduced in the learning process, the so-called momentum μ.
The momentum is basically an interpolation factor that allows to consider the value of the previous updating factor in the computation of the current one, as in eq. 3.3.

$$V_{t+1} = \mu V_t - \alpha \Delta L(W_t) \tag{3.3}$$

$$W_{t+1} = W_t + V_{t+1}$$

The learning rate α is given by the product of the base learning rate (defined in the solver) and the learning rate multiplier (defined in the net definition). The same states also for the values of the weight decays factor (i.e. each weight has its own λ, which is computed as the product between a base weight decay and the one exposed in the net definition).

The param section, previously introduced, groups in a single piece of code the multipliers needed to apply this flexible and layer dependent approach.

Finally, since in some cases a continuous decreasing of the learning rate improves the stability of the final result (increasing the converge to a local minima), a dedicated factor γ is defined in Caffe, which is multiplied every s iterations (with s equal to the stepsize)

to the base learning. This reduction helps to avoid jumping effects and consolidate the final result.

Good values for momentum and weight decay for AlexNet are typically: 0.9 for the first and 0.0005 for the seconds; which corresponds the ones suggested by the authors of R-CNN to finetune their models and so the ones applied during the following tests.

3.2.1 Quality Measures

Up to now we have presented how the framework operates for instructing the selected NN, however, no indications have been defined about how the validation of the extracted models is actually handled at training time.
As for any other training in fact, the final goal of the presented learning procedure is to return a sufficiently robust model capable at obtaining good performance even over unseen conditions.
Two different measures are returned by Caffe for achieving satisfactory results in this case:

1. The Accuracy of the DNN, which is a measure that defines the fraction of correctly classified samples. It is defined as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

where:

- TP (true positives) is the number of correctly classified positive samples.
- TN (true negatives) is the number of correctly classified negative samples.
- FP (false positives) is the number of misclassified negative samples.
- FN (false negatives) is the number of misclassified positive samples.

2. The Loss of the DNN, which corresponds to the negative logarithmic average of the probabilities to have correctly classified an analyzed region (eq. 3.5).

Both the presented indications are obtained exploiting a suitable SOFTMAX classifier (which is introduced by Caffe at the end of the net during the training), in order to allow an easy instruction of the selected NN with a bench of SGD iterations.

$$E = \frac{-1}{N} \sum_{n=1}^{N} log(\hat{p}_{n,l_n}) \tag{3.5}$$

$$\hat{p}_{n,k} = \frac{e^{w_k \cdot x_n}}{\sum_{k'} e^{w_{k'} \cdot x_n}}$$

where E is the error function we want to minimize, $\hat{p}_{n,l_n}$ is the predicted probability to assign sample n to its right class l_n, $\hat{p}_{n,k}$ is the probability to assign sample n to class k, w'_k is the vector of coefficient associated to a generic class k' and x_n is the feature vector extracted from sample n.

3.3 Piotr Toolbox

In order to allow an easy and powerful processing of the obtained results in MATLAB, in the following, a comparison of the detections retrieved from the tested detectors, with the set of annotations produced by the Caltech researchers, will be required.
To avoid a complete rewriting of all the code needed to read, convert and compare the annotations with the extracted BBs, a suitable toolbox capable at solving all these operations has been exploited in this work, the so-called Piotr Toolbox.

The Piotr Toolbox is meant to integrate the MATLAB image-processing toolbox, facilitating the manipulation of images and videos with easy and efficient scripts.
In order to reduce the computational load, a set of MEX files is distributed in this toolbox together with a set of suitable wrappers, which can be used to allow an easy integration of the implemented functionalities with the prepared MATLAB scripts.

The reference to the Piotr Toolbox is of fundamental importance in the present work, for all the guidelines contained in the implemented functions, which allows to achieve comparable results with the ones presented in the literature.

First of all, a typical problem that needs to be solved in detection tasks concerns in fact, how to realize the best possible matching between the realized detections (aka DTs) and the annotated ground truths (aka GTs).
In order to obtain satisfying results for this problem, the Piotr Toolbox suggests to use a modified version of the Pascal Criterion.

The Pascal Criterion states that a ground truth and a detection match if their Intersection Over Union (IoU) is over a given threshold (typically 0.5), eq. 3.6. Which makes the overall process of distinguishing true positives (matched detections) and false positives (unmatched detections) really easy and straightforward.

$$match = \begin{cases} 1 & \text{if } IoU > threshold \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

However, the presented criterion doesn't consider how to process regions that we marked as to ignore, because containing occluded pedestrians or made by groups of people. In this situation, a suitable adjustment must then be taken into account.

The modification that Dollar et al. suggest in their toolbox basically consists into a suitable filtering of the realized matches. Namely, any possible match that occurs with any GT marked as "to ignore" is suitably discarded (i.e. it is not considered neither as false positive nor as false negative).
This modification allows to discard all the strange situations that we highlighted in Chapter 1, giving a powerful tool to check the performance of any proposal algorithm.

Besides the criterion introduced for efficiently match the retrieved BBs however, another fundamental aspect that should be considered during the matching concerns the GTs normalization process.

Observing in fact the content of the ground truths annotated in the Caltech Dataset, it is possible to observe how a lot of annotated BBs are really small (i.e. with an height smaller then 50px) or have different aspect ratios (i.e. width/height).
So, in order to efficiently compare the BBs produced by any detector that is only able to scan the images using a fixed-size sliding window, a normalization of the underlying GTs appears strictly necessary for accomplishing a satisfying match of the retrieved detections.
If this aspect is neglected, a lot of unmatched detections, caused by the different sizes of the retrieved bounding boxes and the ones extracted from the annotations, may in fact be produced, compromising the quality of the final measure retrieved.

Dollar et al. faced this problem and showed in [12] how the average GT aspect ratio basically corresponds to 0.41.
Consequently, fixing the sliding window with this aspect ratio and normalizing the ones of the selected ground truths with the functions provided in the toolbox, an increasing in the DT-GT match probability can be easily realized.

As we have just mentioned however, the Caltech Pedestrian Dataset is full of little GTs, representing faraway pedestrians.
Therefore, in order to produce the most valuable and comparable results with the ones

presented in the literature, a suitable configuration is typically applied during the generation of the GTs annotations, the so-called: reasonable setting ([12]).
In this particular setting, only well-visible pedestrians taller than 50 px are considered for matching, while all the others are marked as "to ignore".
Asking in fact to a detector to properly recognize regions with a dimension smaller then 50×20 pixels, which can also be occluded, appears as useless and unreasonable for the most of the detection tasks.
For this reason, the "reasonable settings" of Dollar et al. appears in the literature as a typical configuration applied for comparing different solutions trained over the Caltech Pedestrian Dataset and so, also the one applied in the present work.

3.4 Choice of the Region Proposal Algorithm

After the introduction of the main tools exploited in our study, the analysis we realized for producing our solution can finally be presented.
The first applied step concerned in this case the choice of a suitable region proposal algorithm capable at retrieving a reduced amount of regions, which contain all the possible pedestrians available over the selected Caltech Dataset.
As illustrated in section 2.7, in order to avoid an application of AlexNet in sliding window (which will be very expensive over each single image), an algorithm capable at extracting a set of regions that really likely contains all the possible interesting objects is strictly necessary.

The proposer that should be selected however must satisfy some desirable properties such as:

- Reduced complexity. It must be fast.
- Precision. It must extract with accuracy position and dimension of the various bounding boxes that could contain interesting objects, in order to give to the NN the possibility to recognize in a very robust way every possible target.
- Recall. It must return a set of proposals which really likely contains all the available targets situated in the analyzed image.

In order to compare the performance of different proposers and select the best available option, a suitable measure that allows to summarize at least the two last mentioned

properties must so be selected. The measure we selected in this case is the so-called Average Best Overlap (aka ABO).

The ABO (defined in eq. 3.7) expresses the mean optimal overlap (defined as IoU) between each ground truth and the best region extracted, representing a consolidated measure in the Computer Vision field for validating the performance produced by such kind of algorithms (utilized for example by Uijlings et al. in [10] to prove the goodness of the Selective Search method, figure 3.2).

$$ABO = \frac{1}{G^c} \sum_{g_i^c \in G^c} \max_{l_j \in L} Overlap(g_i^c, l_j) \tag{3.7}$$

where G^c is the set of ground truths of class c, $g_i^c \in G^c$ is the current analyzed ground truth, L is the set of bounding boxes proposed by the algorithm and $l_j \in L$ a single region.

Differently from the ROC and the LAMR introduced in 1.1, the ABO is particularly suited for comparing proposal algorithms, since gives not only an idea of the capability to detect a single pedestrian in an image, but also a measure of how good the best retrieved regions are on average (i.e. to what extent they contain all and only the target pedestrians).
Following this idea, 4 main different solutions have been compared in this study that seem to be the most promising in view of what we have seen in Chapter 2.
The selected algorithms are:

1. ACF
2. LDCF
3. HOG+SVM
4. Selective Search

Taking as reference the set06 of the Caltech Pedestrian Dataset, in order to prove in multiple operative conditions the power of all the selected solutions, 3 main comparisons aimed at highlighting the different capabilities to propose good regions have been realized.

Basically, fixed a minimum height of the GTs that should be considered (e.g. 20, 50, 90 px), the following approach has been applied for every algorithm:

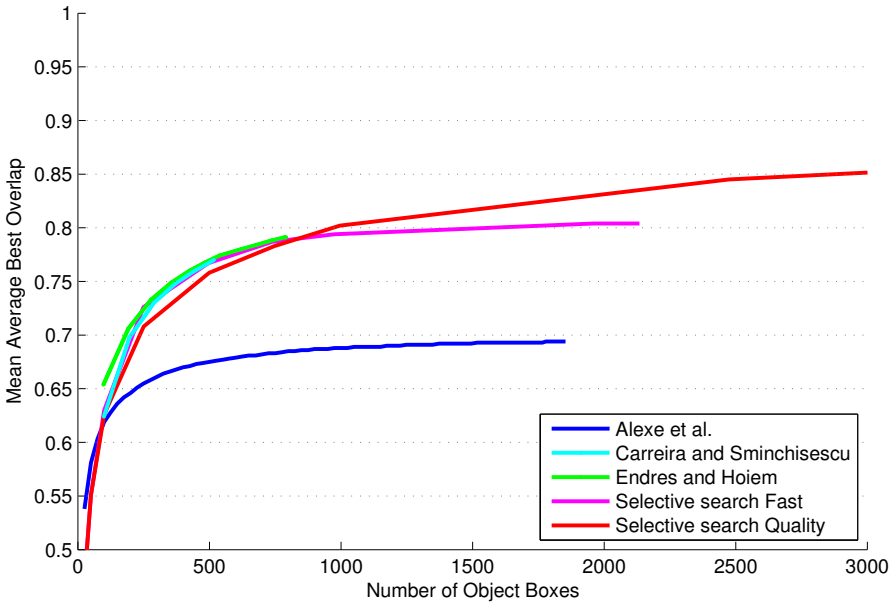

FIGURE 3.2: Trade-off between number of objects retrieved and the Mean Average Best Overlap produced by various algorithms (i.e. the mean ABO computed over multiple classes of objects), reported by Uijlings in [10].

1. A collection of regions retrieved from the set06 has been initially realized.

2. The extracted proposals have been order by score of relevance (or estimated priority for Selective Search), in order to have the most promising boxes at the top of the collection and the less promising at the bottom.

3. For every image of the test set06, the detections have been decomposed in groups which include: the first proposal of each image, the first two proposals of each image and so on.

4. For each group of proposals the ABO has been computed, giving the possibility to draw a plot (ABO - number of considered proposals) that compares the capabilities of the different algorithms to retrieve a small number of good regions, containing all the pedestrians available in an input image.

The obtained results are illustrated in figure 3.3.

As it is possible to see, ACF appears as the best possible proposal algorithm that we can apply if we consider pedestrian taller then 50 px, slightly outperforming LDCF. HOG+SVM and Selective Search appear instead in 3rd and 4th position, showing especially this latter poor performance over the Caltech Pedestrian Dataset.
This result mainly comes from the fact that Selective Search aggregates a set of initial

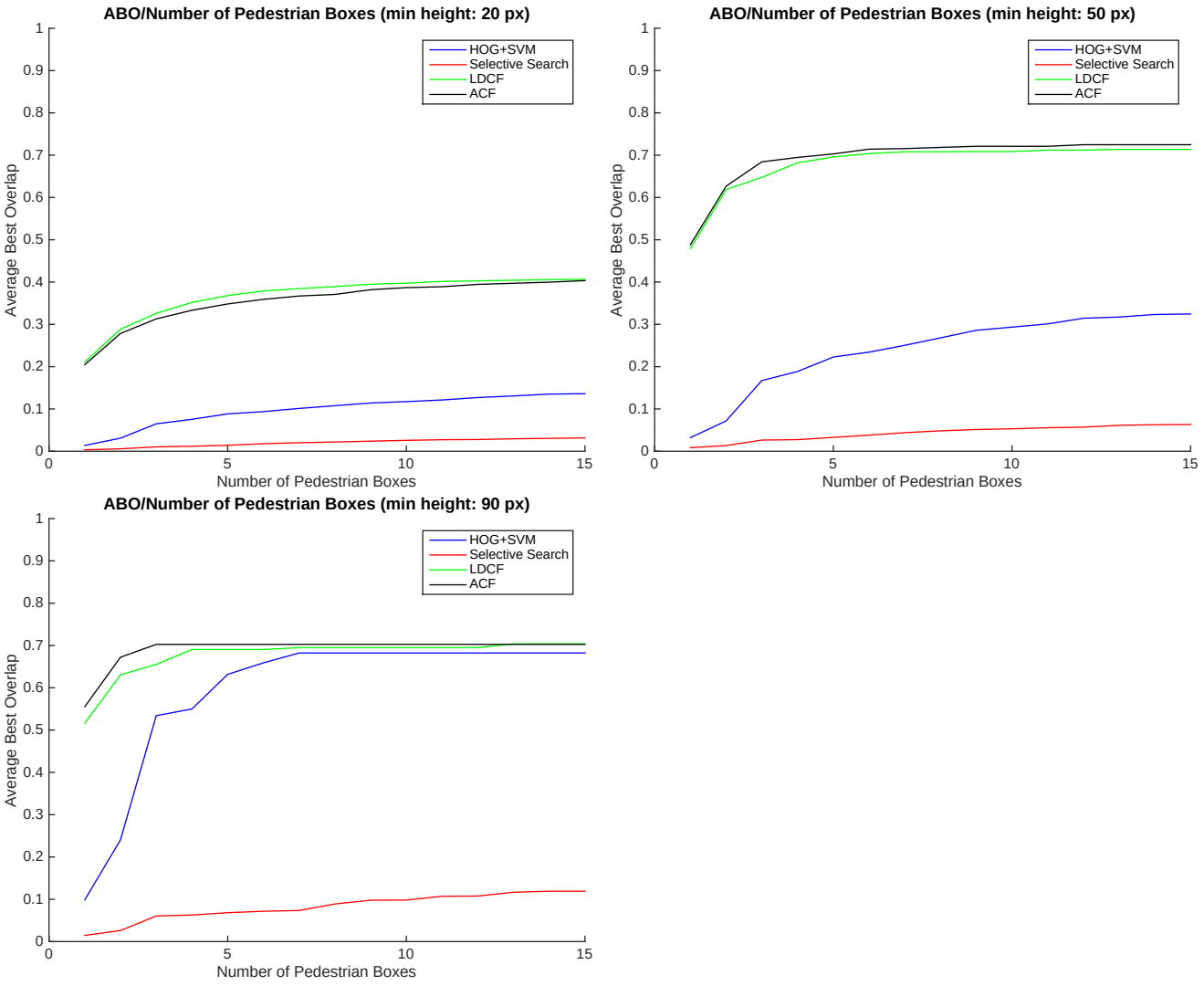

FIGURE 3.3: Comparison of ACF,LDCF,HOG+SVM and Selective Search over the Caltech Dataset set06 with various minimum pedestrians' height.

regions extracted from the images, recurring to similarity measures that are computed step by step. If these regions are initially too big or pedestrians are not so evident in the image, Selective Search fails aggregating regions that instead should be disjointed, producing in this way really poor sets of proposals.

Following these results, it appears clear that R-CNN with Selective Search simply represents an unfeasible solution over this dataset, for the inaccuracy of the proposal algorithm as well as the excessive computational cost it presents.
Indeed, this particular solution runs in the order of seconds over an Intel i7, showing a computational cost absolutely too expensive for any real time situation.

Due to the results here exposed, the detector that will be here illustrated has been constructed considering only ACF or LDCF for promising regions extraction.
In particular, due to the similarity these two solutions presents, a further analysis of the performance produced by both these algorithms has been realized in this work (section 3.11), showing how in the end LDCF, despite the slightly lower proposal capabilities we have here illustrated, turns out as a better choice wrt ACF in the final pipeline, thanks to the higher discriminative properties it presents.

3.5 Embedded Prototype

ACF and LDCF are publicly released with Matlab + MEX code, which makes this kind of detectors available but hardly portable to an architecture different from an x86/x64. For this reason, if an embedded prototype of the following work should be realized and the conditions are a bit less stringent then the ones presented in the Caltech Dataset, a solution based on HOG+SVM could be taken into account. This kind of detectors, in fact, appears available in a lot of different open source libraries such as the famous OpenCVs, which could be easily applied over multiple platforms thanks to the high portability they present.

In order to check the availability of the presented solution even on this kind of architectures, an embedded prototype based on HOG+SVM and a finetuned AlexNet over the Pascal VOC dataset has been realized in this work.

The architecture selected in this case is the Nvidia Jetson TK1, which is the first mobile processor to have the same advanced features of a modern desktop GPU, while still using the low power draw of a mobile chip.
It mounts a Tegra TK1 SoC (i.e. CPU+GPU on the same chip) with a quad-core 2.3GHz ARM Cortex-A15 CPU and a Tegra K1 GPU with 192 CUDA cores. It shows a computational power up to 326 GFLOPS and CUDA capability equals to 3.2.
Thanks to this configuration, an implementation of the CAFFE framework, which uses optimized OpenCV libraries, can be installed and a C++ prototype can be easily realized recurring to a set of CUDA parallel functions already available.
The dataset selected for our test in this case is the Performance Evaluation of Tracking and Surveillance 2009 (aka PETS 2009), which displays better environmental conditions compared to the Caltech Dataset, with well visible pedestrians and a resolution equal to 768×576.

Applying the presented solution over this simpler dataset, the results obtained have been quite impressive.
A frame rate equal to 2.5 fps has been reached, ensuring restricted computational times and good quality detections.

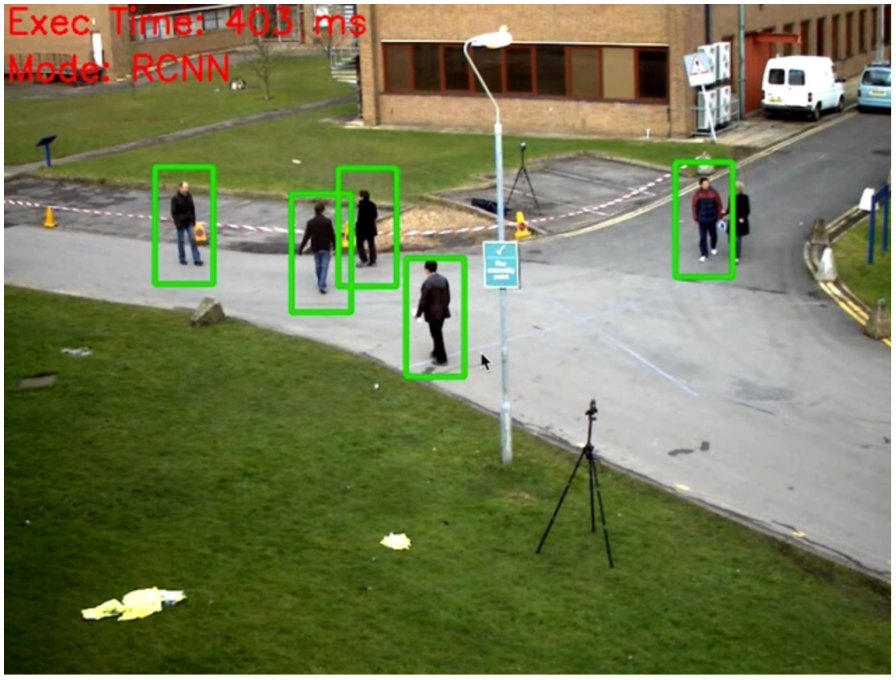

FIGURE 3.4: Screenshot of the realized embedded prototype which runs over the Jetson TK1 at 400 ms with HOG+SVM as preselector and AlexNet as CNN.

Since an analysis of the performance produced over this dataset is out the scope of this work, no related measures have been computed in this case.
However, in order to analyze possible future implementations of the final prototype, a computational analysis of the implemented solution has been carried on.

Two distinct DNNs (AlexNet and NIN) have been tested over the Jetson TK1 with our C++ prototype, collecting 50 times the amounts of time required to process different batches of images and averaging the data. The obtained results are expressed in figure 3.5.

As it possible to see, thanks to the reduced dimension of the Network In Network architecture, which is 10 times lighter than AlexNet, NIN outperforms its competitor in the profiling just presented, requiring less than an half of the time needed by AlexNet to complete the processing.
However, thanks to its higher complexity, AlexNet may produce better performance in

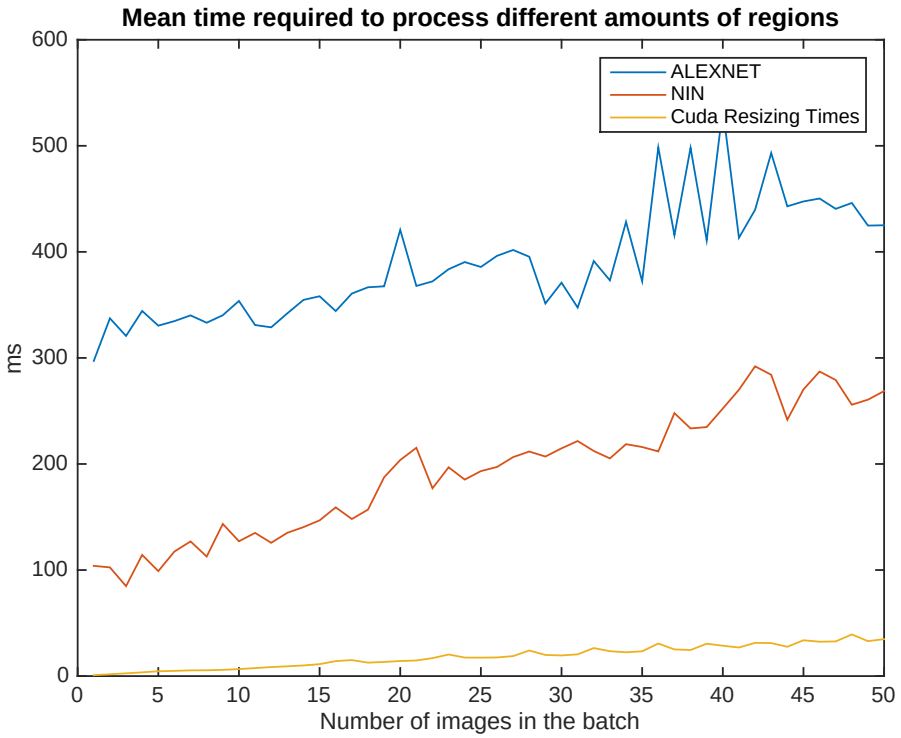

FIGURE 3.5: Mean time required to process batches of different size (i.e. containing a different amount of regions) over the Jetson TK1.

the end both in term of accuracy and robustness. A careful analysis aimed at establishing the model to use over this dataset should therefore be further considered.

3.6 Finetuning According to Hosang et al.

After the selection of a valid proposal algorithm, the next block that needs to be realized in our architecture is a suitable classifier capable at differentiating in the most robust way, all the retrieved positive samples from the negative ones.
Following the work presented by Hosang et al. in [5], we decided to focus our attention over the already presented AlexNet model.

Independently by the selected network however, if the original model is not already trained for detecting the exact class of objects one may be interested into, a specific finetuning operation needs to be applied for achieving satisfactory results.

Reviewing what already presented in Chapter 2 and giving a stronger theoretical background extracted from the experience gained during this work, what the authors suggest in [5] is to:

1. Finetune an already trained model generated from a very large and various dataset instead of training from scratch.

   The main idea is to carry the detection capabilities acquired in a generic situation to a specific one, correcting as much as possible the initial weights with a set of stochastic gradient descent iterations.
   The number of iterations, the learning rate and any other regularization factor must be defined during this phase, in order to let the net converging to the best possible model.

2. Recur to a very large training dataset.

   Due to the large and deep structure of AlexNet it is easy to overfit the training set during the finetuning operation, compromising the final performance produced at testing time.
   Increasing the dimension and variability of the dataset helps in this case to avoid this situation.
   Positive and negative samples coming from a lot of different situations help, in fact, at generalizing to unseen conditions, producing in the end more robust and useful models.
   All the finetuning procedures that should be realized over the Caltech Dataset are so suggested to be accomplished with the following configuration:

   - Training set: set00, set01, set02, set03, set04, set05 of the Caltech10x Dataset

- Testing set: set06, set07, set08, set09, set10 of the Caltech Dataset (i.e. where we take one frame every 30).

3. Use positive samples that are extracted from the ground truths and not those produced by the proposal algorithm.

   Typically, whenever there is the need to train a generic black box model, it is common practice to provide the learning procedure with the typical data that the model will observe during its real utilization.
   This good practice however doesn't hold in this case, or better, it must be adapted.

   If we would like to train a model that must recognize pedestrians in fact, regions extracted by the proposal algorithm may produce training datasets distorted both in dimensions and positions.
   In order to avoid this situation and let the net learn what exactly a pedestrian is, well-visible targets extracted at all the possible scales from the GTs must then be used.

4. Use a fixed positive:negative ratio.

   Analyzing the context where the Caltech Dataset has been acquired (i.e. a moving car that goes around the Californian streets), it appears evident how, in this particular set of data, a great deficit among positive and negative samples is present. For a lot of time a car may in fact meet no pedestrians on its way.
   Therefore, in order to avoid the net converging to a solution that will discard all the possible regions, a fixed ratio among the number of positive and negative patches, which should be used at training time, must then be established.
   Hosang et al. suggested in [5] to use a positive:negative ratio equal to 1:5 (one positive sample for every 5 negative samples) to reduce the effect of this situation.

5. Use a linear SVM as final classifier.

   In order to produce the best possible model capable at recognizing in the right way all the regions proposed by ACF or LDCF, Hosang et al. suggested in their work to apply a linear SVM as a final classifier over the 4096 features extracted by AlexNet. This operation, however, cannot be realized directly in Caffe but requires a small "net surgery" over the final model extracted.
   As already presented in fact, the framework recurs only to SOFTMAX classifiers during the training of the network.
   Discarding the classifier realized during this procedure appears so as a mandatory

step, in order to directly retrieve the features extracted and apply the desired SVM.

Despite the quality of the indications here presented, the 5 illustrated points do not cover all the possible sub-problems that emerge whenever a complete finetuning of a DNN needs to be realized.
So, in order to give a better understanding on how these large and deep NNs can be finetuned, in the following sections a set of different tips that may help during all the learning procedure will be highlighted and explained.

3.7 Random Cropping

The first step that should be taken into account, in any finetuning procedure, concerns how to build a suitable training set capable at producing good performance in the final model.
As it is possible to see from the previous section, among all the indicated suggestions, no indications have been defined about the dimensions of the regions that should be used during the training phase.
AlexNet accepts in input patches of 227×227 pixels, however, generating larger regions, from which randomly extract different training samples at each learning iteration, represents a valid option often applied in the Computer Vision field (figure 3.6) for not overfitting the generated training dataset (which represents a likely situation in this case due to the dimensions of the selected model).

Before presenting how to produce such enlarged training samples however, another consideration needs to be done.
Enlarging the proposed BBs is not a common practice just at training time for avoiding overfitting, but represents a fundamental aspect even at testing time, in order to adjust potential errors in the position and dimension the proposer may produce.
ACF and LDCF are not perfect and they can retrieve detections that are a bit translated or resized wrt the correct BBs indicated in the ground truth.
Augmenting the dimensions of the BBs extracted by the proposal algorithm can then be possible to include excluded part of the detected pedestrians that have been discarded during the proposing phase, increasing the probability to achieve a satisfactory classification and good quality performance in the end (figure 3.7).

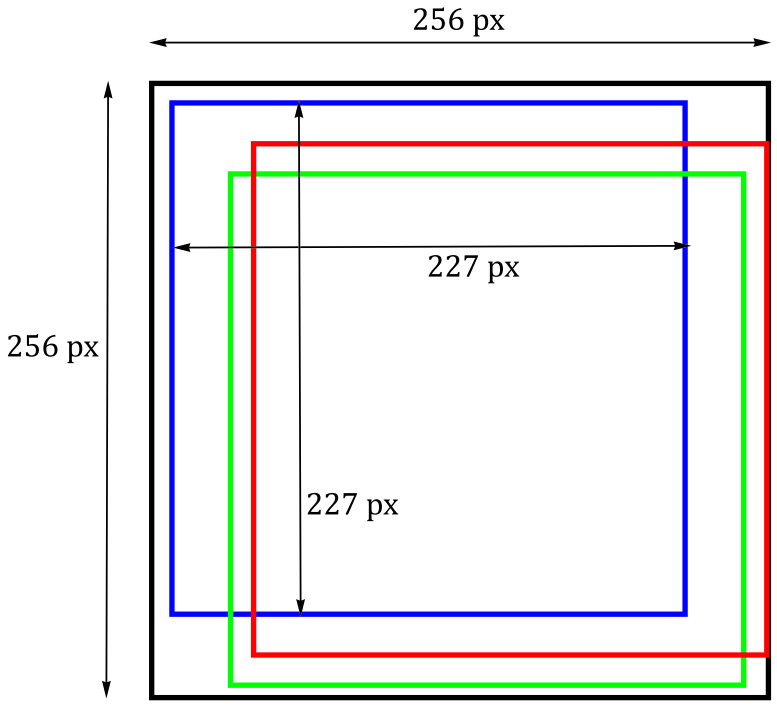

FIGURE 3.6: Example of a possible random-crop, where 3 regions of 227 × 227 are extracted from a larger region of 256 × 256

A procedure that unfortunately comes at the cost of an unnecessary background context in the regions retrieved.

Computing the average amount of padding to add becomes in this way of fundamental importance for achieving satisfactory results, since it influences not only the BBs retrieved at testing time, but also the ones that should be employed during the training for replicating similar operative conditions.

The following approach has so been applied:

1. First of all, we decided the dimensions of the final resized regions from which randomly extract the training samples. We chose to realize patches equal to 256 × 256 pixels, which is the typical size exploited by Caffe for training AlexNet.

2. Then, the average amount of padding per side, which is required by the proposed regions to entirely include the pedestrians has been computed.

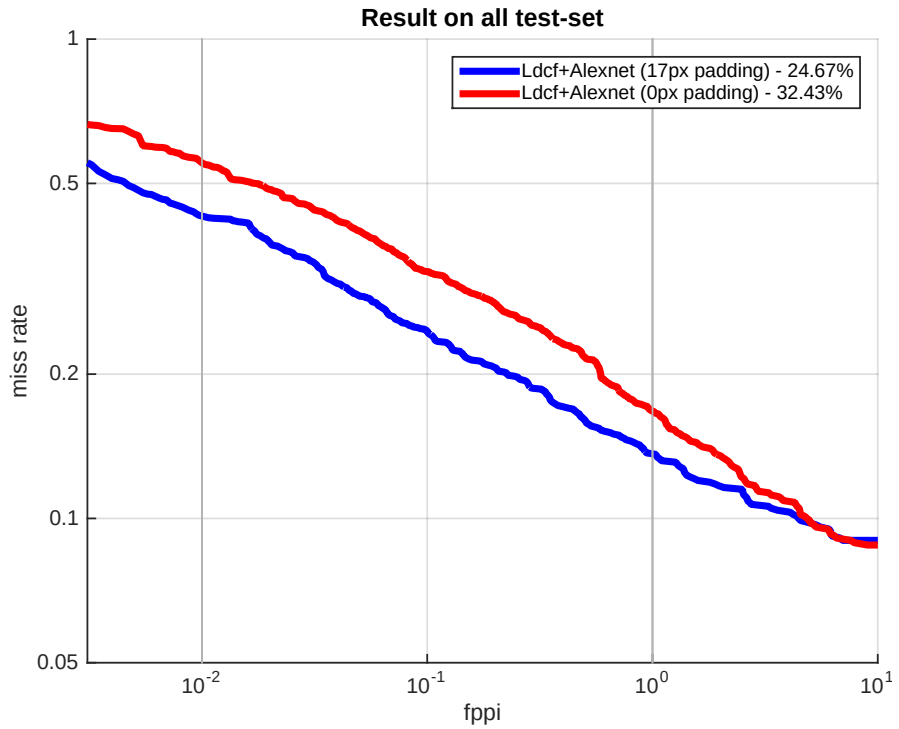

FIGURE 3.7: Effect of context padding.

Since the amount of padding to add is strictly related to the dimensions of the proposed regions (the padding required by a 50×20 patch is not the same required by a 200×81 patch, but typically smaller), a normalized measure has been extracted. We opted in this case to resize every patch retrieved by the proposal algorithm to a 227×227 region and compute the value of the padding in this final space.

A procedure that has been done simply considering that:

$$p_x^{ini} : \bar{w} = p_x^{fin} : 227 \tag{3.8}$$

$$p_y^{ini} : \bar{h} = p_y^{fin} : 227 \tag{3.9}$$

where p_x^{ini} is the amount of padding to add on each side along the x axis in the original region to contain all the detected pedestrian, $\bar{w}$ is the enlarged width of the retrieved proposal (i.e. the original width plus the amount of padding to add in the original space), p_x^{fin} is the amount of padding in the final space and the same states for the y axis (where $\bar{h}$ corresponds to the enlarged height).

and reversing the formulas.

The results we obtained in this way are illustrated in figure 3.8, where as it is

possible to see, the padding distribution[1] assumes a typical poissonian trend with mean equal to 17 px, which represents our desired amount of padding.

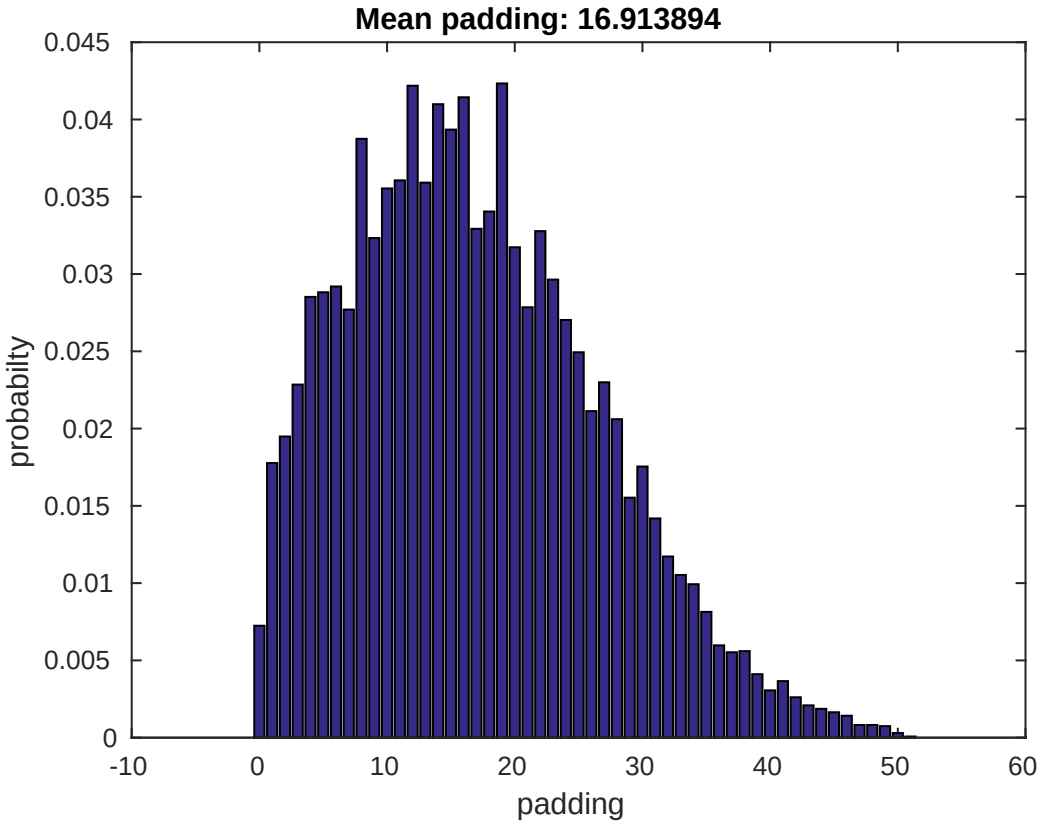

FIGURE 3.8: Probability distribution of the amount of padding to add in order to have a detection that contains entirely its best associated ground truth

$$p_{region} = max(p_{left}, p_{right}, p_{top}, p_{bottom}) \tag{3.10}$$

3. Computed the amount of padding that should be exploited at testing time, the last step that remained to be done concerned the determination of the amount of context δ, which should be included in the enlarged 256×256 regions around the associated GTs (fig. 3.10).

   The main idea was in this case to find a suitable value of this measure, such that, the mean amount of background introduced per side at training time corresponds to one produced on average per side at testing time (fig. 3.9).
   This allowed in fact to instruct NNs capable at dealing with typical operative conditions, where various amounts of context can be placed near the borders of

[1] Distribution obtained taking only one padding per region, that is, the maximum one which contains also all the other 3, eq. 3.10

the retrieved BBs, due to imperfections in the proposal algorithm or misleading padding operation (e.g. a padding operation applied on a perfect BB retrieved by ACF/LDCF).

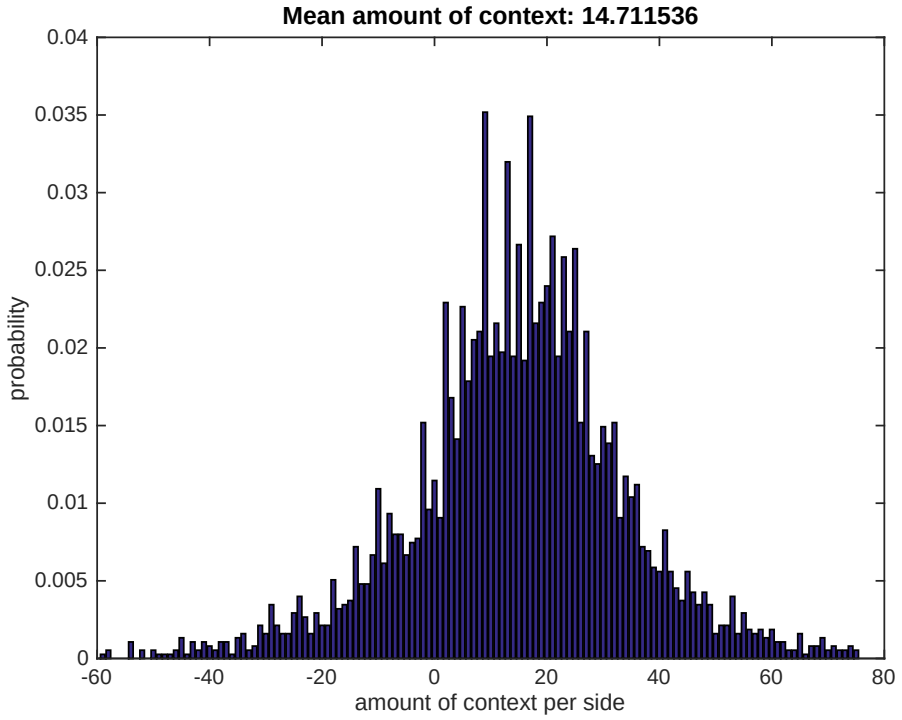

FIGURE 3.9: Distribution of the amount of context per side introduced in the resized 227×227 region after the padding operation at testing time. Negative values indicate borders inside the pedestrian region.

Checked therefore how the reference framework operates during the cropping phase, it turned out that Caffe exploits a uniform pdf for extracting the various training samples from the enlarged ones, consequently imposing an average amount of context per side p equal to the mean of uniform distribution (eq. 3.13), whose maximum and minimum directly depend on the value of δ (eq. 3.11, 3.12).

$$p_{max} = \delta \tag{3.11}$$

$$p_{min} = \delta - (256 - 227) \tag{3.12}$$

where p_{max} and p_{min} are respectively the maximum and minimum possible padding achievable, cropping randomly a 227×227 path in a 256×256.

Substituted in this way eq. 3.11, 3.12 in 3.13 and fixed $p = 15$ px (which corresponds to the average amount of background produced per side by the proposal

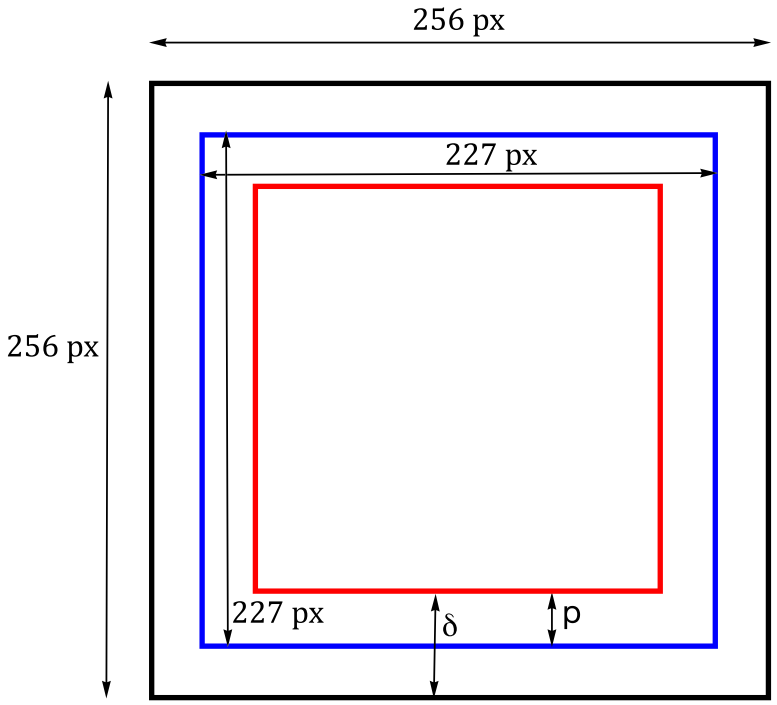

FIGURE 3.10: Illustration of our considered situation, in red it is indicated the perfect pedestrian detected by the proposer, in blue its relative padded version and in black the overall 256×256 region, where all the possible 227×227 training samples can be extracted.

$$p = \frac{p_{max} + p_{min}}{2} \tag{3.13}$$

where p_{max} and p_{min} are respectively the maximum and minimum possible padding achievable, randomly cropping a 227×227 patch in a 256×256.

algorithm), the only possible value of our desired parameter has easily been obtained:

$$p = \frac{p_{max} + p_{min}}{2} = \frac{\delta + \delta - 256 + 227}{2} = \delta + \frac{227 - 256}{2} = 15$$
$$\delta = 15 - \frac{227 - 256}{2} = 29.5$$

All the procedure here presented has been realized recurring to information extracted from BBs retrieved by LDCF and thanks to the same nature of the two proposal algorithms we selected in section 3.4, it has also been considered true for ACF.
In general however, this kind of analysis should be replicated for every different detector and for every different dataset that one may be interested into.

3.8 Sampling of the Negative Regions

In order to generate a suitable training dataset that allows to achieve satisfactory results, another fundamental step that should be taken into account concerns the selection of negative and positive samples aimed at identifying a sufficient vary bench of samples.

As already introduced, for the positive samples this is not a problem since all the unoccluded GTs need to be exploited for producing the largest possible set of data.
However, the same situation doesn't state also for negative regions due to the the large number of samples the detector retrieves in this case. A suitable selection of negative regions must then be applied, in order to respect the fixed positive:negative ratio we presented in section 3.6.

Before starting to introduce the approach we suggest in this case however, a further clarification deserves to be made.
The amount of data we need to handle in this case is extremely large, in the Caltech10x training set we are talking about a final dataset of 1.7 GB of JPG images, even more if we prefer to encode them as PNG in order to avoid any kind of compression lost.
So, whenever we need to find a suitable selection approach, we have to consider that efficient or greedy algorithms will turn out to be preferable than exhaustive ones, due to the reduced computational burdens and processing times they present.

In order to guarantee the most possible various dataset, suitable solutions aimed at maximizing global quality measures that determines the goodness of our final training set (e.g. the minimum distance among the Histograms of Colors of the selected samples, see figure 3.11) could be taken in account.
However, this kind of approaches requires time and does not necessarily produce good results in presence of regions with reduced dimensions (since it would be really hard to discriminate correctly different small regions).
Possible studies aimed at extending feature representations of the regions extracted with spatial and temporal information (i.e. locations of the regions extracted and source

frames) may then be applied to improve the final results, but they are out the scope of this work.

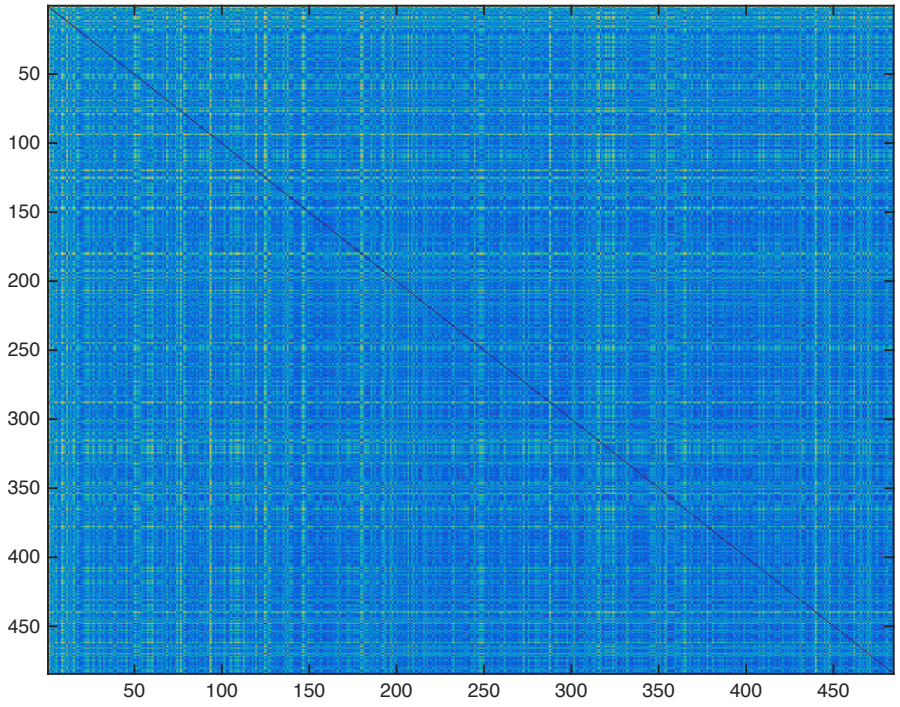

FIGURE 3.11: Distance matrix computed applying an L^2 norm over the Histograms of Colors extracted from the training dataset.

For the sake of simplicity and flexibility, a random selection of the negative regions has been applied in this work.

First of all, all the positive samples from all the training sets have been extracted and saved in suitable folders as PNG images, in order to reduce any possible compression loss.
Once all the positive data for each independent set has been collected and enumerated, the number of negative samples has been extracted and a draft of the negatives regions has been applied.
In order to diversify the final set of regions, the negative samples have not be drawn image by image, but a collection of all the extracted negative regions has initially realized and only then the draft has been applied.

Obviously, the presented solution is not perfect and doesn't guarantee that two identical regions are not present in the final set of regions extracted. However, if the set of negative samples is sufficiently large, the probability to have two identical regions is enough low that the presence of duplications can be easily neglected.

On the Caltech10x dataset, our random selection turns out to be sufficiently good to approach and outperform the previous state of the art.

3.9 Network Surgery and Multiple Initializations

Built the training set, the next step required to adapt the initial AlexNet model, towards the identification of only pedestrians, concerns the alteration of the structure of the net. Checking in fact the original AlexNet model, it's possible to observe as the final SOFTMAX classifier returns 1000 values instead of only 2 (i.e. pedestrian and no pedestrian), which corresponds to the 1000 probabilities of associating an analyzed sample to each of the 1000 classes available in the ILSVRC dataset[2].

In order to finetune the initial model, a substitution of the final fully connected layer responsible to compute the values required by the classifier needs so to be applied.
Basically, instead of reading the 4096×1000 weights contained in the model file, a random initialization of 4096×2 weights responsible to detect only the presence or absence of pedestrians has been realized in this work.

Unfortunately, the random initialization can radically change the initial starting point of the loss function, potentially leading to really different local minima after the completion of the finetuning operation (figure 3.12).

Multiple initializations are so an option that could be taken into account in order to face this situation and achieve the best possible performance from the finetuning of the modified AlexNet over the selected dataset.

[2]Corresponding to the set of images over which the original AlexNet has been initially trained by Alex et al.

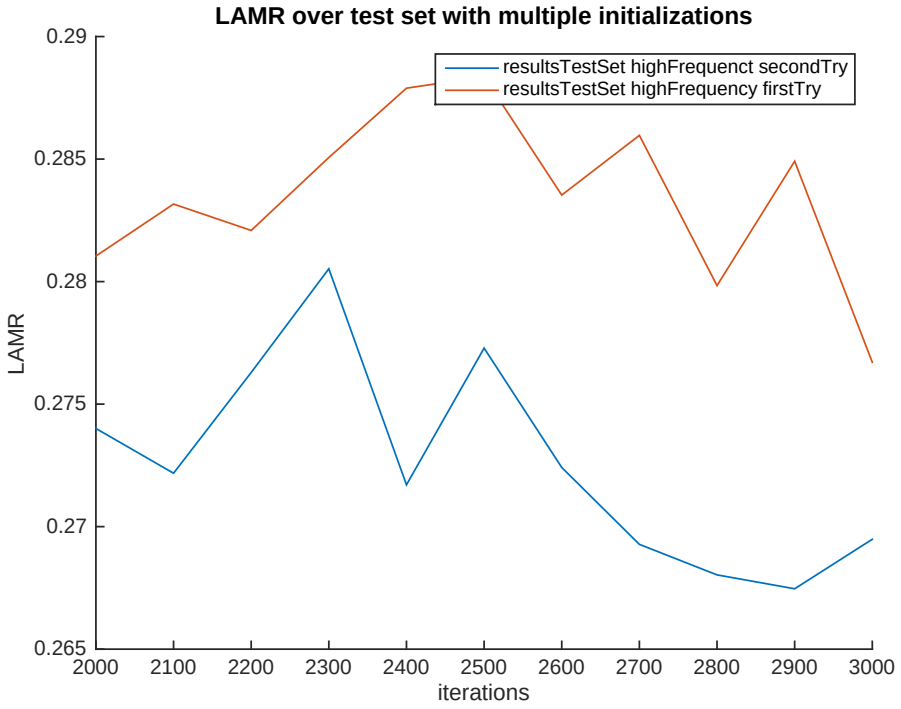

FIGURE 3.12: LAMR obtained executing two different trainings on the same training set but with two different random initializations.

3.10 Validation Procedure

Once we have presented how to build the underlying training dataset and the surgery required for adapting the net to the identification of pedestrians over the Caltech Dataset, the next step involved concerns the so-called validation procedure.
As in any other black box approach, whenever there is the need to learn a new model there is always the risk of overfitting the training data (unless the selected model has a really poor and simple structure). To avoid this, multiple validation approaches have been suggested in the years, such as:

- Hold-out cross validation, where the training set is split in two parts of different dimensions. One (the largest) used for training while the other (the smallest) for returning a quality measure (as our loss function), which guides the selection of the best model to avoid overfitting.
  It is probably the most common and simple validation process that can be applied, but it has the disadvantage of not using all the training data for learning the overall best model.

- K-folds cross validation. It represents an extension of the Hold-out approach where, in order to exploit the whole training set, instead of splitting the training data in two parts and use only one for training and only one for validating, what is suggested is to split the overall dataset in K parts (aka folds), train with K-1 of them and validate over the remaining. The procedure is then replicated selecting a different validation fold.
  Differently from the Hold-out procedure, at the end of the K-folds cross validation, K different validating measures are returned instead of just one.
  In order to give a clear idea of which model should be selected, an aggregation process must then be suitably applied to obtain a unique validating function from the K ones.
  A simple option in this case is the one that recurs to an arithmetic average of all the extracted quality measures, eq. 3.14

$$L = \frac{1}{N} \sum_{i=1}^{N} l_i \tag{3.14}$$

  where L is the average loss function and l_i is the validating function extracted from the i-th fold.

- Leave-p-out validation. It is the degenerate case of K-Fold Cross Validation, where K is chosen in order to produce folds of exactly p samples.
  Instead of splitting the training set in K group, in the Leave-p-out approach what is suggested is to take exactly p samples for validating and all the other for training.
  The overall final validation measure is obtained as in the K-folds case, averaging the different measures in a unique quality measure.

Analyzing these 3 approaches, it is quite evident that any application of the Leave-p-out approach will be misleading.
Considering in fact that for training we have 6 different sets (set00, ..., set05) of different dimensions (as illustrated in table 3.1) and recorded in different conditions, if only p-samples are taken at a time for validation and the others for training, similar regions coming from the same set of data will necessarily be considered for both the two phases, reducing the quality of the final retrieved measures.

Discarded the Leave-p-out approach, the remaining approaches that can be applied are the Holdout and K-folds cross validations.

Set	#Images
set00	8559
set01	3619
set02	7410
set03	7976
set04	7328
set05	7890

Set	#Images
set06	1155
set07	746
set08	657
set09	738
set10	728

TABLE 3.1: Number of images situated in each subset of the Caltech training and testing dataset. Different numbers of images with different contents result in different number of pedestrians for each set and so, in different amounts of training samples.

Thanks to the simplicity the Holdout approach presents, one may be tempted to train its own model using this kind of solution, where for example the first 5 sets are considered for training and only the last for validation.
However, this procedure, despite its higher quality wrt the Leave-one-out approach, would produce worse results then the K-folds method over our particular models.
It must be considered in fact that due to the depth and width of the NNs we are here considering, having the largest possible training dataset represents a mandatory feature to avoid overfitting.
For this reason, the best cross validation procedure, which can be applied over such complex and large models, is represented by the remaining K-folds cross validation.

Typically, a good rule of thumb consists into the splitting of the training set in folds of the same size to produce in every case training procedures with the same amount of data. However, this kind of approach is not applicable over the Caltech Dataset.
As illustrated for the Leave-p-out approach, in our training set 6 different sets of different size are available, containing videos recorded in different times and with different conditions. So, applying a regular division of the training set in K folds of the same size necessarily brings to split one set between validation and training data, which, as already explained, is a situation that must be avoided for obtaining good results.
In order to check the net every time on different pedestrians and in different operative conditions (different lights, different places etc.), a 6-fold cross validation where each fold corresponds exactly to one set of the training dataset (i.e. fold0 = set0, fold1=set1,...) has been applied in this work.

In particular, two main K-folds approaches have been compared to check which one could be better for selecting the best model to test. We wondered in fact if it would be better to average values obtained at multiple iterations (which is independent by the

size of the training set) or at multiple epochs (which instead depends by the training set size).

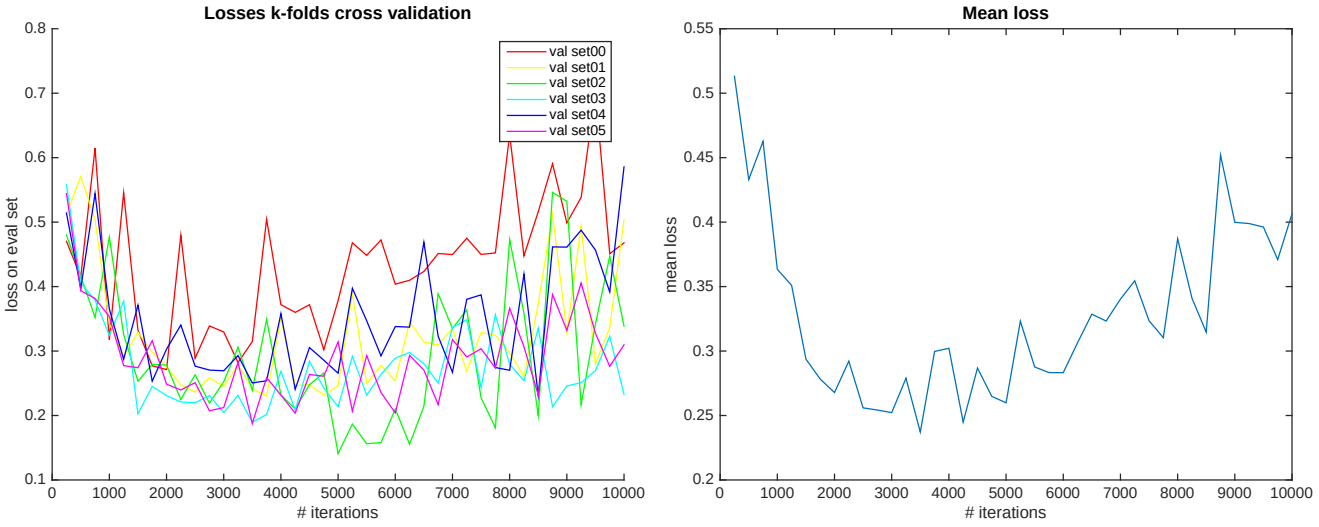

FIGURE 3.13: Losses obtained over the 6 possible validation sets and final average loss.

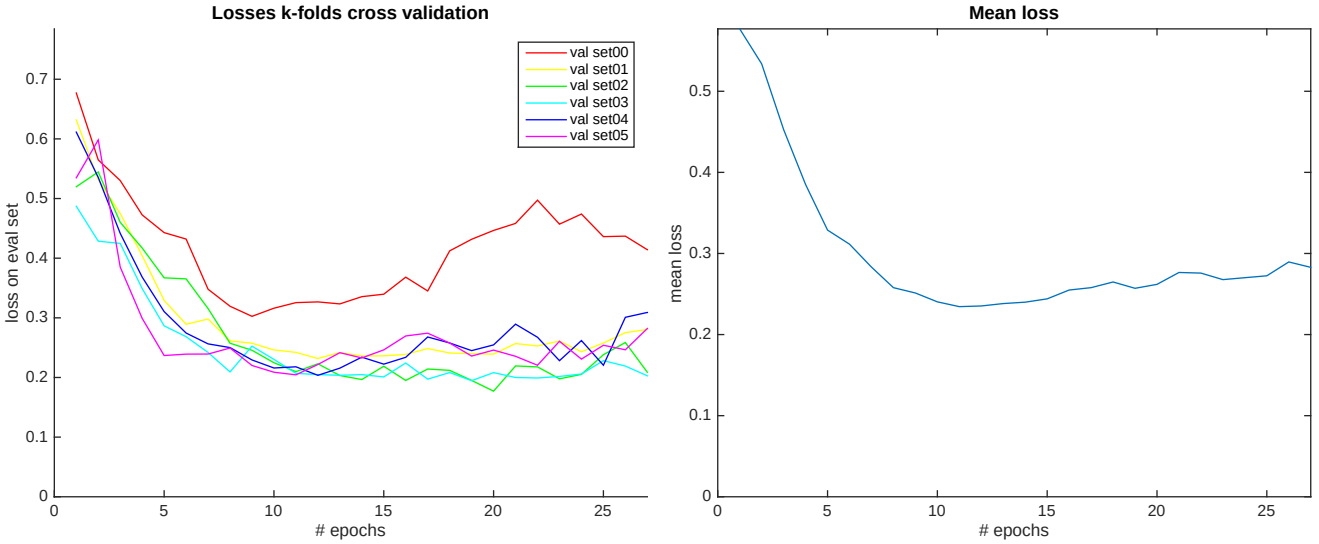

FIGURE 3.14: Losses obtained over the 6 possible validation sets and final average loss.

Taking the loss at multiple iterations and averaging the values extracted from the various validations set, as illustrated by eq. 3.14, the suggested model appears to be the one at 3500 iterations (see figure 3.13).

However, looking at the plot it is possible to observe as the loss function starts a plateau around iteration 3000 which goes up till iteration 5000.

This behavior is of fundamental importance in this particular kind of analysis since shows a continuos change of the net's weights, which however doesn't return a payoff in

term of loss reduction. Basically, the model is overfitting.
In order to avoid a performance degradation, a typical choice that may be applied consists in this case to take not simply the best recomended model, but the one at the beginning of the plateau.
Following this procedure and training the final SVM over the overall training set, it is in fact possible to see how in the end the model at 3000 iterations represents a better option since produces a LAMR equal to 25.43% instead of 26.18% for the one at 3500 iterations.
Unfortunately despite all our considerations, such result appears to be quite similar to the one proposed by LDCF and not enough to justify an application of AlexNet+SVM in the overall architecture.

Checking instead the loss produced by the K-folds approach, which averages the loss values at multiple epochs (figure 3.14), it is possible to see that the model suggested is radically different and corresponds to one at 4200 iterations (of the overall training set). However, also in this case it's evident that the loss function reaches a plateau around epoch 10 (~3800 iterations) and then slightly improves until iteration 4200. Applying the same procedure as before and checking the model produced at iteration 3800, it turns out that this particular choice is even worse than the one obtained at iteration 3000. For this reason, we preferred to apply as our cross validation approach the one explained at the previous point.

A comparison of the net selected by the 3 approaches is illustrated in figure 3.15.

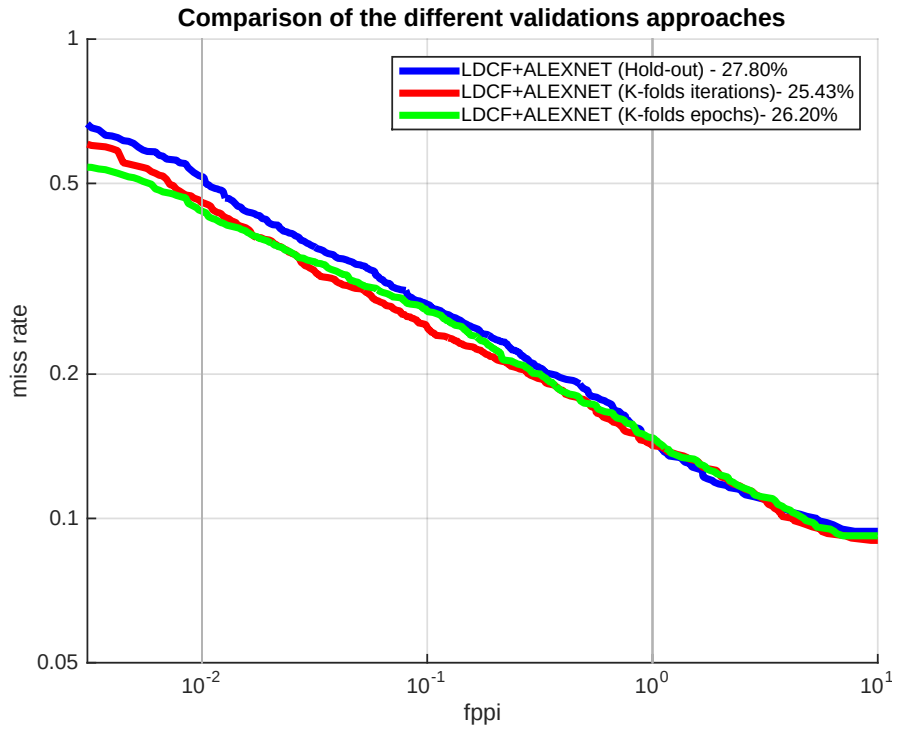

FIGURE 3.15: ROC obtained with the Hold-out, K-folds averaged on multiple iterations and K-folds averaged on multiple epochs approaches.

3.11 ACF/LDCF Thresholding

As expressed so far, despite the realized analysis, the performance produced by our final architecture is quite poor and doesn't justify the application of the finetuned DNN after the selected region proposal algorithms.
However, up to this point the architecture we have presented applied the proposal algorithm as only a tool aimed at retrieving sets of interesting regions, without discriminating any of them wrt the score assigned (exactly as it is realized in the standard R-CNN approach).
In our particular situation, ACF and LDCF are, to all intents and purposes, real detectors that return scores proportional to the confidence levels with which they identify the presence or absence of pedestrians.
Therefore, taking inspiration from the cascaded architecture that we have presented in Chapter 2 (where each stage of the cascade is meant to discard regions that most likely do not contain any targets), a double thresholding approach has been replicated in this work.

Basically, instead of letting the DNN to analyze every single region returned by the

proposer, a thresholding has been applied on the returned scores for discarding patches that can be considered with high confidence as "no-pedestrian".
It must be considered in fact that a region, to be classified as a positive sample, needs in this modification to be "approved" not only by the final DNN, but also by the proposal algorithm itself, reducing in this way the probability to have a misclassification wrt the basic approach (eq. 3.15).

$$P(\text{misclassification}) = P(\text{ACF/LDCF misclassifies}) * P(\text{DNN misclassifies}) \tag{3.15}$$

Probability to misclassify an input sample.

In order to apply such kind of approach however, a suitable value of the ACF/LDCF threshold should be determined for cutting enough wrong patches at the first stage, without discarding too many positive patches.
To determine therefore the right value of this measure, the LAMR at multiple ACF/LDCF thresholds has been computed over the validation set 05, showing how a good value for both LDCF and ACF appears to be around 85 (figure 3.16).

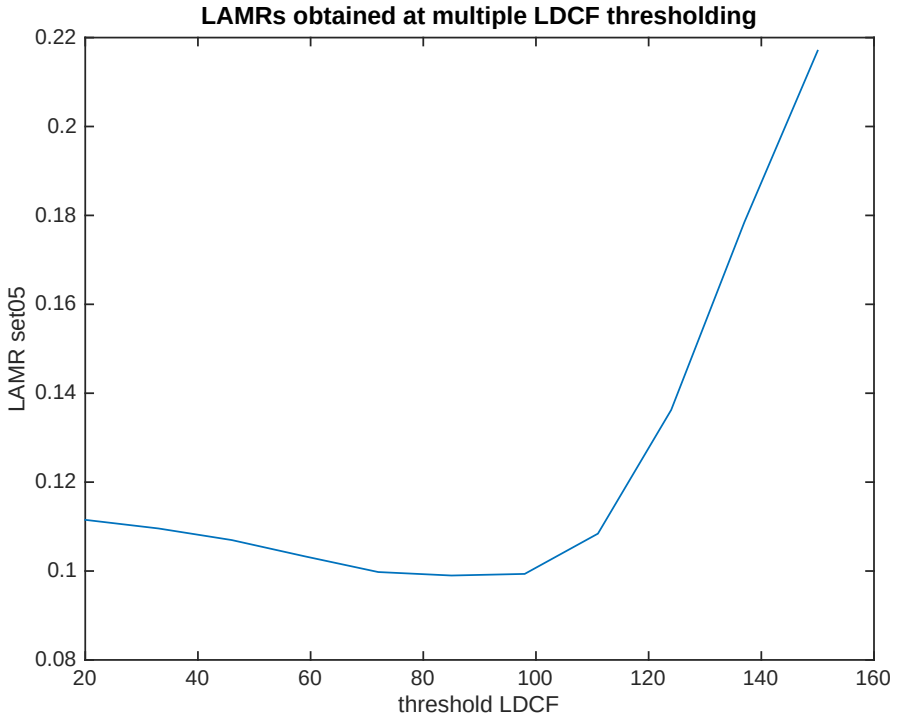

FIGURE 3.16: LAMR obtained over the set05 applying different thresholding on the scores returned by LDCF. The minimum LAMR is obtained with a threshold equal to 85.

Thanks to the application of this thresholding, the best finetuned AlexNet we selected during the validation procedure appears now to present a LAMR equal to 22.49% instead

of the original 25.43% (figure 3.17), significantly outperforming the best previous result introduced by Hosang et al. and achieving the 2nd position among all the detectors compared in [5] (the 1st considering only single frame detectors[3]).

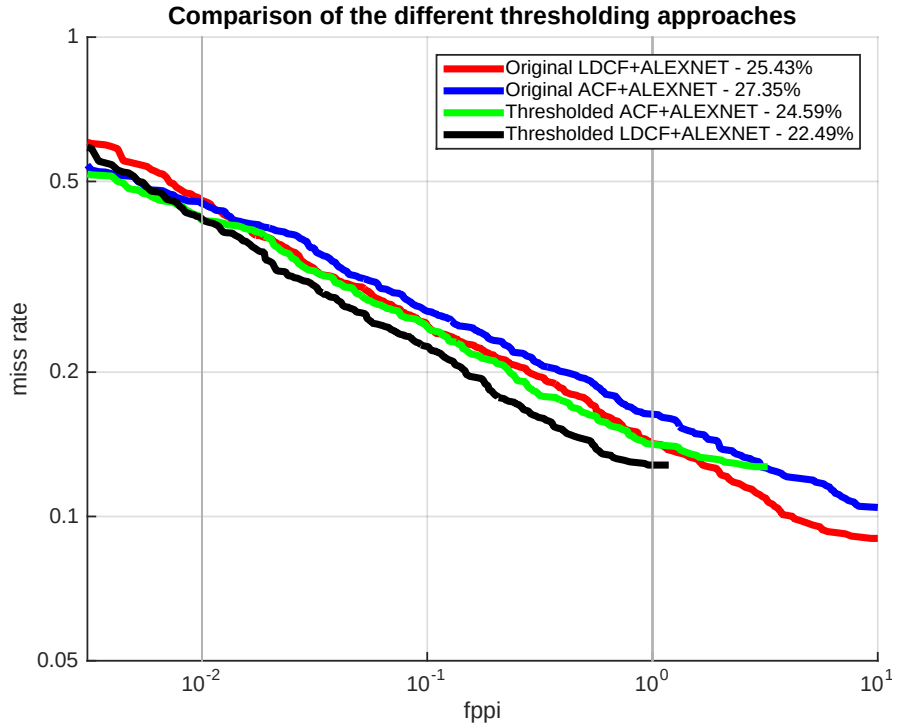

FIGURE 3.17: ROC obtained thresholding the BBs returned by ACF or LDCF and classifying only the approved proposals with our best validated AlexNet.

As it is possible to see, despite the better capability of ACF to retrieve good regions wrt LDCF, thanks to the higher discriminative capability of the latter, LDCF turns out to be the best proposal algorithm we may apply to obtain the best possible results.
However, due to the computational effort required by LDCF for computing the final results, in applications where the time is important (as the one presented in the next Chapter), ACF could still represent a better choice that deserves to be taken into account for the high process frame rates it presents.

[3]Detectors that do not use information coming from previous frames.

3.12 Overall Finetuning Process: a Tutorial

Summarizing what we have introduced in this section, starting from a network already trained on a very huge and vary dataset (such as ImageNet), our presented approach for building a DNN detector basically requires to:

1. Select the best proposal algorithm wrt the considered application (e.g. ACF, LDCF, Selective Search ...).
2. Select the best DNN wrt the operative conditions that must be faced (accuracy and speed represent interesting point to take into account).
3. Detect which is the best amount of padding to add to the proposed regions, in order to obtain patches that will likely contain all the targets that should be identified.
4. Build the largest possible training dataset in order to prevent overfitting:
   (a) Fix a positive:negative ratio to prevent the net converging to a model that discards all the input samples.
   (b) Use all the accepted GT annotations (e.g. height $\geqslant$ 50 px over the Caltech Dataset) for extracting positive samples.
   (c) Apply a random cropping approach for reducing the probability to overfit the data.
   (d) Determine the amount of context to introduce in the training samples to replicate typical operative conditions.
   (e) Randomly select the negative samples from the overall training set, for improving the variety of the final dataset.
5. Apply K-folds cross validation and average the various losses obtained at multiple iterations.
6. Finetune over the overall training dataset up to the amount of iterations suggested.
7. Optionally: apply multiple initializations and repeat the training to obtain the best possible result.

Following this procedure step by step and determining on a validation dataset the best threshold to apply over the proposer algorithm, we demonstrated how a final solution that not only approaches state of the art detectors, but also outperforms them, can

finally be obtained.
DNNs appear therefore as really useful tools for properly detecting and recognizing objects even in low quality images, as the ones situated in the considered dataset.

A real time application of this model with a suitably realized tracking system will be explored in the next chapter, showing how these models can be applied even in harder and different situations, highlighting the relative difficulties and strengths that show up in these particular contexts.

3.13 Appendix: NIN vs ALEXNET

In order to make things faster and try to give an alternative solution to the application of AlexNet over the Caltech Dataset, a solution based on a NIN architecture has also been investingated in this work.

The NIN architecture, differently from AlexNet, has a number of weights 10 times smaller thanks to the application of only a reduced number of fully connected layer wrt the multiple convolutive layers presented by AlexNet.
We decided to focus our attention also to this kind of networks for the comparable performance it achieved over the ILSVRC dataset, where it appeared as a valid and lighter alternative to the larger AlexNet.

Unfortunately, the operative conditions of the Caltech Dataset are so bad that any learning of the NIN network we realized turned out to be ineffective, producing a LAMR of around 50% over the Caltech test set.
No thresholding has been applied over the ACF/LDCF proposer in this case, due to the real high value returned by this reference measure, which pushed us to immediately discard this network for its low discriminability.

Despite this result, the main idea to look for lighter models capable at producing comparable results to the one realized with AlexNet is still valid. Future research could so be realized for finding other solutions capable at producing comparable results, but with lower computational efforts.

3.14 Appendix: Alexnet in Caffe

```
name: "AlexNet"
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mirror: true
    crop_size: 227
    mean_file: "ADDRESS OF THE MEAN FILE TO USE"
  }
data_param {
    source: "ADDRESS OF THE TRAINING DATASET"
    batch_size: 256
    backend: LMDB
  }
}
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  transform_param {
    mirror: false
    crop_size: 227
    mean_file: " ADDRESS OF THE MEAN FILE TO USE "
  }
data_param {
```

```
    source: " ADDRESS OF THE VALIDATION DATASET "
    batch_size: 50
    backend: LMDB
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
```

```
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm1"
  type: "LRN"
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm1"
  top: "conv2"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
```

```
  }
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm2"
  type: "LRN"
```

```
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
```

```
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu4"
  type: "ReLU"
```

```
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
```

```
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
```

```
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc7"
  type: "InnerProduct"
  bottom: "fc6"
  top: "fc7"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
```

```
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu7"
  type: "ReLU"
  bottom: "fc7"
  top: "fc7"
}
layer {
  name: "drop7"
  type: "Dropout"
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8 "
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8 "
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 2
    weight_filler {
```

```
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "fc8 "
  bottom: "label"
  top: "accuracy"
  include {
    phase: TEST
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "fc8 "
  bottom: "label"
  top: "loss"
}
```

3.15 Appendix: Solvers in Caffe

test_iter: 167	#number of batches that make the validation set
test_interval: 50000	#number of iterations to do before validate the networkover the #validation set
max_iter: 6000	#overall number of iterations to apply in the training phase
base_lr: 0.001	#base learning rate utilized during the finetuning
momentum: 0.9	#momentum value
weight_decay: 0.0005	#weight-decay factor value
gamma: 0.1	#lr drop factor
lr_policy: "step"	#type of solver to use (step means SGD)
stepsize: 20000	#number of iterations to do before drop the learning rate
display: 50	#number of iterations to do before display some result in the #console
snapshot: 50	#number of iterations to do before take a snapshot of the current #model
snapshot_prefix: "PREFIX"	#prefix that will be applied to each model snapshot
solver_mode: GPU	#defines where to run the solver, if in the CPU or in the GPU

Chapter 4

Detector Time Profiling Analysis and Proposed Real-time Architecture

4.1 Introduction

Up to now we have presented how the detection problem can be faced in order to retrieve the pedestrians located in an analyzed frame and which are the strongest and most powerful detectors we can use in order to accomplish this task.
However, despite the power and quality of the detector we trained so far, one of the main defects associated to DNN solutions is related to their speed. Due to their dimensions in fact, this kind of models appears typically slow in a lot of different HW architectures currently available in the market, requiring processing times which make these solutions hardly applicable to real-time scenarios.
Think for example to image-based car safety systems: in order to guarantee a good protection level to both people in the car and pedestrians, car safety systems need to acquire and process images in a really fast way, due to the high speed the vehicle may present. Unfortunately, images acquired in these conditions often present distortions, artifacts and a low quality in general, precluding the application of simple and fast detectors.

For this reason, a solution able to combine in a proper way, both the high speed of low quality detectors and the accuracy of stronger classifiers, may be demandable in this

kind of situations in order to extract good quality information, while keeping at the same time sufficiently high processing frame rates.
The solution we are going to illustrate in this chapter proceeds exactly in this way.
Applying a tracking-by-detection approach and exploiting the particular structure of the realized DNN detector, a suitable system capable at extracting good detections, while running at more than 10 fps[1], has been realized in the present work.

4.2 Selected HW Architecture

The first step in our analysis concerns in this case the definition of the underlying hardware architecture that we are going to use.

For simplicity, in view of the requirement that the prototype should combine multiple solutions mainly available in MATLAB (which does not work on ARM-based architecture), we decided to select as reference system a MacBook Pro (Retina, 15-inch, Mid 2014) with the following characteristic:

- CPU: Intel Core i7 2,8 GHz
- RAM: 16 GB 1600 MHz DDR3
- GPU: NVIDIA GeForce GT 750M 2048 MB (CUDA compatibility 3.0)

Which represents a good configuration for exploiting both the intrinsic parallelism available in DNN solutions (which fit well on modern GPU architecture) and the available ACF/LDCF implementations (which do not run on ARM configurations due to the large use of x86/x64 SIMD instructions they do).

Moreover, since in the literature also other algorithms are tested on configurations similar to the one exploited in this work (e.g. [14]), we believe that our indications could be of interest even in different situations from the one here considered.

[1]Which appears as a demandable property for automotive configurations as illustrated in [13]

4.3 Time Profiling Analysis of the Detectors

Defined the HW architecture, the next step involved in our analysis requires an estimation of the average frame rates produced by the various detectors.

Running on different images the available implementations and averaging the results, it turns out that the only algorithm capable at running with a frame rate larger than 10 fps is the simple ACF detector (thanks to the various approximations and reduced number of operations it applies).
All the other strong solutions run instead in the order of decades of frames, which is totally unacceptable for any real time application (see figure 4.1).

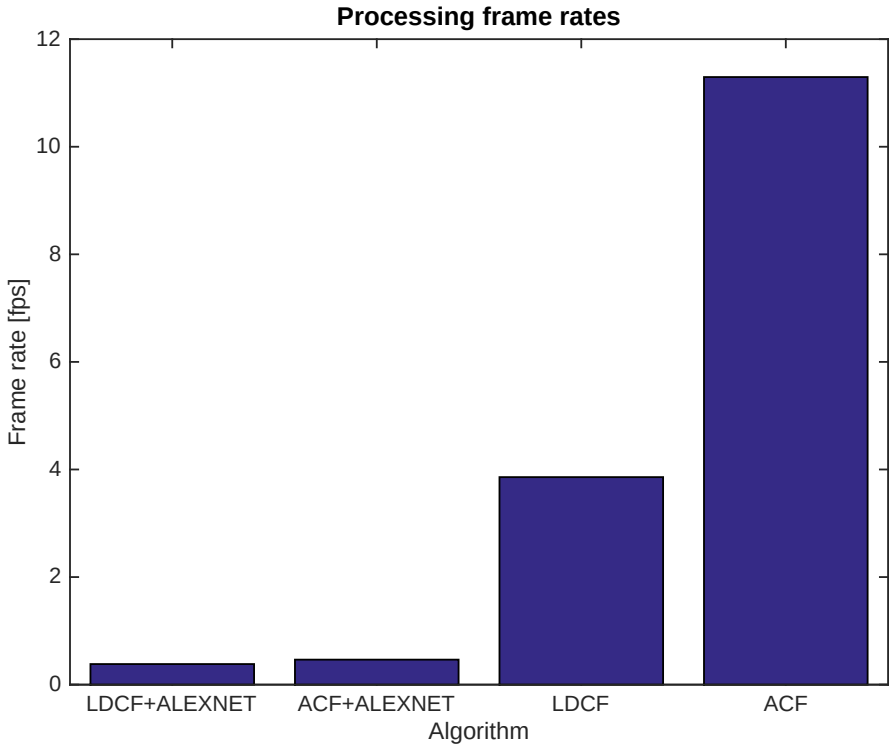

FIGURE 4.1: Processing frame rate extracted from the various algorithm over the reference HW architecture with the original configuration.

However, it must be considered that in order to exploit at best the massive parallelism available in modern GPUs, the original AlexNet model, if not suitably modified, collects all the regions extracted in batches made by 256 patches each one and only then proceeds with the extraction of the interested features. If the number of regions retrieved is not a multiple of 256, blank patches are added up to that limit.

Despite this particular optimization can be really useful whenever there is a large amount of data to analyze, in our situation is basically impossible to have 256 pedestrians in a single image. So, realize large batches filled only with a restricted number of patches is nothing more than a waste of time.

In order to estimate the optimal size our batches should present, an analysis aimed at identifying how many regions should be considered from the ones retrieved by the proposal algorithm, must then be carried on. The main idea is to avoid processing an excessive amount of regions, limiting to a fixed upper bound the number of patches that must be analyzed.

Considering in fact batches of only k regions, filled with the top-k ones retrieved by the proposal algorithm, the computational burden exposed by our NN over a single image can be significantly reduced.

Since LDCF appears already too slow to reach in the final architecture a processing frame rate at least equal to 10 fps, we decided in this case to consider as proposal algorithm in our case the lighter but still useful ACF.

Running this latter over all the training set and considering for each iteration only the top-k detections retrieved (sorted by score), a comparison of the miss rates produced by ACF and the NN processing time has been realized. The results obtained are presented in figure 4.2.

As it is possible to see the execution time appears almost linear wrt the dimension of the batches (jumps are due to the different number of warps the GPU will produce for different amounts of data), while the miss rate appears to decrease exponentially, reaching the minimum around 25 regions per image.

Since we are looking for a good configuration that allows to achieve a valid trade-off between the miss rate produced by the selected region proposal algorithm and the NN execution time, we decided to consider in this work only the top-15 BBs retrieved by ACF. Such configuration presents in fact:

- A miss rate equal to 16.2%, far from the minimum only 0.5%
- A contained NN execution time equal to 177.5 ms

Running again the ACF+AlexNet and LDCF+AlexNet detectors over the images of the Caltech Dataset, the new optimized configuration has been finally tested (figure 4.3).

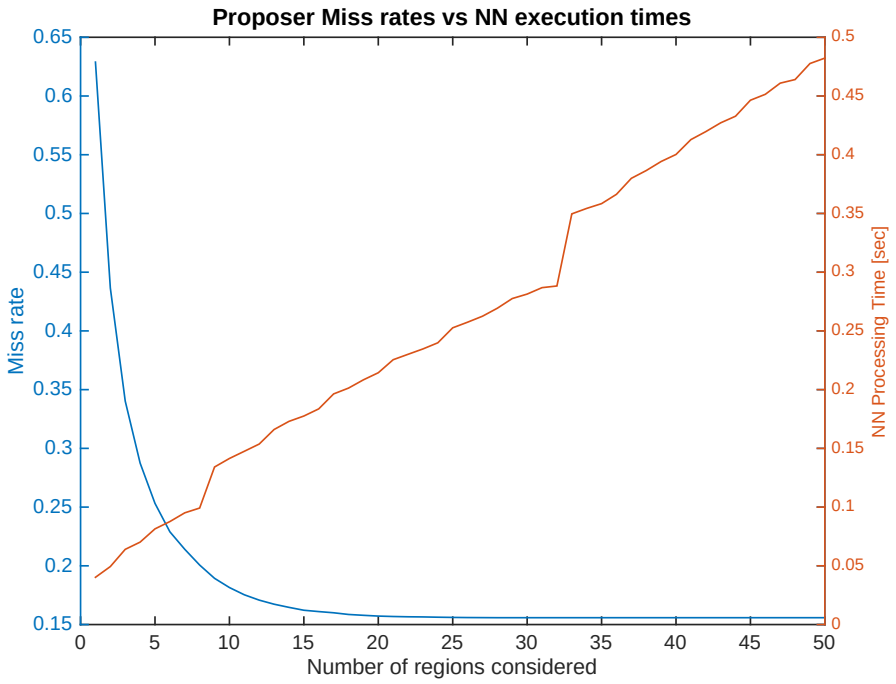

FIGURE 4.2: Miss rate produced by ACF considering only the top-k detections retrieved (with score larger than 80) vs the AlexNet execution time required to complete the analysis of the extracted patches.

As it is possible to see, LDCF+AlexNet is still requiring a considerable amount of time to complete a detection wrt the other solutions, due to the two strong classifiers that run one after the other. On the other hand, our ACF+AlexNet implementation not only increases its initial frame rate of almost 9 times but even outperforms the one presented by LDCF, representing for this reason a promising solution for both the speed of the selected proposal algorithm and the final accuracy it is able to achieve.

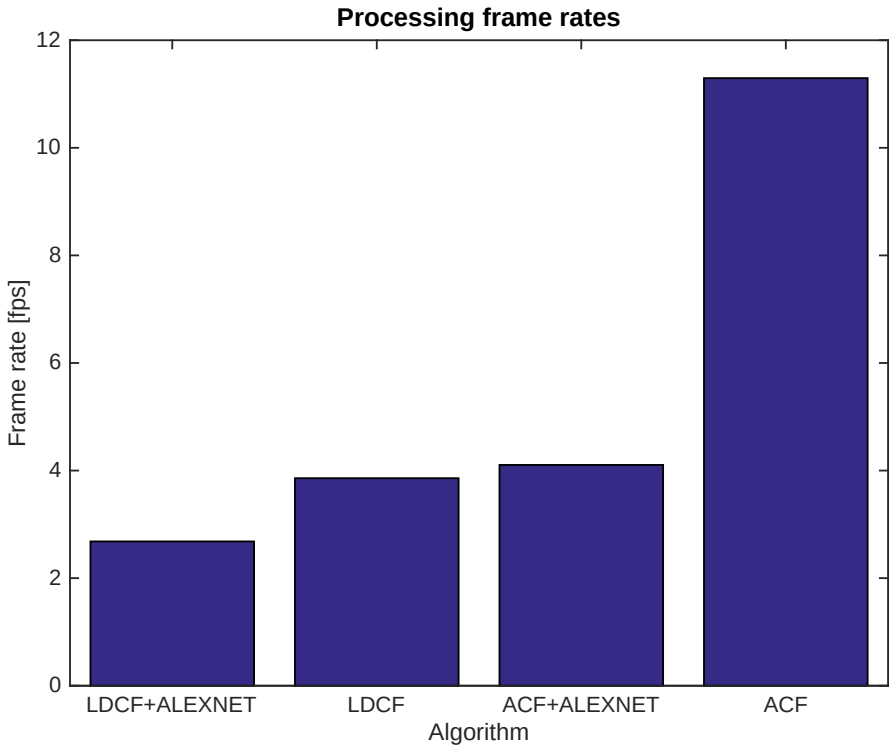

FIGURE 4.3: Processing frame rate extracted from the various algorithm over the reference HW architecture with our restricted batches.

4.4 Exploiting Intra-frame Information

In the previous chapter we introduced the problem of detecting pedestrians considering that each acquired frame is completely independent one to each other, which is a typical solution for single-frame detectors.

However, in any real-time application where there is a single camera recording images from the world, the information contained in any single frame is not uncorrelated to the one present in the following or previous ones.

Think for example to our pedestrians, if a person appears in the camera FOV of a given frame, it is really unlikely that he will disappear exactly in the next one, thanks to the high frame rate the acquisition system presents.

Information extracted in the past and associated to a particular pedestrian, which endures in the camera FOV for different frames, is so valuable not just for a single instant but for the entire life period of the pedestrian itself.

Such information can so be exploited to improve both speed and accuracy of any detection solution we decide to take into account, avoiding to repeat complex feature

extractions for every single acquired frame.
The solution we implemented in this work proceeds exactly in this way.

As illustrated in the previous section in fact, despite our optimization, any application of a strong detector (e.g. LDCF, ACF+AlexNet or LDCF+AlexNet) does not satisfy the target processing frame rate we would like to achieve, due to the related computational complexities these solutions present.
However, looking at the presented NN-based detectors it is possible to see that, differently from other solutions like ACF or LDCF, our NN detectors are not built by a single unique block but by a cascade of two: a complete stand alone detector and the NN itself. Hence, running these different components in parallel at their respective operating frequencies and merging in a proper way the different information extracted, we could potentially be able to exploit at the same time both the high speed of ACF and the good accuracy produced by our finetuned DNN.

In any case, solving this problem is not easy. All the information extracted should in fact be maintained along the time and associated to the correct pedestrians, avoiding to mix information associated to different targets, which could lead in the end to erroneous results.
To avoid this kind of problems and exploit at best the information extracted, a tracking system aimed at maintaining and coupling all the information extracted in the previous frames with the last available one, has been implemented.

Due to the complexity that such system presents, before proceeding with the presentation of our real-time architecture, in the following sections an introduction of the main algorithms implemented is presented, highlighting their main functionalities and the various problems they solve.

4.5 Hungarian Algorithm

Whenever we need to track objects frame by frame, a typical problem that arises concerns how to identify if a detected object is already tracked by the system and how to couple the extracted BBs with the already tracked ones.

In order to deal with this situation, the so-called Assignment Problem (i.e. the problem to identify a suitable assignment that couples tracks and detections one to each other, minimizing an overall cost function) needs so to be solved.

It appears clear that every full grid search, which tries to couple in a suitable way detections and tracks associated to a given frame, should be immediately discarded due to the excessive amount of computations it will require.
Fortunately in the 50s, Munkres et al. introduced an algorithm capable at solving the assignment problem with a polynomial complexity equal to $O(n^3)$ (where n is the number of elements in both the coupling sets), significantly reducing the cost of the overall assignment and making this kind of task available even for real-time applications. We are talking about the so-called Hungarian Algorithm [15].

In order to solve the Assignment Problem and determine which track is coupled with which detection, the Hungarian Algorithm requires the definition of a matrix that contains the costs needed to realize each possible assignment. A suitable measure aimed at identifying in a robust way which elements should be coupled with which one, needs so to be defined.
The measure we introduced in this case is presented in eq. 4.1 and simply represents the fraction of area not overlapped between a given track i and a detection j associated to the last acquired frame.
Since the higher this area, the smaller the probability that the match between track i and detection j is true, this measure represents a valid option that allows to match not just BBs which are closer, but also that have similar size.

$$cost_{i,j} = 1 - IoU_{i,j} \tag{4.1}$$

Where $\text{IoU}_{i,j}$ is the intersection over union among track i and detection j.

The definition of this measure alone, however, may not be sufficient to realize a satisfying tracks-detections matching. Think for example if a tracked pedestrian disappears from the camera FOV, it is clear in this case that the associated track should not match any of the retrieved detections and that any other result will be misleading and erroneous.

In order to identify this particular situation and check if an assignment is valid or not, instead of coupling only the available tracks with the various extracted detections, we decided to extend the generated cost matrix with some mock columns and rows, producing an associated extended version of $(n + m) \times (n + m)$ cells (where n is the number of tracks and m the number of detections).

Imposing a zero cost to all the cells of the added mock rows/columns where the associated tracks/detections does not present any detection/track with IoU larger than a given

threshold, it has finally been possible to force the Hungarian Algorithm at realizing an assignment where every unmatched element appears coupled with a fake one.

Analyzing in this way the produced result and checking the presence of any mock element in any retrieved couple, the identification of which elements do not match with any other one has finally been achieved.

In order to guarantee a sufficiently scale-invariant but at the same time robust algorithm, a relaxed version of the Pascal Criterion has been introduced, fixing the matching threshold for the Hungarian Algorithm at 0.8.

4.6 Kalman Filter

One of the major problem that may arise during the detection and tracking process is how to correctly assign detections identified in the past to information retrieved from the last available frame.

As we have presented in the previous section, a typical solution to solve the assignment problem and couple tracks with new detections retrieved from the last acquired frame, is given by the Hungarian Algorithm.
In order to realize correct and valid assignments, however, the BBs associated to the various tracks available in the system need to overlap as much as possible with the ones identified by the the detector and associated to the same targets, or low IoUs will be produced and no match will be realized.

While this problem can be negligible if our overall system is capable at processing really fast the received input frames, it becomes of fundamental importance whenever the targets present fast motions or the system suffer lacks in the processing frame rate.
In order to overcome this situations and produce stronger results, a solution based on Kalman Filters (aka KFs) has been introduced in this work.

Kalman Filters are a particular form of optimal estimators that infers measures of interest from indirect, inaccurate and uncertain observations, recursively updating the current state estimation of the target with the new noisy information that is progressively collected (i.e. the new observations).
They are defined as optimal estimators since if all the noises involved are gaussian (i.e. distributed according to a guassian pdf) and white (i.e. each noise assumes values over the time completely uncorrelated to the previous ones) Kalman Filters minimize the

mean square error of the estimated parameters. A property that makes this kind of solutions a really common choice nowadays for multiple applications where there is the need to:

- Estimate unmeasurable measures related to accessible observations.
- Predict the values of a stochastic process that we may be interested into.
- Clean collected observations from the noise they may contain.

Among the presented features, the one that is particularly interesting for us is for sure the prediction capability.
Exploiting the equations of the Uniform Accelerated Motion and considering that both the accelerations over the x and y axis change only according to an underlying white noise (that represents all the unknown dynamics associated to the car and pedestrians motions), the filter presented in eq. 4.2 has been realized. As it possible to see, no external information from the one that can be extracted from the various input frames is required to describe and predict the positions of our interested tracks.

$$\begin{cases} x(t+1) = x(t) + v_x(t) + e_x(0,\lambda) \\ v_x(t+1) = v_x(t) + a_x(t) + e_{v_x}(0,\lambda) \\ a_x(t+1) = a_x(t) + e_{a_x}(0,\lambda) \\ y(t+1) = y(t) + v_y(t) + e_y(0,\lambda) \\ v_y(t+1) = v_y(t) + a_y(t) + e_{v_y}(0,\lambda) \\ a_y(t+1) = a_y(t) + e_{a_y}(0,\lambda) \\ x_{out}(t) = x(t) + e_{x_{out}}(0,\eta) \\ y_{out}(t) = y(t) + e_{y_{out}}(0,\eta) \end{cases} \tag{4.2}$$

Where $x(t)$, $v_x(t)$, $a_x(t)$, $y(t)$, $v_y(t)$, $a_y(t)$ are position, velocity and acceleration along both the x-axis and y-axis of the target's center. Any e involved is a gaussian white noise with zero mean and variance equal to λ for state variable and η for output variables.

Exploiting this structure and building the related optimal predictor, better matches between detected BBs and the tracked pedestrians have been realized, predicting the various positions that the tracks may present in the last analyzed frames.

However, to obtain satisfactory results, an estimation of the λ and η variances of eq. 4.2 needs to be realized. A wrong configuration of this parameter may in fact produce poor results in the end (if not even harmful), since influences the importance of the extracted observations wrt the estimated states, changing the way we do predictions.

In order to estimate such variances, a simple approach based on a grid search has been applied in this work, extending the information contained in the Caltech Dataset and identifying a good set of values able to minimize the MSPE (i.e. Mean Square Prediction Error) produced by the filter (see section 4.8).

4.7 Proposed Architecture

After we have presented how to solve the assignment problem and how to predict the location of an identified pedestrian in the future, the next step of our analysis finally concerns the structure of the realized real-time architecture.

As already presented in section 4.4, exploiting a DNN based detector in real-time scenarios is not an available option due to the high latency these detectors present. In order to apply such strong solutions even in this kind of situations, the capability to merge in a suitable way the information coming from two different threads, which run the two distinct part of our detector in parallel, could be a demandable feature to produce high speed and accuracy.
So, in order to check which kind of results the presented proposal may achieve, a system implementing the following pipeline has been realized:

1. Whenever a new frame is available:
   (a) The finetuned AlexNet is checked in order to identify if it has completed the processing of BBs already extracted from previous frames.
   (b) If it has finished, the BBs with a score larger than a given threshold are retrieved with their relative positions and dimensions.
   (c) All the tracks available in the system and associated to these BBs are marked as pedestrians while all the others are suppressed.
2. Then, the system proceeds at extracting new information from the last received frame applying ACF over all the input image:
   (a) All the boxes with a score larger than a detection threshold are directly considered as pedestrians.
   (b) All the boxes with a score smaller than the detection threshold but larger than a tracking one are instead considered as valid, iff matched with a track already marked as pedestrian by the NN.

(c) Tracks which are not matched with any BB are finally suppressed.

All the match realized at this step are produced predicting the positions of the already detected pedestrians and applying the Hungarian Algorithm of section 4.5.

3. Finally, if the NN is free (i.e. it is not processing any other BB), all the BBs retrieved in the last frame and with a score larger than the tracking threshold are sent for further analysis to this stronger classifier and start new tracks.
   Among all the unmatched BBs, all the boxes with a score larger than the tracking threshold, but not the detection one, are considered as "to be validated" and not marked as pedestrians till the NN returns its final response.
   The main idea is to consider in each frame as pedestrians all the BBs which present an ACF score really large or have in the recent past been validated by the finetuned NN.

4. After all these steps, the system returns to point 1 and the overall process starts again, skipping all the frames received from the instant in which the analysis has started.

(See the activity diagram of figure 4.4 for a high level representation).

Despite all the algorithms implemented in this architecture are all very fast and do not require a lot of computational power. The necessity to have two threads that runs in parallel, the overhead required to handle the parallel processing and the time spent in memory transfers (needed to transfer the extracted BBs from the main memory to the GPU memory) weigh over the overall pipeline, reducing the processing frame rate produced by ACF from 11.3 fps to 8.3 fps.

In order to increase again the processing frame rate and reach at least the demandable 10 fps, some modifications over one of the two running components need to be applied. Since the NN runs in background over the GPU and does not influence for this the CPU processing (if not with memory transfers and initial preprocessing), our choice fell in this case over ACF.
Two different possibilities are available in this case:

- We can try to restrict the amount of pixels analyzed recurring to estimations of the regions where pedestrians can be situated (using for example the Stixel World introduced by Rodrigo et al. in [16]).
- We can try to speed up the classification process involved in our detector.

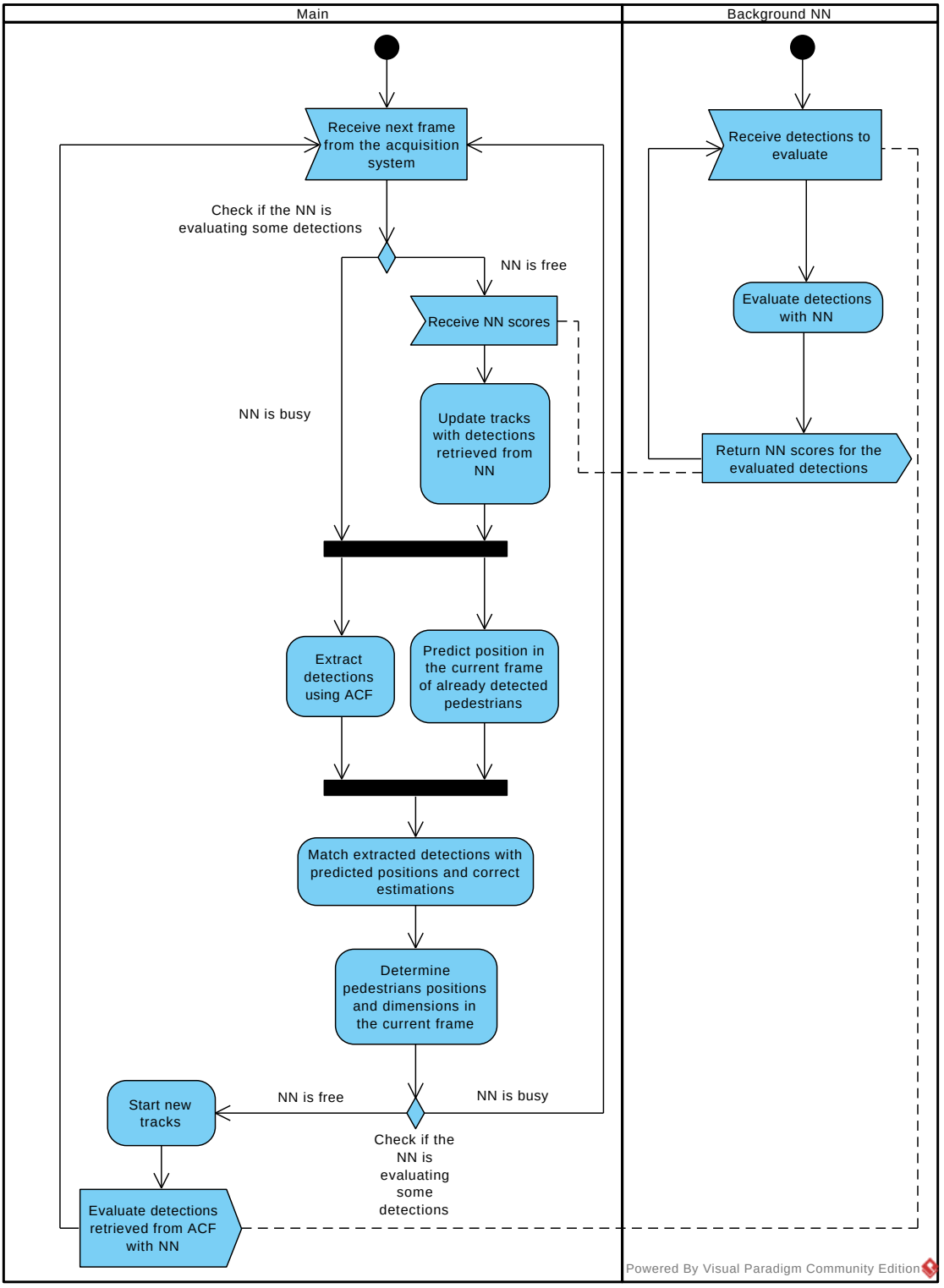

FIGURE 4.4: Activity diagram of the proposed architecture

In order to produce the more general results that can be easily extended even to different situations from the well known pedestrian detection task, we decided to focus our attention in this case over the analysis of the classification process.
Analyzing this process, it turns out that in order to retrieve with good accuracy all the available pedestrians situated in the received input frames, in their last ACF version, Piotr et al. decided to introduce as their final pedestrian classifier a soft-cascaded classifier [17] instead of a simply cascaded one.

In a soft-cascade classifier instead of having multiple distinct cascade stages (each one with its related thresholding process), a single long boosted classifier is trained and multiple thresholdings are applied over the partial cumulative scores returned by the various single weak classifiers. Namely, given a boosted classifier H_K with K weak classifiers defined as in eq. 4.3.

$$H_K(x) = \sum_{i=1}^{K} \alpha_i h_i(x) \tag{4.3}$$

Where $h_i \in \{-1, 1\}$ is the score returned by the i-th weak classifier and α_i its associated weight.

An input BB is considered as a pedestrian only if:

$$H_k(x) > \theta + \phi * k/K \quad \forall k \in \{1, ..., K\} \tag{4.4}$$

Where $H_k(x) = \sum_{i=1}^{k} \alpha_i h_i(x)$ and θ and ϕ are the so-called rejection threshold and the recalibrating parameter.

The combination of the rejection threshold θ and recalibrating parameter ϕ allows to discard BBs that present small partial cumulative scores in early stages, applying at the same time a more and more strict filtering as the cumulative score becomes determined by more classifiers.
Tuning the values of θ and ϕ is possible in this way to adjust the trade off between accuracy and speed, determining the amount of BBs that should be discarded in early stages wrt the ones that instead will be retrieved in the end by the algorithm. Obviously the larger the amount of BBs we decide to retrieve in end, the higher the computational burden of the overall classifier.

Following these theoretical considerations, multiple combinations of θ and ϕ have been tested in our work over the Caltech Set 05, producing the results presented in figure 4.5.

In order to guarantee a minimum amount of margin over the processing frame rate, which is strictly image-content dependent, only the configurations that produce a processing frame rate around 15 fps (i.e. $fps \geqslant 14$) have been considered.
As it is possible to see, the most promising configuration appears to be the one with:

- $\theta = -0.75$
- $\phi = 0.010$

since produces the minimum LAMR while keeping at the same time a reduced computational burden.
Running our architecture with this new selected configuration and checking the operative frequency, it turns out that our system is now capable at running at 14.96 fps over the Caltech test set, which is satisfactory for our requirements.

Since the different configuration applied in the ACF soft-cascaded classifier produces in the end different detections wrt the original ones that may be extracted, in the following, in order to realize a meaningful comparison capable at showing the real improvements introduced by our solution, all the architectures that will be tested must be considered as implemented with this optimized values of θ and ϕ.
We will refer in the following to this particular version of ACF as "fast ACF".

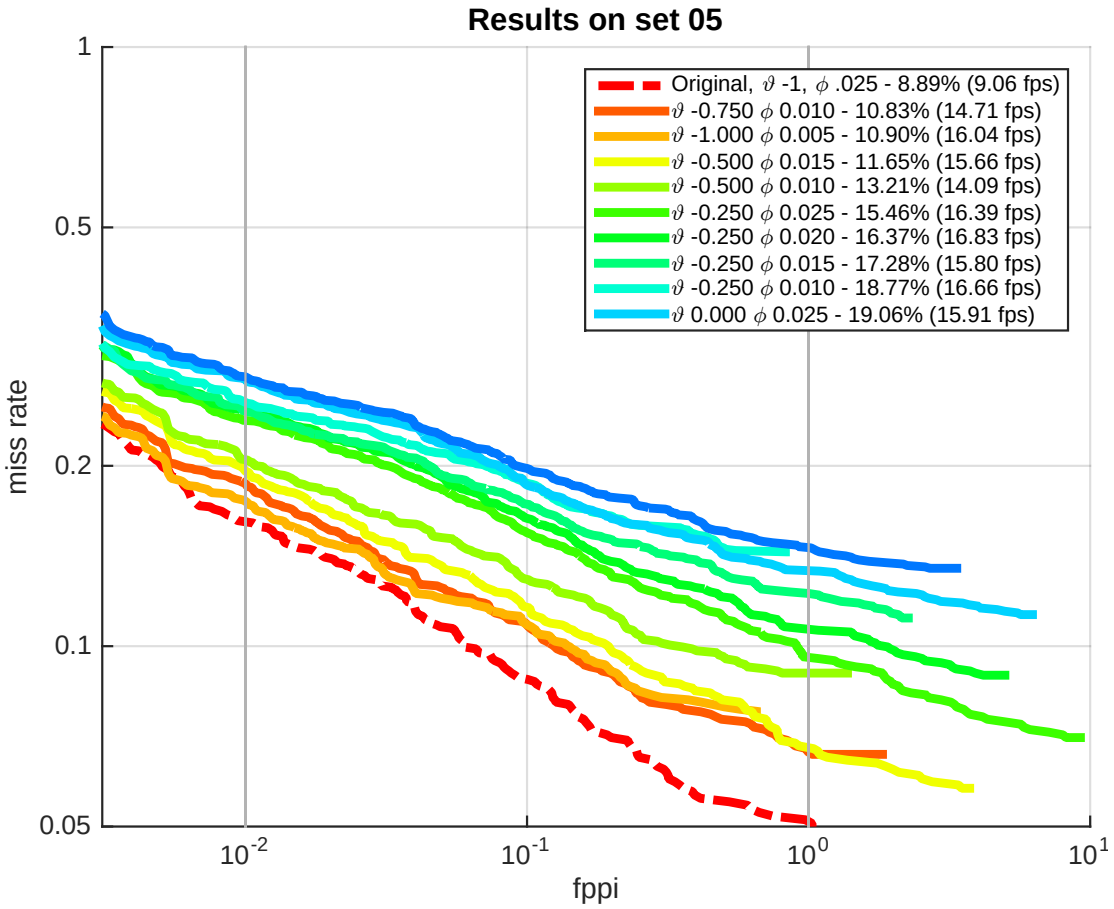

FIGURE 4.5: ROCs of the top-10 configurations tested that present a processing frame rate around 15 fps. For comparison, also the ROC of the original ACF tested over the Caltech set 05 has been presented.

4.8 Caltech Dataset Extension and KF Variances estimation

As we have illustrated in section 4.6, the estimation of the parameters λ and η involved in our implemented KFs is of fundamental importance to produce appreciable results. So, in order to find a good estimate of variances able to produce satisfactory results with the implemented architecture, a grid search aimed at checking the performance of multiple combinations of variances has been realized in this work.

However, in order to realize such estimate, an extension of the already presented Caltech Dataset needs first to be realized.
It must be considered in fact that, while this particular dataset is suitable for detection

tasks due to the multiple annotations introduced over every frame, it misses every possible tracking information. For example, we know if there are pedestrians in an image, but we do not know if taken two images the same pedestrian is situated in both of them and where.

To overcome this situation and introduce tracking information over the Caltech Dataset, a solution based on the already presented Hungarian Algorithm has been applied.
All the GTs situated in two consecutive frames, which have a sufficiently large IoU, have in fact been considered as a unique pedestrian and matched by a unique ID.
The dataset obtained in this way is not anymore a simple representation of detections, but contains also tracking information associated to the movements that each singular pedestrian applies frame by frame (which is of fundamental importance in our case).

Thanks to the extension here introduced, the following procedure has been finally applied:

1. First of all, the fast ACF computed in the previous section has been applied over all the images of the Caltech Set 05 used as reference for this estimation.

2. Acquired all the detections retrieved by ACF, all the extracted BBs have been matched with the underlying GTs, following as usual the well-known Pascal Criterion.

3. Once all the GTs available in the Caltech Set 05 have been coupled with the various detected BBs, all the 190 pedestrians situated in such set have been extrapolated.

4. For each of them a suitable KF has been initialized with the BB associated to the first time where the pedestrian appears, considering as initial state estimate null velocities and null accelerations.

5. Finally, a set of predictions and corrections have been carried on with different values of λ and η in order to estimate the quality of each configuration.
   For each prediction the associated prediction error (i.e. the euclidean distance between the predicted center of the analyzed BB and the associated GT center) has been computed and an overall Root Mean Square Prediction Error (RMSPE) has been evaluated (eq. 4.5). The results obtained are illustrated in figure 4.6.

$$RMSPE = \sqrt{\frac{1}{N}\sum_{p=1}^{P}\sum_{n=1}^{N_p} d_{p,n}^2} \tag{4.5}$$

Where P is the overall number of available pedestrians, N_p is the overall number of frames the p-th pedestrian appears, N is the overall number of predictions computed and $d_{p,n}$ is the distance between the predicted center of pedestrian p at its n-th occurrence and its associated GT.

As it is possible to see, the RMSPE reaches its minimum for:

- $\lambda = 10^2$
- $\eta = 10^5$

appearing for this as a good estimate of our desired variances.

All the estimates have been computed considering the covariance matrix of the initialization equal to:

$$P_1 = I * \alpha \tag{4.6}$$

Where P_1 is the covariance matrix of the initialization and I the identity matrix

Where α has been fixed to a high value (i.e. 10^5) in order to quickly loose the wrong initialization of velocities and accelerations imposed in the filter.
Experimental results has shown that even changing the value of α, the minimum reached by the best λ and η basically presents the same value of the one produced by our selected configuration.

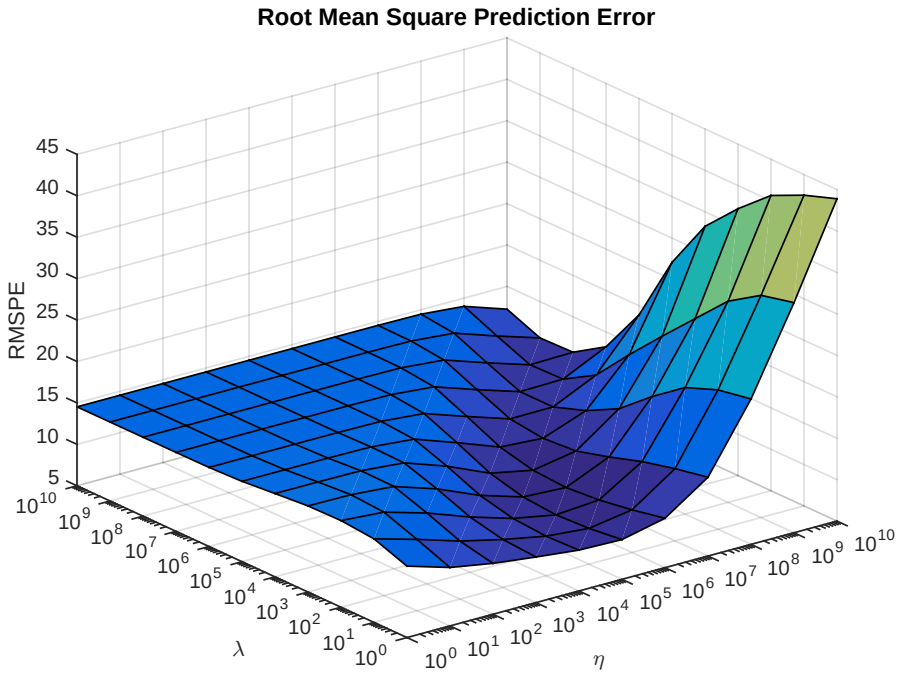

FIGURE 4.6: Root Mean Square Prediction Error extracted from set 05.

4.9 Performance Evaluation

Presented our architectural proposal and estimated all the parameters involved in the various implemented algorithms, the last step that remains to be faced is the performance evaluation of our solution.

Differently from the one implemented in the previous chapter, the evaluation we need to implement in our case is slightly different wrt the one presented for the simple detection task. It must be considered in fact that, applying our proposed architecture as exactly presented in the previous section:

- Two different scores are involved instead of only one, due to the two different classifiers involved in the detection process.
- Different sets of detections may be retrieved if the proposed solution is applied multiple times over the same set of images, due to fluctuations in the performance of the underlying HW architecture, which bring the system to skip different amounts of frames time by time.

In order to give a well defined evaluation of the performance produced by our proposal, more information needs so to be extracted from the presented architecture, to handle both the multiple scores retrieved by the system and the different performance produced at multiple runs. In particular what we need is to:

- Identify which are the average operative frequencies of both the main thread and the underlying NN, in order to realize a unique set of detections representative of the average performance produced by the system.
- Check the detection performance at variations of all the thresholds involved in our system, in order to extract the real power of our system at all the configurations it may be configured.

Proceeding in this way, the frame rates associated to the main thread and to the underlying AlexNet have been collected. The results are presented in table 4.1.

	Frame Rate
main thread	14.96 fps
AlexNet	4.21 fps

TABLE 4.1: Main thread and of our underlying AlexNet processing frame rate.

As it is possible to see, while the main thread is actually able to efficiently process 1 frame over 3, the NN does not.
Therefore, in order to realize a set of detections that represents the performance of the realized system at the computed average frame rates, all the BBs extracted in our analysis have been collected analyzing exactly one frame over 3 with our fast ACF, while considering a NN detection only every 9 frames (see figure 4.7).

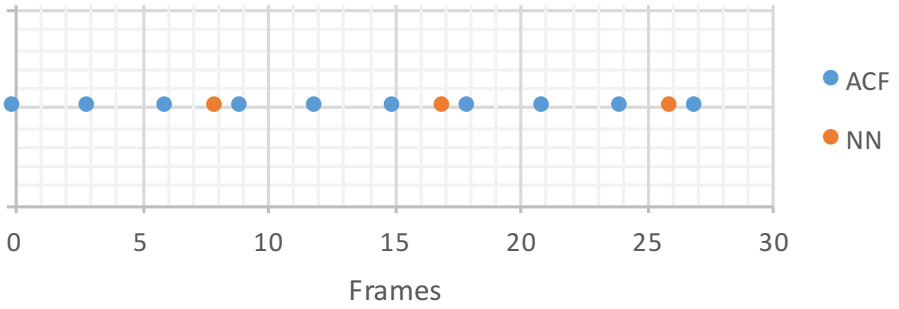

FIGURE 4.7: Detections realized by our architecture if applied in real-time.

Collected the detections and recorded all the scores returned by ACF and the NN, the overall realized system appears now just as a complex detector that returns scores

of multiple natures over the retrieved BBs. In order to evaluate the performance of such system the already presented ROC can now be computed simply varying all the 3 thresholds involved in our architecture (i.e. the ACF detection threshold, ACF tracking threshold and NN detection threshold) and extracting the related fppi and miss rates produced.
The results obtained in this way are presented in figure 4.8.

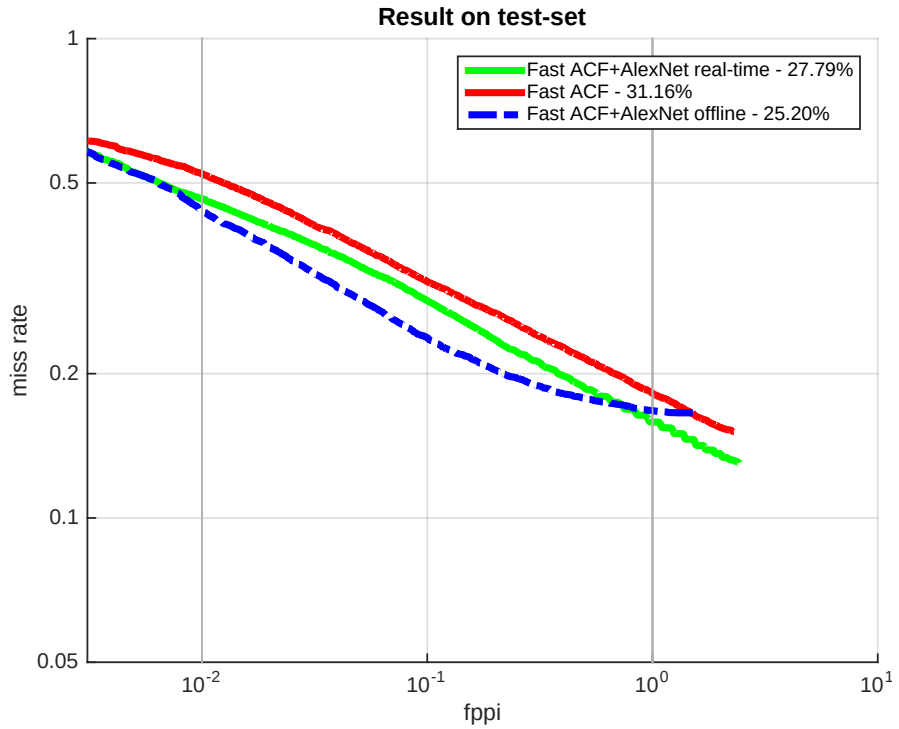

FIGURE 4.8: ROC of our fast ACF compared with the one of our architecture and with the one of our stronger classifier applied over the Caltech10x Test Set .

As expected, the results produced by our architecture are placed in the middle between the one produced by the plain ACF and the stronger ACF+AlexNet, representing a good trade-off between the accuracy and speed generated by the two presented solutions, especially near typical operative point (e.g. $\sim 10^{-2}$ fppi).

However, we could wonder why the presented architecture does not approach better the performance produced by the stronger ACF+AlexNet.
Despite our finetuned AlexNet plays an important role in the overall system, it must be considered that the amount of BBs it can influence is limited and strictly dependent by:

- The capability of the system to match a detection with a track already available.
- The number of frames a single pedestrian appears in the camera FOV.

In fact:

- If a pedestrian appears for few frames, the system simply has not the time for proceeding with a strong classification and the implemented NN could appear as useless.
- If a pedestrian moves too fast, it could happen to produce inaccurate predictions, reducing the capability to match the available tracks to the last retrieved BBs and loosing the possibility to exploit all the information extracted from the underlying NN.

Following this reasoning and looking at the percentage of matched detections that are also evaluated by the NN, it appears in fact that only the 60% of the retrieved BBs are also evaluated by the NN, while all the others are analyzed only by the presented fast ACF.
The speed presented by the selected fast detector and the underlying stronger classifier is so matter of interest, which should be properly considered in order to produce good results in every context the presented architecture could be taken into account.

Analyzing in our case the performance produced over the Caltech Dataset, the ROC produced by our solution appears to outperform in every single operative point the one produced by ACF, while working at the same time at a near operative frame rate.
The architecture realized in this chapter appears for this as a promising solution to take into account whenever not just the accuracy is relevant, but also the time required to compute the results plays a fundamental role.

Chapter 5

Conclusions and future works

In the presented work a refined and precise way to train DNN classifiers for accurate pedestrians detections has been presented.
Thanks to the illustrated method, the finetuned AlexNet has been capable at achieving really good performance over the Caltech test set, outperforming the previous state of the art and giving the possibility to realize an interesting solution for real-time applications.

Multiple aspects have been evaluated during the training procedure, highlighting the importance to:

- Augment the amount of data available in the training phase.
- Select with accuracy the negative regions proposed by the proposal algorithm.
- Apply suitable validation procedures aimed at exploiting at best all the available training set.
- Exploit as much as possible not just the information retrieved by the final strong classifier, but also the one extracted by the selected proposal algorithm.

A set of useful guidelines has been presented in this way, which we hope can help during the training of such complex and deep classifiers, even in different contexts from the one here considered.

Besides the training procedure, a suitable system capable at running at more than 10 fps and exploiting the network trained with our procedure has also been presented.
Recurring to the well-known tracking-by-detection approach, a combination of a lightweight

detector and a strong classifier has been illustrated, showing how fast and accurate solutions can be realized simply maintaining along the time information already extracted from preceding processed frames.
The selected HW architecture played a fundamental role in our system, since the possibility to execute multiple elaborations in parallel, without affecting the latency of each single component (i.e. ACF and AlexNet), represented the key for increasing the amount of information extracted without affecting the operative frame rate.

Due to the magnitude of the problem here presented however, the proposed solutions should not be considered as the end of this work, but further studies aimed at improving both the accuracy and the frame rate produced by the overall system should be explored.

In particular:

- Solutions based on cascades of DNN might be of interest for the good detection performance and the contained computational burdens they may present.
  Thanks to the reduced amount of computations these solutions dedicate in fact to clear background regions, good accuracies and good execution times are generally achieved, representing in this way valid solutions for fast and accurate detections.

- Studies aimed at checking the applicability of DNN classifiers over feature space different from the initial RGB channels may be realized.
  An enhancement in the quality of the final solution could in fact be achieved applying such complex classifiers not simply over the acquired RGB channels, but over a set of feature channels a priori computed (which represent a much stronger starting point over whom instruct and apply the selected network).

Other researches could so be devoted in this sense to an analysis of these and other solutions, looking for different and maybe more powerful NNs capable at further improving the performance we have here presented.

Bibliography

[1] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] P. Dollar, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36 (8):1532–1545, Aug 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2300479.

[4] Woonhyun Nam, Piotr Dollár, and Joon Hee Han. Local decorrelation for improved pedestrian detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2014.

[5] Jan Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele. Taking a deeper look at pedestrians. *arXiv preprint arXiv:1501.05790*, 2015.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[10] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[11] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[12] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[13] Hyunggi Cho, Paul E Rybski, Aharon Bar-Hillel, and Wende Zhang. Real-time pedestrian detection with deformable part models. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 1035–1042. IEEE, 2012.

[14] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418. IEEE, 2013.

[15] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[16] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE, 2012.

[17] Lubomir Bourdev and Jonathan Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243. IEEE, 2005.