Handling the speed-accuracy trade-off in deep-learning based pedestrian detection systems

Ujjwal Ujjwal

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Gestion du compromis vitesse-précision dans les systèmes de détection de piétons basés sur apprentissage profond

Ujjwal

INRIA Sophia Antipolis, STARS

Présentée en vue de l'obtention du grade de docteur en Automatique, Traitement du Signal et des Images d'Université Côte d'Azur
Dirigée par : François Brémond
Co-encadrée par : Aziz Dziri
Soutenue le : Novembre 13, 2019

Devant le jury, composé de :
Président du jury :
Frédéric PRECIOSO, Professeur, Université Côte d'Azur
Rapporteurs :
Christian WOLF,Maître de conférences, HDR, INSA de Lyon
Alexandre Alahi, Professeur assistant tenure track, EPFL
Thierry Chateau, Professeur, Université Clermont-Auvergne

Inria informatics mathematics

INSTITUT VEDECOM
DU VÉHICULE DÉCARBONÉ ET COMMUNICANT ET DE SA MOBILITÉ

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

PHD THESIS

Handling the speed-accuracy trade-off in deep-learning based pedestrian detection systems

Ujjwal

INRIA Sophia Antipolis, STARS

To obtain the title of Ph.D. of science in Signal and Image Processing
of the Université Côte d'Azur
Thesis Supervisor: François Brémond
Co-Supervisor : Aziz Dziri
Defended on : Novembre 13, 2019

In front of the jury composed of :
President of the jury:
Frédéric PRECIOSO, Professor, Université Côte d'Azur
Reporters:
Christian WOLF,Maître de conférences, HDR, INSA de Lyon
Alexandre Alahi, Assistant Professor tenure track, EPFL
Thierry Chateau, Professor, Université Clermont-Auvergne

Inria informatics mathematics

INSTITUT VEDECOM
DU VÉHICULE DÉCARBONÉ ET COMMUNICANT ET DE SA MOBILITÉ

GESTION DU COMPROMIS VITESSE-PRÉCISION DANS LES SYSTÈMES DE DÉTECTION DE PIÉTONS BASÉS SUR APPRENTISSAGE PROFOND

Ujjwal
Directeur de thèse: François Bremond
STARS, Inria Sophia Antipolis, France

RÉSUMÉ

L'objectif principal de cette thèse est d'améliorer la précision des systèmes de détection de piétons à partir d'image, basés sur l'apprentissage profond sans sacrifier à la vitesse de détection. Pour ce faire, nous effectuons d'abord une analyse quantitative systématique des diverses techniques de détection de piétons à partir d'image. Cette analyse nous permet d'identifier les configurations optimales des différentes composantes d'un système de détection de piétons. Nous examinons ensuite la question de la sélection des meilleures couches convolutionnelles pour extraire les caractéristiques visuelles pour la détection des piétons et proposons un système appelé Multiple-RPN, qui combine plusieurs couches convolutives simultanément. Nous proposons le système Multiple-RPN en deux configurations - une fusion-tôt et une fusion-tardive; nous démontrons ensuite que la fusion-tôt est la plus performante, en particulier pour la détection de piétons de petites tailles et les cas d'occultation de piétons. Cette étude fournit aussi une évaluation quantitative de la sélection des couches convolutionnelles. Nous intégrons ensuite l'approche de la fusion-tôt avec une étape de segmentation pseudo-sémantique pour réduire le cout de traitement. Dans cette approche, la segmentation pseudo-sémantique permet de réduire les faux positifs et les faux négatifs. Ceci, associé à un nombre réduit d'opérations, permet d'améliorer simultanément les performances de détection et la vitesse de traitement (20 images/seconde) ; les pe.fo
rmances sont compétitives avec celles de l'état de l'art sur les bases de données caltech-raisonable (3.79% de taux d'erreurs) et citypersons (7.19% de taux d'erreurs). La dernière contribution de cette thèse est la proposition d'une couche de classification des détections potentielles, qui réduit encore le nombre d'opérations de détection. Il en résulte une réduction de la vitesse de détection (40 images/seconde) avec une perte minime de performance de détection (3.99% et 8.12% de taux d'erreurs dans les bases de donnéealtech-raisonable et citypersons respectivement) ce qui reste compétitif avec l'état de l'art.

Mots clés: Vision par ordinateur, Détection de piétons.

HANDLING THE SPEED-ACCURACY TRADE-OFF IN DEEP-LEARNING BASED PEDESTRIAN DETECTION SYSTEMS

by
Ujjwal
Supervisor: François Bremond
STARS, Inria Sophia Antipolis, France

ABSTRACT

The main objective of this thesis is to improve the detection performance of deep learning based pedestrian detection systems without sacrificing detection speed. Detection speed and accuracy are traditionally known to be at trade-off with one another. Thus, this thesis aims to handle this trade-off in a way that amounts to faster and better pedestrian detection. To achieve this, we first conduct a systematic quantitative analysis of various deep learning techniques with respect to pedestrian detection. This analysis allows us to identify the optimal configuration of various deep learning components of a pedestrian detection pipeline. We then consider the important question of convolutional layer selection for pedestrian detection and propose a pedestrian detection system called Multiple-RPN, which utilizes multiple convolutional layers simultaneously. We propose Multiple-RPN in two configurations – early-fused and late-fused; and go on to demonstrate that early fusion is a better approach than late fusion for detection across scales and occlusion levels of pedestrians. This work furthermore, provides a quantitative demonstration of the selectivity of various convolutional layers to pedestrian scale and occlusion levels. We next, integrate the early fusion approach with that of pseudo-semantic segmentation to reduce the number of processing operations. In this approach, pseudo-semantic segmentation is shown to reduce false positives and false negatives. This coupled with reduced number of processing operations results in improved detection performance and speed (20 fps) simultaneously; performing at state-of-art level on caltech-reasonable (3.79% miss-rate) and citypersons (7.19% miss-rate) datasets. The final contribution in this thesis is that of an anchor classification layer, which further reduces the number of processing operations for detection. The result is doubling of detection speed (40 fps) with a minimal loss in detection performance (3.79% and 8.12% miss-rate in caltech-reasonable and citypersons datasets respectively) which is still at the state-of-art standard.

Keywords: Computer vision, Pedestrian Detection.

To Maa, Shubhra and Taru.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Pedestrian Detection

Pedestrian detection refers to the determination of regions in an image (*or in individual frames of a video stream*), such that each determined region contains one pedestrian. As such, it is a specific instance of the more general problem of object detection in computer vision – i.e limited only to pedestrians. Pedestrian detection is usually studied as a separate problem owing to its several applications which were recognized quite early on – e.g. *video surveillance*, *person tracking* and *autonomous driving*. Earliest works on pedestrian detection [4, 11, 156, 125, 111, 173] have primarily centered on the aforementioned

application domains. In the aforementioned applications like *autonomous driving* and *video surveillance*, the performance of a pedestrian detection system has a major cost factor in terms of *safety concerns*. For instance, failure of an autonomous driving system to detect a pedestrian crossing a street can result in mishaps with a potential to damage human life. Since late 1980's, when the first experiments with prototypes for autonomous vehicles [151] and automated *video surveillance* [62, 121] were carried out, these applications have entered mainstream usage on a large scale. This has led to a corresponding escalation in the safety concerns associated with the performance of pedestrian detection systems [78]. The large-scale adoption of these systems is expected to steadily increase in the future [59, 26]. In [27], it is reported that in real-life scenarios, contemporary detectors' ability to detect pedestrians in advance of fatal collisions vary from $< 30\%$ to $> 90\%$ of fatalities. This range needs to be bridged towards lower values to ensure the feasibility of *self-driving* cars in real-life scenarios on a large-scale. Thus despite being worked upon since 1980's, pedestrian detection is still a relevant problem and plays a key role in ensuring the safety of modern applications such as *autonomous driving*.

From the previous discussion we gather that, in real-life pedestrian detection systems, simply detecting pedestrians is not important on its own; it has to be coupled with the ability to detect pedestrians well in advance. The fundamental objective of this thesis is to jointly address these coupled problems of *detection accuracy* and *detection speed*; a fundamental practical problem in autonomous vehicles. From a technical perspective, pedestrian detection and its parent problem of object detection are formulated as machine learning problems. In the next section, we understand the nature of pedestrian detection as a machine learning problem. This renders greater clarity to our understanding of the technical challenges of pedestrian detection and hence recognizing the technical areas of our contribution.

1.2 Pedestrian Detection as a Machine Learning Problem

Pedestrian detection is a localization problem. This means that a pedestrian detection system needs to delineate locations in an image with pedestrians by drawing a rectangular bounding box around the pedestrians This localization problem in machine learning terms is a union of two problems – a) A two-class *classification* problem where every image-region must be classified for the presence or absence of a pedestrian and, b) A *regression* problem where given a set of features representing an image region, estimation of the 4 corner coordinates of the rectangle bounding a pedestrian at that location; if one exists; must be made.

This is an appropriate stage to point out some technical differences between pedestrian detection and its parent problem of object detection. Firstly, object detection is a

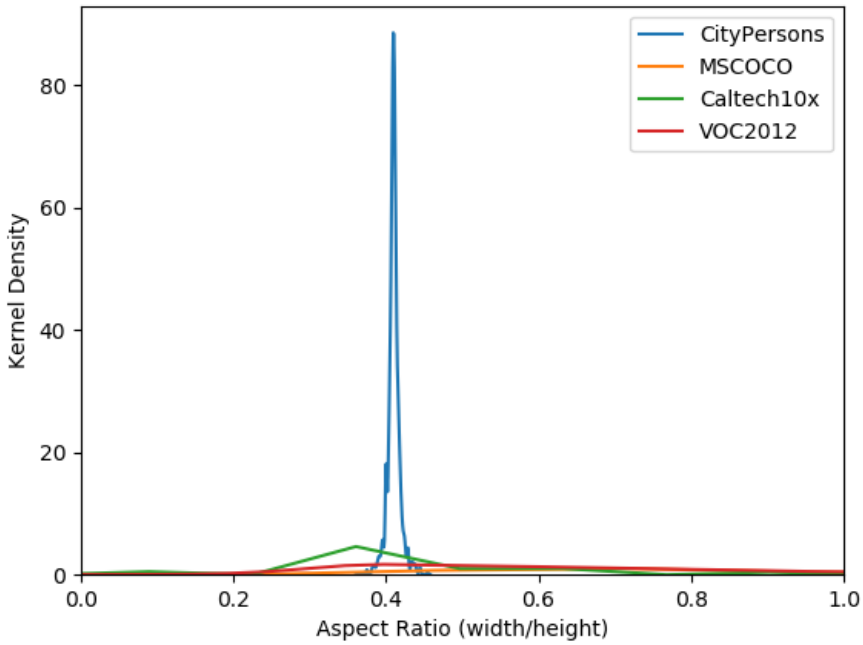


Figure 1.1: A plot of aspect ratio (*width/height*) distribution of "pedestrians" in 4 different datasets –2 general-category (MSCOCO and Pascal VOC) and 2 pedestrian-specific (CityPersons and Caltech10x). It can be seen that while there is a clear peak in the distribution for pedestrian datasets, the distribution for general-category datasets is relatively uniform.

localization problem involving $C + 1$ classes, where C refers to different classes of objects (*an addition of* 1 *is made to reflect the background class, that is image regions which correspond to no object of interest among* C *classes)*. In contrast, pedestrian detection is a 2 class problem. Secondly, pedestrians have a relatively rigid orientation (*e.g.- upright pose and small deformation due to walking stance*), while in *general-category* datasets like MSCOCO [99] and Pascal VOC [48], object categories exhibit a wider range of poses and deformation. This is illustrated in figure 1.1, where it can be seen that the aspect ratio distribution of bounding boxes for pedestrians bears a major similarity amongst pedestrian detection datasets and differs significantly from that in general object category datasets. Pedestrian detection techniques have evolved to put special emphasis on these characteristics (*e.g.*- techniques such as SDS-RCNN [14] and RPN-BF [177] utilize the mode of aspect ratio distribution of pedestrians to demonstrate improved performance). This trend is not observed in *general category* object detection systems.

As in any machine learning problem in computer vision, the quality of features extracted from images is of quintessential relevance in pedestrian detection. Most of the

early pedestrian detection systems [31, 50] were dependent on manually specified functions for feature extraction. For example the *histogram of oriented gradients* (HoG) feature [31] can be modelled by a nonlinear function which computes gradients, divides its input into grids and performs binning of gradients in each cell of a grid. The parameters of this function (e.g:- *number of orientations*, *number of cells* etc) has to be set manually and cannot not be learnt during the training process automatically. While these parameters can be tuned for specific datasets through thorough experimentation, it is difficult to tune them to perform well across different datasets. With the advent of deep learning in computer vision, the trend has shifted towards utilizing a *deep neural network* (DNN) for feature extraction. This is on account of the fact, that DNNs are trainable as against manually specified functions whose parameters were generally non-trainable. This facilitates the learning of task-specific features by DNNs, thereby leading to better performance by DNNs in nearly all domains of computer vision including pedestrian detection. **Following this observation, we limit ourselves to deep learning based pedestrian detection systems in this thesis**.

At this point one can be described, the essential basic differences between the multitude of deep learning based pedestrian systems. Understanding these key differences is the key to understand the contemporary challenges in pedestrian detection followed by the outline of technical contributions presented in this thesis. We enumerate these essential differences below :

1. **Selection of DNN :** Many classes of DNNs are known today – *restricted boltzmann machines* (RBMs), *convolutional neural networks* (CNNs) and *recurrent neural networks* (RNNs) to name a few. Within each of the above classes, there are many variations available. Pedestrian detection techniques utilizing multiple such variations of DNNs are known today [14, 159, 105]. Different DNNs extract varying features dependent upon their topology and the task-at-hand. Hence, this is one of the prominent differences found amongst pedestrian detection systems.

   When processing static RGB images, CNNs are the most commonly used DNNs owing to their suitability in representing static spatial inputs. RNNs are difficult to use with images because of the absence of a unique sequential structure in images (due to their 2D structure). Even if, a convention is adopted (such as raster sequencing), another problem related to the memory problem in RNNs [132]. Processing a large image by modelling it as a sequence of pixels, results in a long RNN, which makes learning difficult due to the vanishing gradient problem [132]. Natural images are highly varying in appearance and this makes it difficult for RBMs to model the joint probability density of images. Due to these reasons, in this thesis, we exclusively work with CNNs for pedestrian detection.

2. **Training and Fine-tuning Strategy :** The training of DNNs determines the nature of features extracted by them. This training is an optimization task guided by an objective function, more commonly known as a *loss function*. This is one of the most significant ways in which different pedestrian detection techniques differ, thereby being one of the most important technical differences amongst pedestrian detection systems.

   A previously trained DNN, can be retrained on a different dataset with a different loss function, taking its previously trained (or *pre-trained*) state as the initialization point for optimization. This process is known as fine-tuning the DNN. Fine-tuning is known to lead to faster training and better results [136, 149] and hence is widely practised in deep learning. For fine-tuning, various strategies are significant such as selection of the pre-trained state (i.e *dataset on which pre-training was done*) [179, 85] and fine-tuning only specific layers while leaving other layers fixed to the pre-trained state [76, 87]. Thus, when training or fine-tuning a DNN for pedestrian detection, the specifics of training and fine-tuning strategies play a crucial role.

3. **Feature Utilization :** Features are extracted by DNNs for an entire image. For the purpose of localization, features corresponding to different image regions are then utilized. This is known as a *feature handling mechanism*. This is one of the most important technical differences amongst pedestrian detection systems, since this mechanism has a direct impact on the quality of final detections.

4. **Post-Processing Approaches :** The detections obtained from a DNN usually encapsulate multiple detections of the same pedestrian and have to be post-processed to eliminate duplicate detections. There are many classes of post-processing techniques such as *non-maximal-suppression* (NMS) and depending upon their usage, the quality of detections can vary, thereby forming it as an important point of difference amongst pedestrian detection systems.

Having outlined a global outlook of the technical subtleties of pedestrian detection as a machine learning problem, we now embark upon a description of the technical challenges in pedestrian detection. However, we take a brief detour and settle the choice of the nomenclature “pedestrian”.

1.3 A Linguistic Clarification of the term “Pedestrian”

The exact nature of the term “pedestrian” in our work needs some elaboration. This is owing to the following reason.

Annotation protocols followed for creating different public datasets for pedestrian detection are different. For instance, in the caltech pedestrian dataset [33] and the BDD100K dataset [174], bounding boxes around instances such as people riding a bicycle are annotated as pedestrians. On the other hand, the citypersons dataset [179], there are separate annotations for person and rider. In most applications, such as *autonomous driving*, all humans need to be detected. Thus, the exact interpretation of "pedestrians" is dataset-specific.

In this thesis, we consider all annotations of humans as "pedestrians". This assumption frees us from fine-grained specifics of different annotation protocols, while also ensuring focus on detecting humans.

1.4 Challenges in Deep Learning based Pedestrian Detection

Pedestrian detection being a specific case of general-category object detection does not make it an easy problem. However, What test cases and instances make it difficult and why ? In this section, we highlight the challenges posed by pedestrian detection first, followed by linking those challenges to deep learning techniques.

Scale : Detection of small-scale pedestrians is a major challenge; scale referring to the height (**H**) of a pedestrian. For instance, in the complete caltch pedestrian dataset [33], the detection quality for pedestrians with $H < 80$ pixels is $\sim 40\%$ lower than that for $H > 80$ pixels [183]. Pedestrian appearance varies significantly between large and small scales. For instance, small-scale pedestrians are mainly delineated by their contour, while large-scale pedestrians have more elaborate appearance with facial and body details. As a result, it is difficult to distinguish small-scale pedestrians with background clutter (*e.g:- trees*). Technically, the challenge is in learning features which decompose small-scale pedestrians and background entities into 2 distinct clusters in the feature space; despite their similar appearance in the image space. CNNs employ spatial pooling between a subset of layers, thereby increasing the feature stride down the network. As a result, the information of small-scale pedestrians gets restricted to a very small set of feature map pixels. In contemporary detectors [130, 102], feature map regions are represented by a fixed length feature vector for classification and regression. The construction of this vector takes place through non-linear operations such as ROI-pooling [130] or operations resulting in interpolation errors [23]. This leads to a less reliable feature representing small-scale entities. In deep learning, scale has received significant attention in both general-category [142] and pedestrian-specific techniques [183, 16] with an objective to build scale-invariant systems. However despite several approaches such as *upsampling of input images*, *use of à trous convolutions* and *training of scale-specific classifiers* and a more

recently proposed method of scale normalization [142], there stands a significant gap to be bridged.

Occlusion : A highly occluded pedestrian is difficult to detect. As in the case of scale, the detection quality in the complete caltech dataset degrades with increasing occlusion levels [183]. Depending upon the entity occluding a pedestrian and the contemporary pose of the pedestrian, occlusion can arise in a wide range of configurations. Some datasets such as caltech [33] and citypersons [179] come with with groundtruth bounding box annotations encompassing both visible and full-body regions of pedestrians. However, especially in the case of higher occlusion levels, the visible bounding box can be particularly small and results in the same challenges as scale. Given a feature representing a region containing a pedestrian, it is interesting to reliably decompose it into two separate feature representations – one representing the visible pedestrian part and the other representing the region occluding the pedestrian. This approach, often known as feature disentanglement [155] is recent and popular, but has been applied to a select few problems such as face recognition and pose estimation; both of which are non-localization problems.

Illumination variations and other Environmental Factors : Under low illumination conditions, detecting pedestrians is particularly difficult. Low illumination conditions rob off most of the information about edges and other basic low-level features from an image. These low-level features are usually detected by the early CNN layers [86, 176], and form the basis for feature decomposition by later layers. The lack of these features under low illumination, changes the statistical properties of the input to a CNN. In addition, under low-level illumination, other features such as image-noise become more prominent, which further cause degradation of performance. Most existing public datasets are limited to daytime environment, where illumination variations are relatively milder. Lack of any corresponding training data makes it difficult to train or tune a system to low illumination conditions. Other environmental factors such as weather, in a similar way cause degradation of performance.

Cross dataset generalization : Generalization performance of a machine learning system is a well known and fundamental problem. Generalization performance analysis of a machine learning system involves studying its test-time performance on various datasets, which can be potentially different from the dataset(s) used for training it. The source of this issue of cross-dataset generalization lies in the distribution function modelling a dataset. A machine learning problem is computing a mapping $\mathcal{F} : \mathcal{X} \longrightarrow \mathcal{Y}$, where the domain $\mathcal{X}$ represents the input data and the co-domain $\mathcal{Y}$ represents the target (*classification*

or regression). Each $x \in \mathcal{X}$ can be thought to be a random variable generated by a probability distribution function $\mathcal{P}$. During training pre-specified $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are used to model $\mathcal{F}$. For different datasets, $\mathcal{P}$ could be different, thereby essentially changing the domain of $\mathcal{F}$. Thus $\mathcal{F}$ modeled during training might not fit a dataset with a different $\mathcal{P}$. This phenomenon is also known as dataset bias [154]. The most common approach to mitigate this phenomenon is to map the data to a different space $\mathcal{R}$ modeled by a distribution function $\mathcal{P}_{\mathcal{R}}$, such that $\mathcal{P}$ and $\mathcal{P}_{\mathcal{R}}$ generate independent random variables. Deep learning has proved to be more successful than handcrafted feature-based techniques in ensuring better cross-dataset generalization; though far from perfect. Fine-tuning a deep learning based pedestrian detection model on a new dataset (*from the same problem domain*) is often found to have better performance than the original model. However, fine-tuning is an impractical technique when taken to real-world applications such as *autonomuos driving*, where a system has to respond favorably in potentially unseen pedestrian instances. These unseen instances may differ in scale, appearance or weather conditions as described before. This makes it pertinent to analyze pedestrian detection systems from the perspective of cross-dataset performance.

1.5 Contributions of the Thesis

First we give a broad outline of the problem areas we have addressed in this thesis. We then follow-up with a summary of different chapters in this thesis in terms of their content and contributions.

To facilitate the outline of our focus areas, we refer to figure 1.2 showing the general pipeline of a deep learning based object or pedestrian detection system. The input image is first processed by a CNN to obtain a feature map. This is followed by a "*region selection strategy*" which selects a set of locations in the feature map for further processing. Some detection systems process all possible locations in the feature map. In general, the region selection strategy can be encapsulated as a transformation which takes as input a feature map and outputs a set of locations or regions inside the feature map which are processed further. More details on this can be found in chapter 2. Given a location or region inside a feature map, it is represented by a feature vector. The exact representation of a region using a feature vector varies from one method to other and the corresponding mechanism is known as the "*feature handling mechanism*" (described in detail in chapter 2). The feature vector representing a region is then *classified* and *regressed*. If classified as a pedestrian, the bounding box obtained from the regression represents the bounding box coordinate of the pedestrian.

From the above description emerges 3 important perspectives which we address in this thesis.

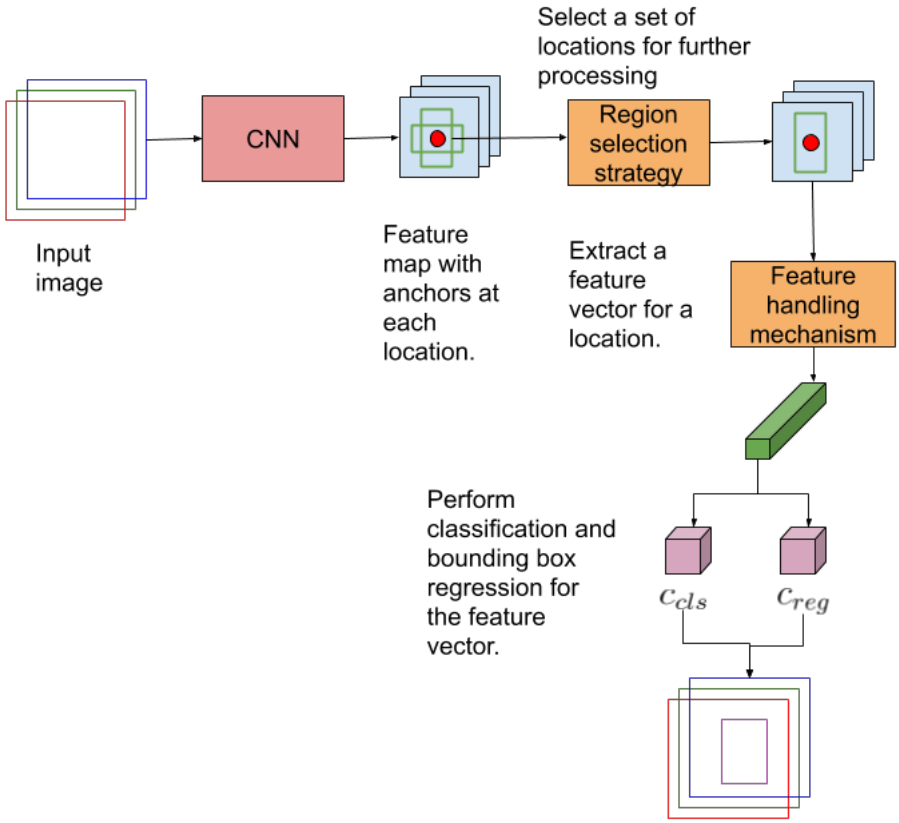


Figure 1.2: General pipeline of an anchor based object or pedestrian detection system.

1. **Training and Fine-tuning strategy :** Training and fine-tuning of deep CNNs is a non-trivial task involving non-linear optimization which is heavily influenced by the selection of various hyperparameters and optimization techniques. In addition, various pedestrian detection techniques share some common elements such as *anchors* which invoke additional hyperaparameters. Another interesting observation is the selection and ordering of datasets used for training and fine-tuning detection systems. For instance, the paper proposing the CityPersons (CP) dataset [179] proposes its use for initial training followed by fine-tuning on the caltech dataset [33] for better results. With different datasets varying in their resolution and other characteristics such as distribution of *scales* and *occlusion* of pedestrians, it is of interest to explore if there is a general scheme of ordering which can be followed for achieving better results. In addition, through existing works, the impact of all the aforementioned factors on detection across *scales* and *occlusion levels* is not clear.

   This perspective hence, aims to set a basis for a more informed training and fine-tuning of pedestrian detection systems. We take this perspective to conduct extensive analysis of various pedestrian detection techniques for various pedestrian datasets, leading to interesting insights into the impact of aforementioned factors on pedestrian detection. In addition, these insights set the foundations for understanding the prominent research directions in pedestrian detection.

2. **Feature Representation :** During the training of CNNs, the computed gradients determine the updates to the network weights. These gradients being computed; starting from the loss value, using the chain rule of derivatives; are hence impacted by the network structure. During testing, the learnt network weights determine the features used for detection. Thus, the network structure is fundamental to the quality of features; a phenomenon which differentiates various techniques from one another and motivating us to explore ways to improve feature representation by making better usage of CNNs. Some ways to improve feature representation explored in this thesis include – *early and late fusion of features from multiple CNN layers*, *usage of visible and full-body bounding box annotations* and *use of semantic segmentation*.

3. **Minimizing False Positives by Feature Selection :** Pedestrians occupy a small portion of the image space compared to non-pedestrian entities. This leads to a class imbalance problem in pedestrian detection leading to lower true positives. Setting higher weights to the minority class usually improves true positive rate, but at the expense of increased false positives [163]. To mitigate this problem and to select the most relevant features, traditionally feature selection techniques such as bootstrapping (*often called hard negative mining*) [143, 50] are employed. This approach has dominated deep learning based methods as well [139, 150, 137], but is not without

problems. For instance, [137] shows that *foreground* to *background* ratio of $1:3$ is a rather precarious ratio for most object detectors and deviating from it can cause performance drop by as much as 3 mAP points.

This makes feature selection an important problem in pedestrian detection. We explore this perspective in this thesis by eliminating potential background regions early in the detection pipeline. Traditionally, this approach forms the motivation for boosting based methods. We however take a simpler approach using semantic segmentation, which impressively eliminates the majority of background samples, and in addition offers an increased inference speed.

The following contributions are presented as a part of this thesis.

1. We conduct an extensive analysis of the contemporary pedestrian detectors over 3 large-scale publicly available pedestrian datasets – caltech pedestrians, citypersons and BDD100K. This analysis considers the performance of contemporary detectors for a wide range of driving environments which include *weather* and *illumination variations*. Our analysis sheds a detailed light on the limitations of contemporary pedestrian detectors. In addition, we report extensive experiments with varying configurations of existing pedestrian detection frameworks. These experiments and resulting analysis further provide a set of guidelines towards designing novel pedestrian detection frameworks. In connection with section 1.2, this analysis sheds light on two major aspects namely, *selection of DNN* and *training and fine-tuning strategy*.

2. We propose "Multilayer-RPN", a pedestrian detection framework – which explicitly and simultaneously utilizes multiple convolutional layers to perform pedestrian detection. We propose two strategies to utlize multiple convolutional layers, *early fusion* and *late fusion*. Our experiments show that early fusion is a better coalescing strategy for utilizing multiple convolutional layers than late fusion. We furthermore show the effectiveness of individual convolutional layers in detecting pedestrians of varying scales and occlusion levels. This contribution in particular, provides quantitative estimates of the effectiveness of individual layers in detecting pedestrians of varying scales and aspect ratios. This quantitative analysis therefore provides a very useful prior knowledge when designing a pedestrian detection system with the objective of scale and occlusin invariance.

3. We propose a "semantic segmentation based" pedestrian detection system, which performs early fusion of multiple convolutional layers followed by semantic segmentation of pedestrian locations to aid pedestrian detection. Our proposed approach selects a small set of relevant image locations for processing and is more effective than the region proposal stage of Faster-RCNN [130]. As a result, our proposed

approach performs fewer computations than other pedestrian detectors and is able to introduce significant improvements in detection rates while not sacrificing detection speed. This proposed approach achieves state-of-art performance on the caltech and citypersons dataset while operating at a detection speed of nearly 20 frames per second.

4. We propose an improvement over the "semantic segmentation based" pedestrian detector, by further reducing the number of computations while improving both detection speed and detection accuracy. We introduce the concept of anchor selection layer which uses anchor specific kernels to select a small set of anchors which overlap well with both visible and full body parts of a pedestrian. Owing to the anchor selection layer, the number of image regions to be processed is reduced significantly and this introduces a nearly 2 times improvement over our previous contribution while still achieving state-of-art performance on the caltech and citypersons dataset.

The outline of chapters in our thesis is as given below.

Chapter 2: We discuss the existing literature on deep learning based pedestrian detection along with a synopsis of their technical summary.

Chapter 3: We present a detailed experimental analysis of various pedestrian detection techniques with various network structures, loss functions and hyperparameters. We gain useful insights from this – particularly the relative impact of various factors pertaining to deep neural networks on the quality of pedestrian detection across scales and occlusion levels.

Chapter 4: We propose and analyze a pedestrian detection system, which utilizes multiple convolutional layers simultaneously and explicity to perform pedestrian detection. In particular we study the role of feature fusion (early vs. late) on the quality and complexity of detection. In addition we gain useful quantitative insights into the role of individual layers in detecting pedestrians across scales and occlusion levels.

Chapter 5: We propose a semantic segmentation based approach to feature selection for pedesetrian detection. We conduct detailed experiments and analysis, thereby exhibiting the performance of the proposed approach on 3 public benchmark datasets – caltech, citypersons and BDD100K. Our findings support our claim that the proposed approach reduces the false positive detections. We also present runtime performance of the proposed approach which shows that the proposed approach has a high inference speed, which is an additional advantage gained by it.

Chapter 6: We propose a new anchor selection layer, which improves upon our prposed approach in *chapter 5*. In particular we show improved detection accuracy and speed, thereby making the proposed method a strong step towards achieving realtime performance in pedestrian detection.

Chapter 2

Pedestrian Detection Approaches in Deep Learning

Contents

2.1 Introduction

Pedestrian detection techniques are either based on handcrafted features [117, 168, 31, 124, 180, 88, 6] or deep learning based features [14, 177, 152, 17, 5, 100]. Over the recent years, deep learning approaches have maintained their superiority over handcrafted approaches. As stated in section 1.2, in view of this, we focus upon deep learning based approaches in this chapter and the rest of the thesis. In this chapter, our particular emphasis is on creating a taxonomy of pedestrian detection techniques. A taxonomy is useful in structuring the repertoire of pedestrian detection techniques to ensure their easy analysis and description. The major highlight of this chapter is in creating a firm technical base for organizing the technical descriptions presented in subsequent chapters. We begin this chapter by noting that nearly all contemporary deep learning based pedestrian detection techniques are derived from *general-category* object detectors. Hence, the mechanisms followed in different *general-category* object detectors have an impact on the design and operation of various pedestrian detectors. With this viewpoint, we first categorize *general-category* object detectors followed by describing them. We then describe various pedestrian detectors as extensions and modifications of *general-category* object detectors.

2.2 Detection Systems – Single-Stage and Two-Stage

The term "**stage**" in object detection literature refers to a portion of an object detection pipeline, such that it produces bounding boxes as output. All object detectors are *single-stage* [102, 127] or *two-stage* [130]. Figure 2.1 outlines the major characteristics of one-stage and two-stage object detectors.

Technically these two classes of detectors differ from each other in following two perspectives.

Object Proposal Extraction : An object proposal is a set of bounding boxes which represent the *regions of interest* (ROIs) which might be potential target objects. The development of object proposal techniques are motivated by the computational complexity of traditional sliding window-based approaches to object detection. In the lack of proposal detection schemes, all possible regions tiled by a sliding window need to be processed for determining if an object spans them. Since usually objects of interest span a limited portion of the image, exhaustive scanning is computationally inefficient. Object proposal techniques serve the purpose of reducing the search space for detection of objects, thereby enabling faster object detection.

Object proposal detection schemes have been proposed for quite some time, even before the advent of deep learning [185, 157, 3, 24, 123, 107]. Early object proposal de-

tection schemes were based on handcrafted features [185, 157, 3]. These were taken up by the first deep learning based object detection systems such as Fast-RCNN [55] (*which used selective search* [157] as the object proposal detection method. This however required a two-step training process – i) *training the selective search* and, ii) *training the Fast-RCNN detector*. Selective search is a non-deep learning based approach. Therefore, usage of selective search cannot be used for an end-to-end learning pipeline. Moreover, selective search can be slow, thereby affecting the inference speed in speed-critical applications like self-driving cars. As part of Faster-RCNN [130], a deep learning based object proposal scheme known as *region proposal network* (RPN) was proposed. Faster-RCNN couples RPN with RCNN (coming from Fast-RCNN) and performs an end-to-end training for jointly learning object proposals and final bounding box detections. RPN has since then been consistently used in all two-stage pedestrian detection approaches based on Faster-RCNN, such as RPN-BF [177], MSCNN [16], SDS-RCNN [14], Multiple-RPN [159] and many others.

The use of object proposal detection schemes is one of the most fundamental differences between two-stage and one-stage detection systems. One-stage detectors as against two-stage detectors do not utilize object proposal detection. Object proposal detection is the focus of the first stage of all two-stage detectors. The second stage of two-stage detectors then conducts classification and bounding box regression over the proposals from the first-stage to determine the final detections. In two-stage detectors, the second stage utilizes another concept of "feature pooling" which is not followed in one-stage detectors and is described in the next paragraph.

Feature Pooling : Feature pooling refers to isolating a portion of a feature map and conducting subsequent operations on it. On an implementation level, this involves making a copy of a portion of a feature map by first allocating a portion of memory for storing the values to be copied followed by the actual operation of copying the values at the allocated memory location. Thus large number of feature pooling operations is a slow operation due to large number of copy operations. Feature pooling is also limited by the available GPU memory (when conducted on GPU).

So, why is featuer pooling necessary ? The object proposals from the first-stage of two-stage detectors need to process each proposal region for final detections. The proposal regions can be of varying scales and aspect ratios. A classifier such as fully connected layer or even a support vector machine (SVM) requires a feature vector of fixed length for classification. Therefore, it becomes necessary to resize each proposal region to extract a fixed length feature vector for further processing. Since, resizing is a lossy operation (owing to interpolation artifacts), it is necessary to make a copy of each proposal region in a feature map.

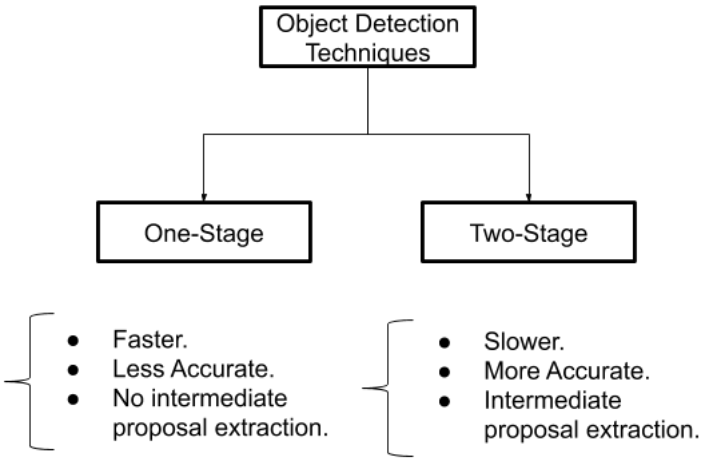


Figure 2.1: Major characteristics of one-stage and two-stage object detectors.

Feature pooling is not employed in one-stage detectors. Instead, one-stage detectors, process all feature map locations, by considering the feature vector at any feature map location to be representative of all anchors centered at that location. For a more detailed explanation of classification and bounding box regression in one-stage detectors, we refer the reader to section 2.2.2.

We now turn to our description of the detailed working of one-stage and two-stage detectors in deep learning. Two-stage detectors are based on Faster-RCNN [130] and one-stage detectors are based on SSD [102]. Thus, we highlight the working of one-stage and two-stage detectors with Faster-RCNN and SSD as template examples.

2.2.1 Two-stage detectors

A two-stage detector computes the final detections in the following sequential steps – a) object proposal extraction and, b) refining the object proposals to obtain the final detections.

All the two-stage detectors are derived from Faster-RCNN. Hence, they share a common mechanism for handling feature maps. This mechanism results in high detection accuracy alongwith lower detection speed. We exhibit this mechanism in figure 2.2. As shown in figure 2.2, first, a minibatch $\mathcal{B}$ of images is fed to a CNN (*known as base network or base head*). The resulting feature map may be processed by a set $\mathcal{C}_{addn}$ of additional convolutional layers to obtain a feature map $f_{\mathcal{B}}$. A set of hypothetical bounding boxes (*anchors*) are tiled over $f_{\mathcal{B}}$. Anchors serve the same purpose as sliding windows [50]. Usually multiple anchors of varying scales and aspect ratios are centered at each location in $f_{\mathcal{B}}$. Each location $\mathbf{P}$ in $f_{\mathcal{B}}$ encodes a vector v, whose length is determined by the specifics of the network architecture. Two parallel branches of 1×1 convolutional layers (c_{cls} and c_{reg})

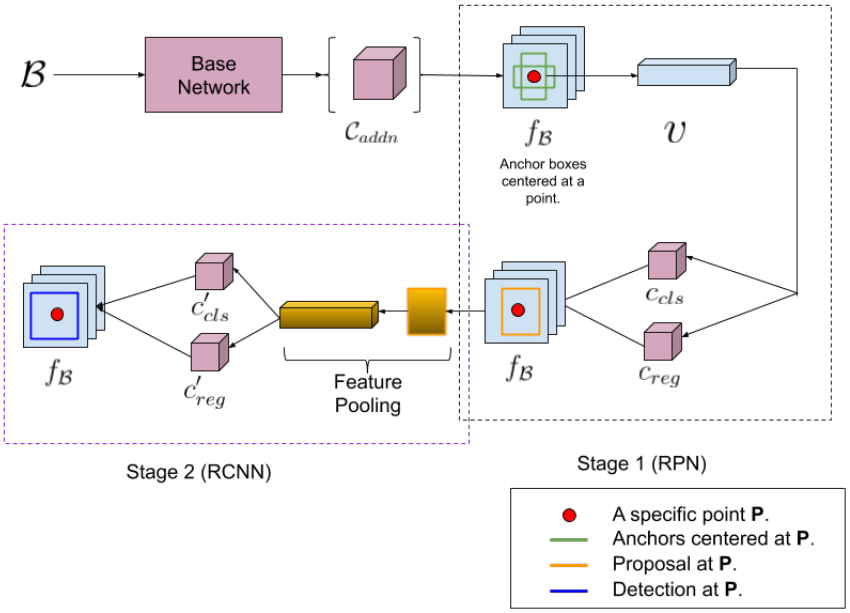
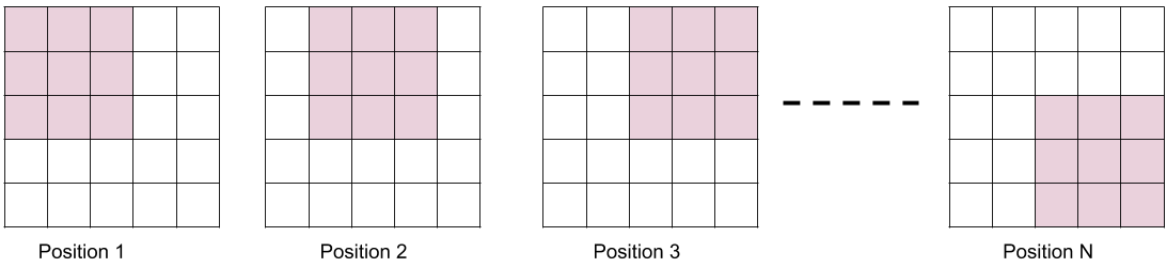


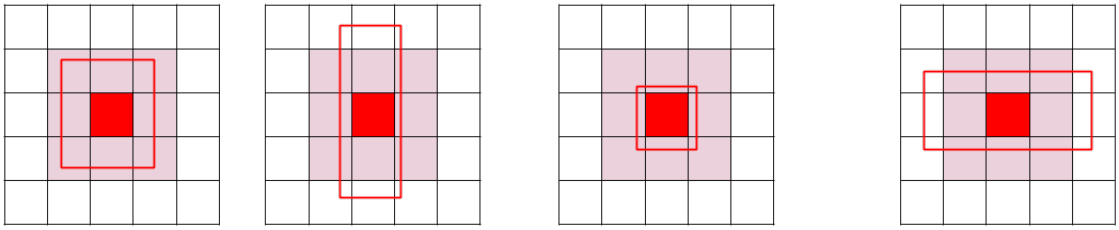
Figure 2.2: Feature handling mechanism of Faster-RCNN and other two-stage detectors.

perform a 2-class classification and bounding box regression respectively for v. The two classes correspond to *object* and *non-object* respectively. During training, the *intersection-over-union* (IoU) between an anchor and groundtruth bounding box determines the target class of the anchor. This 2-class classification and bounding box regression results in a set of object proposals. This pipeline from generating $f_{\mathcal{B}}$ to obtaining object proposals is known as the *region proposal network* (RPN).

Quite importantly, in this first stage all confocal anchors share the same feature vector for proposal classification. Figure 2.3 illustrates this situation. Figure 2.3 (**top**) illustrates different sliding positions of a convolutional kernel sliding over a feature map. In figure 2.3 (**bottom**), the overlap of the kernel with the anchors for a specific location in the feature map is illustrated. From the figure 2.3 (**bottom**), although, all 4 confocal anchors are of different scales and aspect ratios, them sharing the same feature vector is an inaccurate feature handling situation. This feature handling situation is inaccurate because confocal anchors have different shapes and hence varying receptive fields thereby encoding varying information about the input image. Thus, the assumption of one feature vector representing multiple confocal anchors is an imprecise modelling of the information contained in them. In the case of a large discrepancy between the shape of an anchor and that of the filter kernel such as that in the 2^{nd} and 4^{th} columns of figure 2.3 (**bottom**), the output feature vector represents information which significantly differs from the information inside the anchors. In spite of this, this approach for proposal detection is preferred on account of its high speed. On parallel computing devices such as GPUs, the convolutions can be



Different positions of a convolutional filter kernel as it slides over a feature map. For simplicity, here the feature map is of shape 5x5 and the filter kernel is of shape 3x3.



Overlap of a filter kernel with 4 confocal anchors centered at a feature map location indicated in red. The same feature vector (result of convolution at the focus of anchors) represents all 4 confocal anchors.

Figure 2.3: (**Top**):- Different positions of a convolutional kernel sliding over a feature map. (**Bottom**):- The location of a filter kernel for 4 confocal anchors. The dark red pixel is a specific location on the feature map. The pink region depicts the convolutional kernel. The unfilled dark red rectangles are the anchors. Since for each of the 4 confocal anchors, the location of the filter kernel is fixed, the convolutional output value is the same and for all 4 confocal anchors.

computed in parallel across many sliding locations of a convolutional kernel. The heavy work of more accurate classification and bounding box regression of resulting proposals is deferred to the second stage as described below.

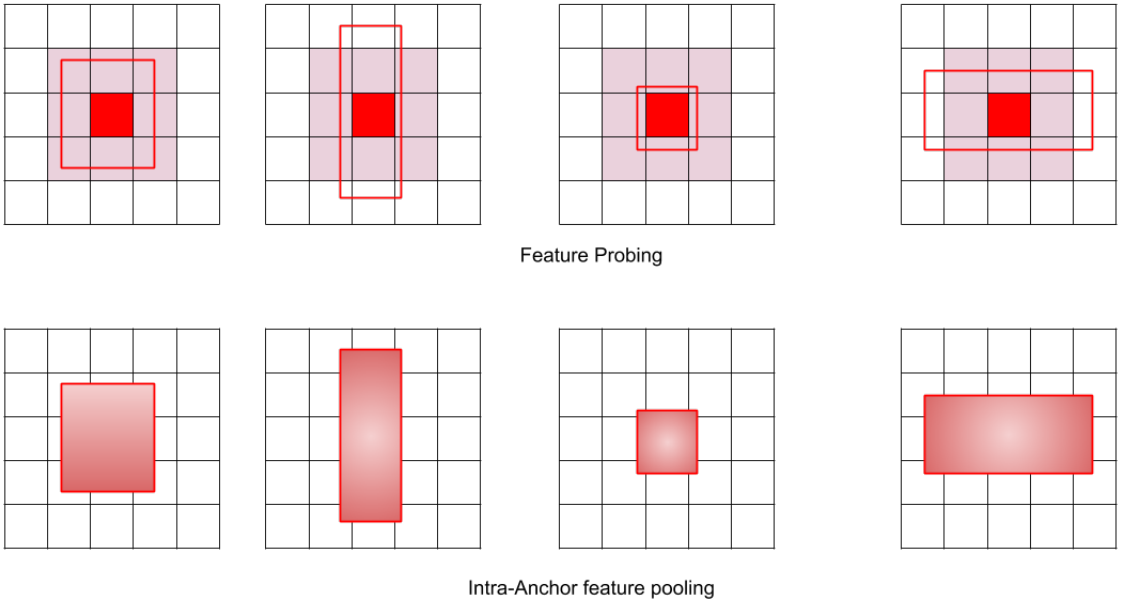


Figure 2.4: Intra anchor feature pooling vis-à-vis feature probing. **Top:** Feature probing at a feature map location indicated by a *dark red* pixel. The pink region shows a 3×3 kernel at the location, while the unfilled dark red rectangles are the anchors centered at that location. Each of those anchors share the same feature vector $\boldsymbol{v}$. $\boldsymbol{v}$ is the result of convolution of the kernel with the feature map at the feature map location. **Bottom:** Intra-anchor feature pooling for the same scenario as in **Top**. Each anchor has a different feature vector, which is obtained by extracting features from inside the anchor and then using ROI-Pooling [130], ROI-Align [63] or resizing [23] followed by flattening. It is evident that intra-anchor feature pooling produces more anchor-specific features.

In the second stage (RCNN), features from inside the proposals are extracted and transformed into a fixed length vector, subsequently used for classification and bounding box regression. This operation is known as *intra-anchor feature pooling*. Compared to the operation of feature vector creation in the first stage (*we call it feature probing*), it is more effective as the features are extracted from inside the anchors, thereby collecting the information specific to each anchor. In this case, the multiple proposals/anchors centered at the same location do not share the same feature vector. Figure 2.4 exhibits the difference between feature probing and feature pooling. There are many approaches to perform *intra-anchor feature pooling* such as ROI pooling [130], ROI align [63] and crop_and_align [23]. In the original Faster-RCNN work [130], ROI-Pooling is used to achieve this transformation. Given a feature map and a rectangular sub-region corresponding to a proposal or anchor, ROI pooling folows the steps as outlined in figure 2.5(**top**). First, the coordinates

of the subregion are approximated by rounding them off to the nearest integer. Then, assuming that a $n \times n$ feature map is to be obtained, the subregion is divided into $n \times n$ grids. Within each grid, max-pooling is performed to obtain the final $n \times n$ feature map which is flattened to get a feature vector. The value of n therefore is a hyperparameter which needs to be defined apriori. In Faster-RCNN [130], $n = 14$. The feature map from ROI-pooling is maxpooled by a kernel of size 2×2 with a stride of 2, to obtain the final feature vector. In Mask-RCNN [63], an improvement over ROI-pooling, called ROI-Align was proposed. The difference between ROI-pooling and ROI-Align is that, the latter does not perform any rounding off of the coordinates of the subregion. Instead, the entire subregion is approximated by bilinear interpolation, followed by the remaining operations which are the same as ROI-pooling. ROI-Align is better than ROI-pooling in that, lack of any rounding off better preserves the information inside the subregion. In a later technical report [23], a "crop and resize" operation was proposed as an alternative to ROI-pooling. This operation involves cropping the subregion features and resizing them to a fixed dimension followed by flattening. During this cropping, the non-integer coordinates of the subregion are handled in the same way as ROI-pooling, i.e by rounding them off to the nearest integer. Experiments [74, 23] have suggested that "crop and resize" results in a more stable training compared to ROI-pooling. "Crop and resize" operation is similar in performance and efficiency to ROI-align operation when working on object detection problems. In the case of applications requiring higher localization precision, such as image segmentation, ROI-Align performs better than ROI-pooling and crop_and_resize operator.

The fixed length feature vector is further classified and regressed by another set of parallel branches of 1×1 convolution (similar to RPN), to obtain the final detections.

2.2.2 One-Stage detectors

One-stage detectors bypass the object proposal stage and perform final detection by directly operating on the anchor boxes. Figure 2.6 shows the general feature handling mechanism for one-stage detectors like SSD [102] and YOLOv2 [127]. Compared to the feature handling mechanism of two-stage detectors (figure 2.2), one-stage detectors do not perform any feature pooling. In addition, for a N-class detection problem, a $N + 1$-class classification is done by c_{cls} (*+1 for background*). Bounding box regression can be performed for each class separately (N-way regression) or class agnostic. During training, as in the RPN stage, the target class for an anchor box is determined by its IoU with the groundtruth. Since no feature pooling is involved in one-stage detectors, they are faster during inference. However, lack of feature pooling also implies imperfect feature extraction as illustrated in figure 2.3, since as in RPN, v represents all the concentric anchors at a location. This often resonates in the detection performance of one-stage detectors,

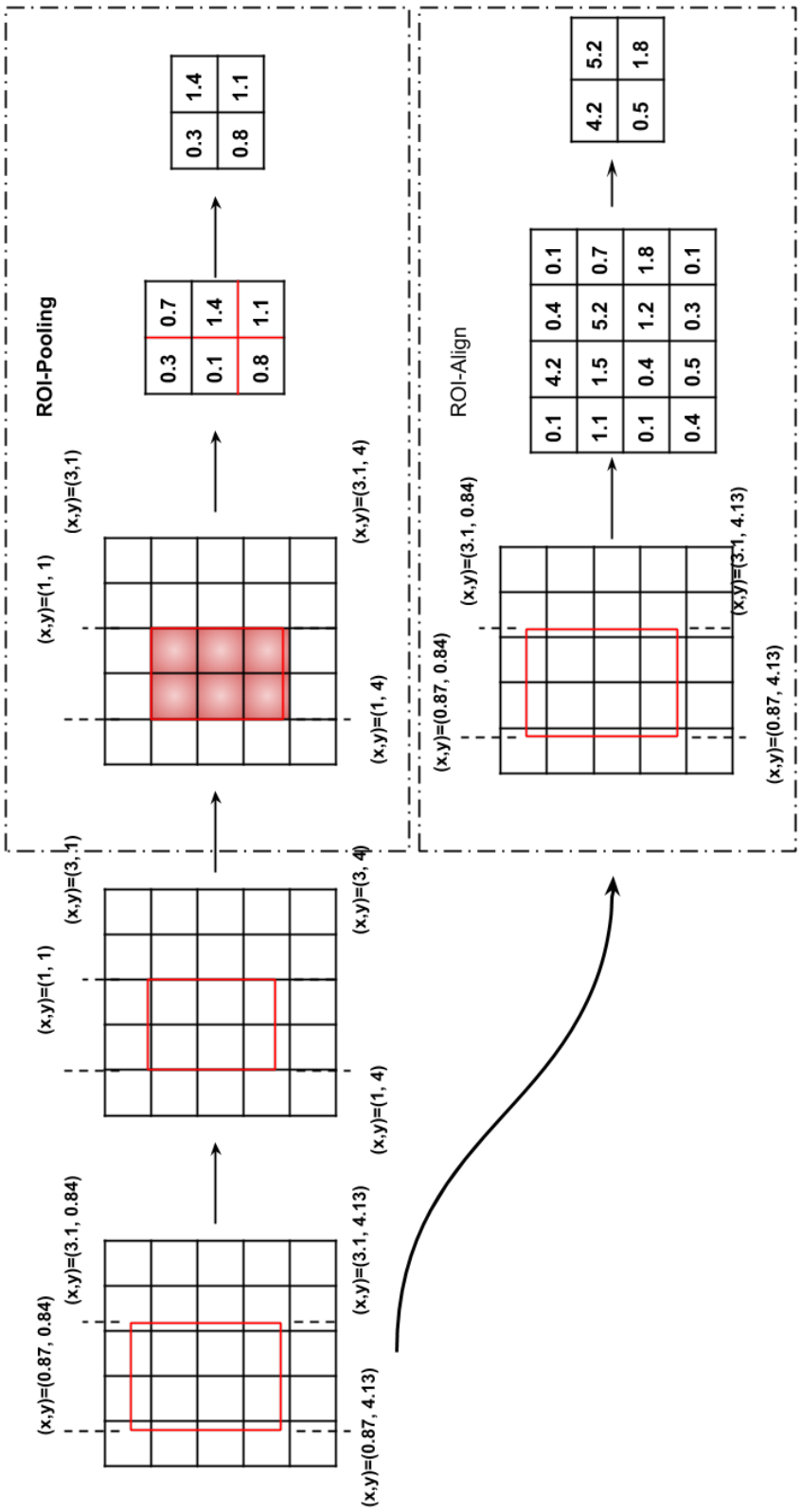


Figure 2.5: Difference between ROI-Pooling [130] and ROI-Align [63] mechanisms for intra-anchor feature pooling

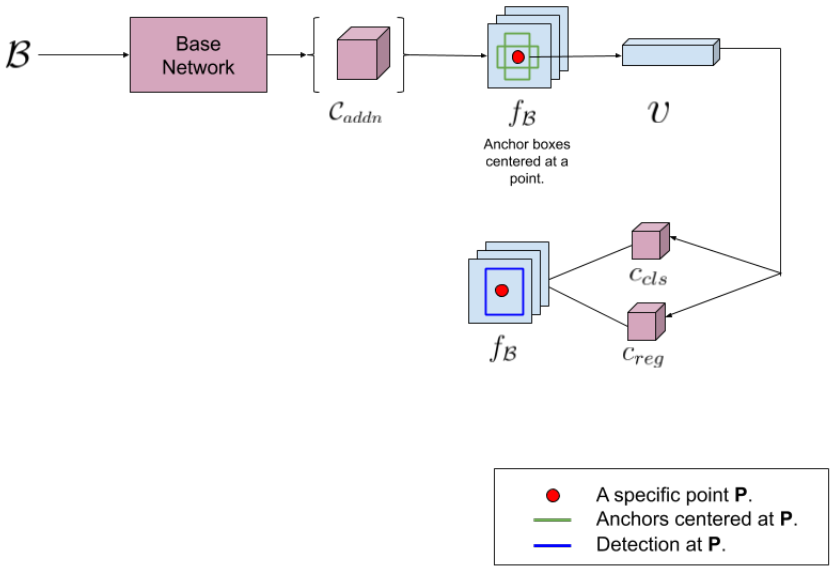


Figure 2.6: Feature handling mechanism of one-stage detectors.

which generally perform worse than two-stage detectors.

One-stage detectors such as SSD [102] and YOLO [127] compensate for the lack of high quality feature pooling by utilizing multi-scale training and testing. Features from various layers of a CNN are extracted and are processed ultimately leading to their late fusion i.e coalescing of their detections.

2.3 Pedestrian detectors as extensions of general-category detectors

Table 2.1 lists some well known pedestrian detection techniques along with a short summary of their main contributions and their performance on the caltech-reasonable testing dataset. It can be gathered from table 2.1 that with deep learning, the performance of modern pedestrian detection systems has indeed improved by a large margin. DLSP [153], though not the first deep learning based technique for pedestrian detection, was the first to report a major performance improvement on the caltech-reasonable test set. Since DLSP [153], the trend has been consistently moving towards deep learning based systems. Historically, the incorporation of deep learning techniques in pedestrian detection began with restricted boltzmann machines (RBMs) [105, 106]. The success of RBMs in learning good quality features from image pixels [68] such as in the MNIST dataset [91] was the primary source of motivation for these early works. The emphasis therefore in these early works was on improving feature representation. Localization of pedestrians was mainly

Method	Year	Miss-Rate (Caltech Reasonable) (Test Set)	Main Features
DDM [165]	2012	Not Available	a) Focus on occlusion handling. b) Inspired from DPM. c) Use of Restricted Boltzmann Machines (RBMs).
UMFL [135]	2013	Not Available	a) 2-Stage system b) Unsupervised sparse coding in stage 1. c) Supervised training in stage 2. d) Evaluation protocol different from other approaches.
JDL [118]	2013	39%	a) Inspired from DPM b) Joint learning of i) Pedestrian parts. ii) Occlusion handling. iii) Deformation handling.
DLSP [153]	2015	11.89%	a) Computationally expensive. b) Inspired from DPM. c) Training of multiple CNNs.
FRCNN [55]	2015	14%	a) Combines i) Region proposal, ii) Classification and iii)Bounding box estimation in one end-to-end network. b) Generalizable to multiple object classes.
SSD [102]	2016	11%	a) Uses a multibox loss function. b) Uses features from multiple layers. c) Merges region proposal and classification in one step.
RPN-BF [177]	2016	9.8%	a) Use of Boosted Forests for classification b) Use of caltech-specific aspect ratio for anchors.
SDS-RCNN [14]	2017	7.6%	a) Use of pseudo-segmentation mask as a prior. b) Use of a second network for bounding box refinement.
YOLO-V2 [128] (As reported in [114])	2017	20.83%	a) Use of anchors. b) Multi-scale training at 3 scales.
MSCNN [15]	2018	8.8%	a) Use of multiple layers. b) Each layer processes a specific scale of pedestrians.

Table 2.1: A brief summary of some popular deep learning approaches to pedestrian detection.

accomplished in a manner similar to sliding windows or spatial pooling [105]. Sliding windows and spatial pooling to determine localization of objects often lead to inaccurate localization, thereby causing ensemble metrics such as *mean-averaged-precision* (mAP) or *log-averaged-miss-rate* (LAMR) to be compromised. These early works highlighted the need for new detectors for joint classification and bounding box estimation.

All pedestrian detectors in use today, are built on convolutional neural networks. These contemporary pedestrian detectors stem from the aforementioned feature mechanisms of either *one-stage* or *two-stage* object detectors. These contemporary detectors extend the ideas of *general-category* object detectors such as Faster-RCNN, SSD or YOLO and specialize them for *pedestrian detection* using one or more refinements such as *loss function*, *network architecture* and *multi-task learning*. The key to understanding the evolution of modern pedestrian detectors is to understand the relative merits of their individual refinements. We hence take the position of describing contemporary pedestrian detectors by their individual refinements. Table 2.2 summarizes some principal pedestrian detection systems with their corresponding enhancements over their *general-category* counterparts.

Method	Number of stages	Primary Contribution Area
RPN-BF [177]	2	Classifier
Multiple-RPN [159]	2	Classifier + Late Fusion
ALF-Net [103]	1	Late Fusion + Focal Loss
MSCNN [16]	2	Late Fusion + Surrounding information
Rep-Loss [166]	2	Repulsion Loss
SDS-RCNN [14]	2	Pseudo semantic segmentation
M-Net [108]	2	Early Fusion

Table 2.2: Enhancements introduced by some contemporary pedestrian detection systems.

2.3.1 Architectural Refinements

The architecture of a CNN refers to the topology of its layers including the hyperparameters specifying each layer. The fundamental purpose of a CNN architecture is to perform feature extraction. During training, a loss function (otherwise also known as *objective function*) guides the weights of the CNN architecture to suit feature extraction for a specific task such as *image classification*. A large body of work exists on CNN architectures and their theoretical analysis [86, 140, 67, 147, 73, 71, 25, 19, 101, 148, 104, 144]. However much of the theoretical analysis of CNN architectures is still in its infancy and does not offer a direct insight into practical matters such as *selection of hyperparameters* and *specification of depth of a network to achieve optimal performance*. As a result most of the insights about the utility of various CNN architectures for specific applications are based on experiments. Generally, rather than designing a new CNN architecture from grounds up, most methods focus on harnessing the power of existing CNN architectures. One of the most common motivations in this regard is to introduce *scale* and *occlusion*-invariant detection.

CNNs being hierarchical and cascaded systems interspersed with pooling layers, causes the resolution of feature maps to decrease as the layers are traversed. Given the input image size ($S_{I_{in}} \times S_{I_{in}}$) and output feature map size ($S_O \times S_O$), the network stride (S_N) is defined as :

$$S_N \triangleq \sqrt{\frac{S_O}{S_{I_{in}}}} \tag{2.1}$$

S_N can be substantial such as 16 in VGG16 [140]. In the inception class of networks [147], S_N is even more drastic (~ 37), if no global average pooling is done and 299, if global average pooling is performed. Let us assume that $S_N = \alpha$. Therefore every pixel in the output feature map covers a spatial area of size $\alpha \times \alpha$ in the input image. Thus, the output feature map does not encode discriminative information about any pedestrian covered by a spatial area smaller than $\alpha \times \alpha$. S_N can be computed for any convolutional layer by considering it to be the output layer. Generally, if there are P pooling layers in a

network, there exist $P + 1$ different feature strides in a network.

To overcome the limitations imposed by S_N, the most common architectural refinement followed by most pedestrian detection approaches for *scale-invariant* detection is to utilize multiple CNN layers [103, 159, 16, 94]. Apart from overcoming the limitations imposed by S_N, use of multiple layers also allows for greater feature diversity. ALF-Net [103], Multiple-RPN [159], SAF-RCNN [94] and MS-CNN [16] are some prominent techniques utilizing architectural refinements motivated by the above idea. Use of multiple layers has been well known in *general-category* object detection for quite some time such as *feature pyramid networks* (FPN) [44], RetinaNet [97], Hypernet [83] and *inside outside net* (IONet) [8]. The use of multiple layers was primarily motivated by the success of image pyramid techniques in object detection such as *deformable parts model* (DPM) [50]. Spatial pyramid pooling [65] was the first deep learning based object detection technique incorporating the use of image pyramids. The difference between earlier approaches such as spatial pyramid pooling [65] and more recent ones such as [44, 159, 103, 16] is that unlike [65] which feeds a spatial pyramid of images to a network, the new approaches feed a single spatial resolution image and consider feature maps generated by various layers as pyramid. These recent approaches have both computational and performance superiority over directly using an image pyramid as input. Efficient processing of image pyramid requires multiple passes through a CNN and is hence computationally less efficient during training and testing. In addition, it has been known through experiments such as those conducted in [58] and [169] that CNN features are not invariant over large variations in scale. Therefore, processing of images of different sizes (*due to pyramidal structure*) is bound to result in features which are hardly optimal for objects of all possible sizes due to limited scale-invariance of CNNs. While, this could be answered by the use of multiple different CNNs, it is still inefficient due to a larger memory footprint required in such a case.

There are generally 2 approaches to utilize multiple CNN layers – *early fusion* and *late fusion*. *Late fusion* is more common and refers to coalescing the detection outputs of multiple CNN layers to generate final detections [159, 44, 103, 16]. In contrast, *early fusion* is where a single detection output is obtained by combining multiple layers' features early on in the pipeline. Pedestrian detection techniques utilizing early fusion include HyperLearner [108] and MCF-Net [20]. Generally late fusion approaches are slower and less accurate compared to early fusion approaches. Late fusion implies multiple detection computations and generally it means more computations. Due to large number of detections generated by late fusion approaches (due to detections available at multiple feature layers), a rather more aggressive application of post-processing techniques such as *non-maximal suppression* is required which often leads to lower detection accuracy.

In chapter 4, we reflect upon the relevance of early and late fusion and show their

relative relevance towards detection pedestrians across scales and occlusions.

2.3.2 Loss Function

Loss function is one of the most fundamental ingredients of deep learning systems. It is the loss function which guides the training of a CNN. A loss function basically expresses the final objective that is desired of a CNN. Pedestrian detection is a 2-task problem – *classification* and *bounding box regression*. Hence, the loss function in pedestrian detection consists of 2 complementary terms – one for classification and the other one for bounding box regression. The classification term is usually the standard cross-entropy term [86]. The regression term is usually the smooth-L1 loss [130]. While these loss terms have been used in a large number of detection systems spanning both *general-category* [130, 102, 44, 94] and *pedestrian-specific* techniques [159, 108, 16], over time their limitations have come about to be recognized. Recently pedestrian detectors have been proposed [166, 103] which report improved performance on pedestrian detection datasets primarily by virtue of their loss terms' formulation.

Cross entropy often does not fair well in the presence of class imbalance. In detection problems there is generally a heavy imbalance between *positive* (i.e potential object samples) and *negative* (i.e potential background samples). In the case of both *single-stage* and *two-stage* detectors, there are far too many negative anchors compared to positive anchors for classification. This class imbalance problem has typically been handled by using a weighted cross entropy term. While this may help in mitigating the class imbalance problem, it does not differentiate between hard and negative examples. For instance, very small-scale pedestrians are known to be much harder to correctly classify than large-scale pedestrians. As a network trains, it successively learns to classify large-scale pedestrians quite well, but continues to struggle for classifying small-scale pedestrians correctly. This often leads to a flat loss value evolution causing the network performance to plateau. Focal loss term introduced in [44] introduces a weighing factor which is dependent upon the probability of classification of a sample.

$$CE(s_t) \triangleq -y_{s_t} log(p_{s_t}) \tag{2.2}$$

$$FL(s_t) \triangleq -(1 - p_{s_t})^{\gamma} log(p_{s_t}) \tag{2.3}$$

Equations 2.2 and 2.3 represent the standard cross entropy and focal loss for a sample s_t respectively. The term $(1 - p_{s_t}^{\gamma})$ is known as the modulating factor of focal loss. The factor γ in equation 2.3 is known as the *focus* parameter. Assuming that s_t is classified correctly and p_{s_t} is large, it implies that the modulating factor is close to 0. Therefore less weightage is given to such a sample, as the network can already classify it fairly well. In

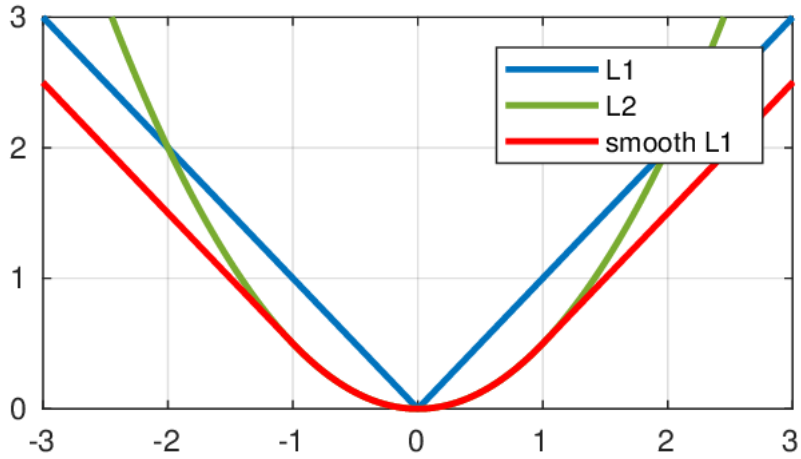


Figure 2.7: A comparison of L1, L2 and smooth-L1 loss functions.

the converse situation, where s_t is misclassified and p_{s_t} is small, the modulating factor is close to 1.0 and hence higher weight is applied to such a sample.

This simple approach has been adopted in a number of pedestrian detection techniques such as in [108, 20] and [103]. Generally this simple modification has been shown to impart $\sim 2-3\%$ of performance improvements.

Regression losses have typically been formulated in terms of the smooth-L1 loss as used in Faster-RCNN [130]. The standard L1-loss ($L1$) imposes a linear penalty on bounding boxes dependent upon their deviation from the groundtruth bounding box. From a visual perspective, very small deviations from the groundtruth bounding box is not a matter of great concern. Smooth-L1 loss which imposes a large penalty on boxes with a large deviation but imposes small penalty to those with small deviations is easier to optimize. Smooth-L1 loss is a piecewise combination of L1 and L2 loss terms and is quadratic in nature near the zero point deviation which corresponds to a bounding box which is in perfect agreement with the groundtruth bounding box. The quadratic nature makes it convex and gets rid of the saddle point which exists in the standard L1 loss. Figure 2.7 illustrates the comparative visual difference in the bahavior of L1, L2 and smooth-L1 losses.

Recently a regression loss function called repulsion loss [166] has been proposed. Repulsion loss is built over the smooth-L1 regression loss. However instead of using the smooth-L1 loss directly, it uses a linear combination of three smooth-L1 loss functions as follows :

$$L_{Rep} \triangleq L_{Attr} + \alpha L_{RepGT} + \beta L_{RepBB} \tag{2.4}$$

In equation 2.4, the first term L_{Attr} is meant to ensure the close proximity of bounding boxes estimated for the same groundtruth bounding box. The second term L_{RepGT} ensures far away separation of a bounding boxe from different groundtruth bounding boxes. The

last term L_{RepBB} is aimed at far away separation of bounding boxes predicted for different groundtruth bounding boxes. The scalar factors α and β balance the last two terms. Determining α and β is crucial but non-intuitive. When a large group of people are closely huddled together, minimization of equation 2.4 is difficult with a high value of α and β. However, in a sparse environment, these values can be high. Experiments conducted in [103, 166] shows that choosing α and β is crucial to obtaining the optimal performance. In view of this [103] adopts the standard smooth-L1 loss for regression.

In chapter 3 we have undertaken an analysis of the impact of some loss functions on the performance of pedestrian detection systems.

2.3.3 Classifier Selection

Most pedestrian and object detection systems utilize fully connected layers for classification. However, other classfiers have been tried. One of the earliest attempts to perform pedestrian detection using other classifiers has been through the use of boosted forests as in [177]. In [177], the output features of a VGG network are fed to a boosted forest with 5 stages, each with different number of trees. The number of trees in each stage is determined as hyperparameters. This choice seems to be motivated by the large variance in the appearance of pedestrians. The adoption of boosted forests in [177] improves the performance on caltech reasonable dataset by $\sim 5\%$. The downside of this approach is the lack of an end-to-end training. Online training of boosted forests though possible is complicated. The complication primarily results from the fact, that as the CNN features evolve in the course of training, the feature splits which determine the tree structure in boosted forests also need to be varied. Attempts have been made to integrate forest based classifiers like random forests with CNNs as in [84]. However, these attempts have produced rather small improvements. In another work [159], gradient boosted trees have been utilized in conjunction with late fusion.

In chapter 4, we outline a proposal for using gradient boosted trees and demonstrate its effectiveness in detecting pedestrians across scales and occlusion levels.

2.3.4 Semantic Segmentation

Segmentation is a powerful indicator of localization than bounding boxes. While a bounding box contains portions of non-pedestrian instances, a segmentation mask is able to encode the specifics of a pedestrian. However, obtaining groundtruth information for segmentation is time-consuming and significantly more costly compared to obtaining bounding box information. Lately, public datasets with segmentation information have been released such as [179]. However, it contains a small number of pedestrians (5000) compared to other much larger datasets such as caltech which come without any segmentation

information. Semantic segmentation information has been used in pedestrian detection since SDS-RCNN [14], which utilizes the pseudo-segmentation mask of a pedestrian. The pseudo segmentation mask refers to the entirety of a bounding box considered as a segmentation mask.

As will be seen in chapter 3, semantic segmentation is an important piece of information whose incorporation frees one from using more refinements such as a dense set of anchor scales and aspect ratios. In this thesis in chapters 5 and 6 we have utilized semantic segmentation in a novel way, not only to improve the feature representation as done in SDS-RCNN [14], but also to limit the number of anchors to be processed which delivers high detection accuracy simultaneously with high detection speed.

2.3.5 Boosting

Boosting [51] is a well-studied area of machine learning. Boosting involves training an ensemble of weak classifiers. Weak classifiers are defined as classifiers which have an average classification accuracy of $< 50\%$. Boosting is a meta-algorithm, meaning that is independent of the choice of classifiers and features as long as each classifier matches the criterion defined before for weak classifiers. Boosting involves iterative training of weak classifiers using reweighing of features. This means that features which are classified well by previous weak learners are given lesser weightage, while those which are not well classified are given a large weightage. There are many boosting algorithms such as AdaBoost and gradient boosting.

Boosting in deep learning is a costly undertaking. Deep learning architectures such as CNNs consist of large number of parameters and hence iteratively training a large number of them is therefore deemed difficult. Nevertheless, this approach has been used in some approaches such as in CFM3 [72]. This approach to utilizing boosting involves training a number of different CNN based detectors and training them in the same iterative fashion as in AdaBoost. The downside of this approach is the substantial increase in the bulk of the overall system. In contrast, the corresponding improvement in performance is small ($\sim 1\%$).

Usually instead of boosting a more ensemble approach is followed in pedestrian detection literature such as in cascade-rcnn [18]. In cascade-rcnn, a cascade of detection heads are setup. Each detection head consists of sibling layers for classification and bounding box regression respectively. For cascade-rcnn with N stages, the i^{th} stage ($1 < i < N$), is trained to detect pedestrians during training with an IoU ($IoU_{(i)}$) such that

$$IoU_{(i-1)} \leq IoU_{(i)} \leq IoU_{(i+1)} \tag{2.5}$$

The fundamental motivation behind cascade-rcnn is to make detection quality consistent

across IoU overlaps. However, cascade-rcnn is a costly approach due to two principal reasons – a) stages of cascade-rcnn are trained sequentially and, b) for large number of stages, cascade-rcnn is slow to train and test. Use of large number of cascade stages also leads to a large memory footprint. This is a limitation in deployment scenarios where large memory devices might not be favored due to cost, bulk or power consumption restraints.

2.4 Datasets

In this section we look at some public datasets for pedestrian detection used for our work. A summary of the basic statistics of these datasets is detailed in table 2.3. While a number of other datasets exist and have been popular such as INRIA [31], KITTI [53], ETH-Zurich [39] and TUD-Brussels [168], in this thesis we focus primarily upon caltech, citypersons and BDD100K datasets. Our choice stems from a couple of following considerations :

- Caltech, CityPersons and BDD100K come from the same domain of *autonomous driving*. Hence, they belong to the same environment of driving, and cross-dataset studies are more disciplined.
- These datasets are large and becoming popular with the pedestrian detection community.
- These datasets taken together, cover a diverse gamut of resolution (table 2.3) and dataset size.

	Resolution	**Number of Images**		**Number of pedestrians**	
		Training	**Testing**	**Training**	**Testing**
Caltech [33]	480×640	128422	121468	188184	146807
CityPersons [179]	1024×2048	2975	1575	19654	11424
BDD100K [174]	720×1280	70000	20000	86047	43215

Table 2.3: A summary of 3 public datasets used in our work.

2.4.1 Caltech Pedestrian Dataset

The caltech dataset [33] is one of the oldest large-scale public datasets for pedestrian detection. It is available in the form of a set of videos, alongwith a predefined *training/testing* split. These videos are taken with a temporal frequency of 30 Hz. Hence, subsequent frames are quite similar for this dataset. Hence, frames taken with a predefined interval are normally used for the evaluation of this dataset. This often brings down the number of images from the figures appearing in table 2.3.

Frames within the testing subset taken at intervals of 30 frames are used for evaluation, while for the training subset the frame interval is reduced to 3 frames. These subsets are commonly known as *caltech1x-test* and *caltech10x-train* sets.

The original annotations for the caltech dataset were obtained through interpolating manual annotations over a subset of frames. This has led to original annotations suffer from alignment problems. A set of new and improved annotations were released in [47] to address these problems. These annotations were made available only for *caltech10x-train* and *caltech1x-test* subsets. Due to this, most modern works Most modern works such as RPN-BF [177], SDS-RCNN [14], MSCNN [16] and our own proposed approaches make use of the *caltech10x-train* and *caltech1x-test* subsets when working with the caltech dataset.

Most evaluations further refine *caltech10x-train* and *caltech1x-test* on the basis of pedestrian height and level of occlusion. The most commonly used subset on this basis is known as the *caltech-reasonable* subset. This contains all pedestrians with a minimum height of 50 pixels and an occlusion of less than 35%. Most current techniques strive for an improved performance on this subset.

2.4.2 CityPersons

The citypersons dataset [179] has been assembled from the cityscapes dataset [28]. For the citypersons dataset, the pedestrians in the cityscapes dataset have been manually annotated with bounding boxes, while maintaining an aspect ratio (width/height) as 0.41. In addition, the visible and full body bounding boxes have been made available. In contrast with the caltech dataset, it has been gathered from 27 different cities in Germany and surrounding areas. It also involves a significantly high pedestrian density (*pedestrians/image*) of 7.0 compared to other datasets like caltech (≈ 1.4) and BDD100K [174] (≈ 1.2). In the paper proposing this dataset [179], benchmark evaluations confirm that Faster-RCNN [130] trained on citypersons generalizes better on the caltech-reasonable test set compared to other datasets such as KITTI [52].

2.4.3 BDD100K

BDD100K is a recently released dataset [174]. The salient feature of this dataset is its diversity of illumination and weather conditions. BDD100K apart from annotations for pedestrian detection, also contains annotations for detection of 9 other entities commonly found on streets. It also contains annotations for lane markings and drivable areas along with full-frame fine instance segmentations over a subset of 10K images. Compared to the other two datasets (caltech and citypersons), BDD100K comes with only full-body bounding boxes.

2.5 Evaluation Metrics

In the object detection literature, *mean averaged precision* (mAP) is the most commonly used evaluation metric. In contrast, log-Averaged Miss Rate (abbreviated here as LAMR) is the most common evaluation metric for *pedestrian* detection. The reason for a more widespread adoption of LAMR compared to mAP, is the fact that in most practical applications of pedestrian detection, there is a high cost associated with the failure to correctly detect a pedestrian. LAMR measures the miss-rate of a pedestrian detection system, thereby being suitable as an evaluation metric for pedestrian detection. This metric is computed using a miss rate curve. A miss-rate curve is a plot of *miss-rate* (MR) vs. *false-positives per image* (FPPI). The miss-rate curve is a *log-log* plot; both its axes having represented in the *log*-space. LAMR is obtained by selecting 9 equidistant points on the x-axis between 10^{-2} and 10^{0}, and the MR values at those points averaged together (MR_{-2}). Some evaluation also do this averaging between 10^{-4} and 10^{0} (MR_{-4}). Since, the threshold of a classifier can only be varied in discrete steps which do not bear a 1-to-1 mapping to FPPI, at some or all of those 9 points MR values may require interpolation from other points for estimation.

It has been stated in [34] that miss rate curves are preferred over traditional precision-recall curves because in practical instances, there is an upper limit on FPPI and this is communicated directly by the miss rate curve. However this evaluation protocol is not suitable for instances where – a) Setting an upper limit on FPPI is not informative or, b) Where a holistic performance over multiple object categories is desired. The rising field of *self-driving* vehicles is such an example. *Self-driving* vehicles need a holistic understanding of the environment which includes but is not limited to *buildings*, *vehicles*, *lanes* and *traffic signs*. Furthermore, setting an upper limit on FPPI may be difficult in uncontrolled environments where pedestrian density can vary over time.

This indicates a need to possibly shift from the LAMR metric to mean averaged precision (mAP). Such a shift could allow a more exhaustive evaluation of *pedestrian* detection vis-à-vis generic object detectors.

2.6 Conclusions

In this chapter we gave an overview of pedestrian detection systems and how they were derived from general object detector. We characterized the single stage and two stage deep learning detectors, and showed the relevance of feature handling in pedestrian detection. This characterization is helpful to understand the landscape of pedestrian detection research in deep learning. We also looked at various refinements and modifications introduced to the general category object detectors for building pedestrian detection systems. These specializations indicate two major trends in pedestrian detection today – *moving*

towards more expressive architectures and, *utilizing more semantic information in contemporary pedestrian detection systems*. We take cue from these observations, and hence set the tone for the next chapter where we conduct an exhaustive quantitative analysis of different refinements which describe the majority of pedestrian detection systems in use today.

Chapter 3

Quantitative Analysis of Pedestrian Detection in Deep Learning

Contents

3.1 Introduction

The construction and training of deep learning pipelines for pedestrian detection is a challenging task. These challenges can be categorized as – *computational challenges* and *design challenges*. Computational challenges refer to constructing and training a deep learning pipeline under a constrained set of computational resources such as CPU/GPU memory. Design challenges refer to designing various components of a deep learning pipeline and training them in an effective manner. An example of design challenge is the selection of hyperparameters such as depth of a network and learning rates. There is very limited body of work on the analytical study of the behavior of deep learning systems under a

given set of hyperparameters. Most of such studies are conducted under simplified scenarios such as in the case of fully connected layers. Modern pedestrian detection systems are built using various components such as very deep CNNs with convolutional, pooling and fully connected layers put together and involve millions of parameters. For example, a pedestrian detector such as SDS-RCNN [14] built using the VGG16 architecture [140] has in excess of 15 million parameters. At the same time, the largest publicly available pedestrian detection dataset (*in terms of number of images with pedestrian(s) in them*) – the EuroCityPersons dataset [13] has slightly more than 40K images in its training set. This exhibits a huge disparity between the number of training samples and the number of parameters to optimize, and this makes achieving the global minimum during optimization substantially difficult. To add to this misery, a large number of design choices are available to anyone willing to design a new pedestrian detector. These design choices include but are not limited to – *selection of a CNN architecture*, *selection of an optimizer*, *selection of a loss function* and *selection of a dataset for initial training*. In the lack of tractable analytical studies which shed light on the impact of any specific combination of the aforementioned and other choices, to the performance of the detector on a dataset (*training or testing*), effective design and training of pedestrian detectors is a formidable undertaking.

In this chapter we conduct a systematic study of the design choices in 4 deep learning based systems popular for pedestrian detection – Faster-RCNN [45], SSD [102], SDS-RCNN [14] and RPN-BF [177]. Of these, Faster-RCNN and SSD were proposed for *general object detection*, while SDS-RCNN and RPN-BF were proposed specifically for pedestrian detection. Our analysis makes use of 3 large and popular public datasets for pedestrian detection – Caltech [33], CityPersons [179] and BDD100K [174]. The objective at the root of this study is to elicit a common set of guidelines which enable one to start off making design choices on a less than random basis. To stress the relevance of this study, we allude to the relevance of various schemes of initialization of deep neural networks, where it is well known that certain initialization schemes such as xavier initialization [57] are better than others such as zero initialization. It is therefore very pertinent to conduct such systematic analysis as we present in this chapter.

Our choice of the aforementioned 4 deep learning systems is based on a combination of two bases – a) **Popularity :** Faster-RCNN [130] and SSD [102] have served as the basis for a large number of pedestrian detection approaches as outlined in chapter 2. Therefore, their analysis serves to provide invaluable insights into the guidelines to follow when training and finetuning other pedestrian detection approaches and b) **Diversity :** SDS-RCNN [14] and RPN-BF [177] introduce major specializations for pedestrian detection. RPN-BF [177] utilizes a combination of CNN (*for feature extraction*) and Gradient Boosted Trees (GBT, *for classification*). It is a remarkably different approach since with the advent of deep learning, most classification approaches have utilized fully connected lay-

ers for classification. SDS-RCNN [14] introduced the concept of a pseudo-segmentation mask which is the first approach of its kind in pedestrian detection for improved feature representation. The idea that the bounding box of a pedestrian can be utilized as a segmentation mask and incorporated in the training process for better feature representation of pedestrians is a major enhancement, thereby leading to our choice of it.

3.2 Related Work

There is a general lack of available literature conducting a focused and elaborate analysis of the design choices and training strategies for pedestrian detection systems. While there have been several papers conducting a systematic review of pedestrian detection [10, 34, 54, 37, 93, 178], most of them such as [10, 34, 54, 37, 93] having been written before the advent of *deep learning*, thus provide little insights into the potential design choices for deep learning systems. The most recent review on pedestrian detection [178] focuses primarily on data annotation quality and its impact on detection performance. While annotation quality is of fundamental importance, it is rarely within the control of system designers. Reviewing large volumes of previously annotated data is desirable, although usually not a preferred practice due to its costly nature. Therefore, it is also important to take discussion closer to the underlying techniques of detection and review their impact on detection performance.

Designing a deep learning system for *pedestrian detection* based on existing methods, involves a selection of wide array of parameters that may affect the performance. As an example, a seemingly simple operation such as convolution comes in a number of flavors such as *à trous convolution* [21] and depthwise separable convolution [25]. To the best of our knowledge, there exists no systematic analysis of how these design choices affect performance when incorporated in existing frameworks such as Faster-RCNN [45]. Often when designing a system for a custom dataset, it is not possible to re-annotate the complete dataset, despite some potential imperfections such as misalignments. A systematic understanding of design factors can aid in getting the best out of a deep learning system design. We therefore propose this chapter not as an exhaustive survey, but rather as an analysis of how different design choices in different frameworks relate to their corresponding performance. We also make observations on the *cross-dataset generalization* of different frameworks.

3.3 Common Experimental Settings

In this section we setup the basis for our subsequent analysis. We describe the settings which were common across all our experiments. We also outline the basis for these settings. We look at each experimental settings in the subsequent subsections.

3.3.1 Selection of Optimizer

Optimizers used in deep learning can be categorized into – a) Adaptive (e.g :- Adam [81] and AdaDelta [175]) and b) Non-Adaptive (Stochastic Gradient Descent (SGD)). Adaptive optimizers internally adjust the learning rate while non-adaptive optimizers do not.

Adaptive optimizers generally lead to faster convergence but poorer generalization compared to non-adaptive optimizers as outlined in [126, 80]. Considering that generalization performance is of major practical relevance, we take cue and decide against using adaptive optimizers for our analysis. Some recent techniques such as Regularized Nonlinear Accelaration (RNA) [133] have been developed to boost the convergence rate of non-adaptive optimizers. Their performance has been validated on image classification problems [134]. These accelaration techniques however are memory intensive, as they keep in memory the gradient history of the network upto a fixed number of past training steps. Our experiments with using the RNA algorithm (*by storing the gradient history of past 20 steps*) with Faster-RCNN (on VGG16 network) led to around 15% better convergence speed, but at the cost of significant increase in memory usage which made its usage untenable with GPUs with a memory less than 12GB. Thus, although proven effective in our experiments, we decide against using RNA algorithm for pedestrian detection analysis. Other methods for accelarating SGD which do not require the storing of gradients are also known such as Predictive Variance Reduction (PVR) [79]. However PVR is not meant for use when SGD is used with the momentum term. SGD with momentum term remembers the update δW_t at each iteration t, and determines the next update (*at* $t+1$) as a linear combination of the gradient at that iteration (i.e $t+1$) and the previous update (δW_t). The coefficient μ of δW_t in this linear combination is known as the momentum term. The momentum term eases the escape of the optimizer from local minima and saddle points [175].

We employ SGD with momentum term for all our experiments. The prime reason guiding our choice is the analytical proof presented in [61] and [167] stating that models trained using SGD are capable of attaining a vanishing generalization error.

One recently proposed technique [80] known as the **Sw**itches from **A**dam **t**o **S**GD (SWATS) uses a combination of SGD and Adam. The training starts with the Adam optimizer and on the fulfillment of certain condition (*known in [80] as the triggering condition*)

switches to SGD, with little memory overhead. We have not experimented with this optimization technique but it could be an important part of future experiments.

One downside of using SGD is the need to carefully choose the learning rate and momentum during training. We next describe our choice of learning rate and momentum in our experiments.

3.3.2 Learning Rate Schedule and Momentum

Our choice of momentum (μ) is 0.9, as it is the most common value used in most of the literature [14, 177]. One probable reason for this value is that, the loss function landscape can be very undulating and a large value of μ is helpful in escaping local valleys and saddle points. Too large a value of μ (e.g:- > 1) however, upsets the contribution of the gradient at that step and gives a much larger weightage to the previous update, and hence is undesirable.

The learning rate (η) is the single most important parameter of an optimizer (*adaptive* as well as *non-adaptive*). The learning rate determines the contribution of the current gradients towards the weight update. A very large value of η is usually the most recognized cultprit behind the non-convergence of a deep network. A too low value of η causes very slow convergence and may also result in oscillations around a local minima region. Generally a constant value of η is not found suitable for training deep networks. As a network begins to converge, the standard practice is to lower the learning rate. Failure to do this could result in the optimizer moving out of the global minima region, thereby upsetting the training process. Generally a warmup phase of training is preferred where the learning rate is set at a high value for the first few thousand steps and then gradually lowered. This practice has been followed in several papers on object detection such as [172]. The initial warmup phase is helpful when training a network from a randomly initialized state. It is not a good practice to follow when finetuning a network, as a high initial learning rate can upset the weights learnt from the pre-training. Moreover, the initial high learning rate during the warmup phase is primarily a mechanism or trick employed for faster convergence. There is no analytical study confirming the superiority of using a warmup phase as against not using one.

In this chapter for all our experiments, the learning rate was set initially at 0.001 and then decreased by 10 times after every $20K$ steps. Due to the highly uneven and undulating nature of the loss function landscape, it is indeed possible for the gradients to blow up suddenly to a very large value. This phenomenon known as exploding gradients is usually seen in the early stages of training a network. While, it is not a regular occourance, it is best practice to safeguard against it by utilizing gradient clipping. Gradient clipping is a process, where before applying the update rule of an optimizer, the computed gradi-

ents are clipped i.e, gradients above a pre-specified threshold are set to the value of that threshold. Thus if G_u is the value of unclipped gradient then the clipped gradient value G_c is determined as follows

$$G_c = \begin{cases} \tau & G_u \geq \tau \\ G_u & otherwise \end{cases} \tag{3.1}$$

An approximate value of τ can be guessed empirically by initially running a few iterations of training without using gradient clipping and observing the minimum and maximum values of gradients over the iterations. Based on that, one can estimate the mean, minimum and maximum values of gradients and can determine τ by observing these statistics. When experimenting with Faster-RCNN [130] or SSD [102], we have found across several networks (e.g:- VGG16, ResNet-152 etc.), that the mode of initial gradient values is around 10, with this value decreasing over the iterations. We therefore set the value of τ as 10.0.

3.3.3 Initializer

We make use of the xavier initialization scheme [57] for initializing any layer whenever training from scratch or fine-tuning some layers needing initialization. This choice of initializer is on account of its superior performance as it helps stablize the training and prevents a network from entering into a local minima position early on.

3.4 Evaluation Protocols

The evaluation protocols in pedestrian detection have been described in detail in chapter 2. In this work, the evaluation has followed the protocol suggested in [47]. In this protocol, *caltech10x* (training subset of the caltech dataset sampled at 3 frames interval) has been used for training. The *caltech1x-test* (testing subset of the caltech dataset sampled at 30 frames interval has been used for testing. The evaluation for *caltech reasonable* is done by considering pedestrians with $height > 50px$ and $occlusion < 0.35$ in the *caltech1x-test* subset. The refined annotations proposed in [47] for *caltech10x-train* and *caltech1x-test* have been used for training and testing respectively.

3.5 Analysis of Design Choices for Pedestrian Detection

3.5.1 Base Network Architecture

The basic design question with respect to *base networks* is their architecture. The feature expressive power of a CNN is empirically related to its depth and the connectivity

amongst its layers (e.g :- Residual connections). This trend has been noticed consistently across images (*image classification*) and videos (*action recognition*) [60]. To quan-

	VGG16 (LAMR)				
	Reasonable	**near-scale**	**medium-scale**	**partial-occ**	**heavy-occ**
Faster-RCNN	13.34	4.09	49.78	40.14	68.10
SSD	15.45	4.13	51.12	43.56	70.03
RPN-BF	9.2	3.86	47.2	18.23	61.4
SDS-RCNN	**7.16**	**2.11**	**47**	**15.46**	**58.27**

Table 3.1: Relative performance of different techniques on caltech pedestrian dataset with VGG16 as base network. **Reasonable** subset is described in chapter 2. For scale analysis, only unoccluded pedestrians are used. For occlusion, only pedestrians with heights $\mathbf{H} \geq 50$ pixels are used. For **Near-scale**, $\mathbf{H} \geq 80$ pixels. For **Medium-scale**, $\mathbf{H} \in [30, 80]$ pixels. **Partial-occ** refers to occlusion $\mathbf{O} \in [1, 35]\%$. **Heavy-occ** refers to occlusion $\mathbf{O} \in [35, 80]\%$. All figures are LAMR percentage values. **The best result in each column is in bold**.

	ResNet-152 (LAMR)				
	Reasonable	**near-scale**	**medium-scale**	**partial-occ**	**heavy-occ**
Faster-RCNN	11.17	3.28	46.60	38.1	64.32
SSD	12.84	3.33	47.94	45.02	65.89
RPN-BF	8.92	3.02	45.22	16.77	60.3
SDS-RCNN	**7.02**	**1.2**	**44.13**	**14.3**	**56.12**

Table 3.2: Relative performance of different techniques on caltech pedestrian dataset with ResNet-152 as base network. Column description same as in table 3.1. All figures are LAMR percentage values. **The best result in each column is in bold**.

titatively look at the impact of base network selection, we trained the 4 techniques on the training set of all the 3 datasets (CAL(10x)+CP+BDD). The training set included all pedestrians irrespective of their heights and occlusion levels. The trained model was then tested on caltech1x-test dataset. The results are summarized in tables 3.1 through 3.4. A visual summary of the quantitative figures is in figure 3.1. The depth of networks in tables 3.1-3.4, increases in the order – VGG16 [141], ResNet-152 [66], Inception-V2 [146], Inception-V3 [147]. In the above order, the increse in depth decreases as we move from VGG16 towards Inception-V3. There is a substantial difference in the number of layers between VGG16 and ResNet-152, while there is very little difference between Inception-V2 and Inception-V3 in terms of depth. Between ResNet-152 and Inception-V2 there is a substantial architectural difference which will be described later on in this section.

Analysis of tables 3.1-3.4 reveals that though network depth boosts the performance of these systems, the improvements are not linear. Improvements though quite pronounced from VGG16 –> ResNet-152 transition, are less pronounced on subsequent transitions

	InceptionV2				
	Reasonable	**near-scale**	**medium-scale**	**partial-occ**	**heavy-occ**
Faster-RCNN	11.15	3	45.92	37.65	63.62
SSD	12.01	3.02	46.68	44.37	65.16
RPN-BF	8.13	2.98	44.47	15.2	58.76
SDS-RCNN	**6.87**	**0.4**	**43.71**	**15**	**55.24**

Table 3.3: Relative performance of different techniques on caltech pedestrian dataset with InceptionV2 as base network. Column description same as in table 3.1. All figures are LAMR percentage values. **The best result in each column is in bold**.

	InceptionV3				
	Reasonable	**near-scale**	**medium-scale**	**partial-occ**	**heavy-occ**
Faster-RCNN	11.08	2.38	44.98	35.14	62.42
SSD	11.74	2.98	45.07	43.17	64.18
RPN-BF	8.06	2.89	43.94	14.86	55.23
SDS-RCNN	**6.5**	**0.04**	**42.84**	**14.74**	**54.21**

Table 3.4: Relative performance of different techniques on caltech pedestrian dataset with InceptionV3 as base network. Column description same as in table 3.1. All figures are LAMR percentage values. **The best result in each column is in bold**.

through InceptionV2 and InceptionV3. Moreover the improvements are more pronounced for the medium-scale and partially occluded pedestrians. We therefore see that as the difference in the depth of 2 networks decreases, so is the difference between their impact on performance. This motivates us to look closer and understand the impact of architectural differences amongst these networks and how they impact the performance.

3.5.1.1 Architectural Differences and their impact on Detection Performance

Each one of VGG16 [139], ResNet-152 [66] and Inception-V2 and V3 [147] are made up of some fundamental building blocks. Instead of describing the complete architecture of these CNNs, we limit ourselves to understanding their building blocks as they uncover the performance trends observed in tables 3.1 through 3.4. Figure 3.2 illustrates the fundamental building blocks for the three base network architectures used in our analysis. VGG16 [140] is built from a cascaded stack of 3×3 filters, each with a zero padding of 1. The entire VGG16 network is built from repeatition of the block shown in figure 3.2(a). The number of 3×3 convolutional layers vary from one block to the other. A cascade of 2 3×3 convolutions, each with a zero padding of 1, corresponds to a receptive field of 5×5. Therefore, as one moves from the early layers of VGG16 to the latter layers, one analyzes the input image at larger scales. The cascaded structure of VGG16 however, makes it difficult to retain information across scales. The very first 3×3 convolutional layer in the

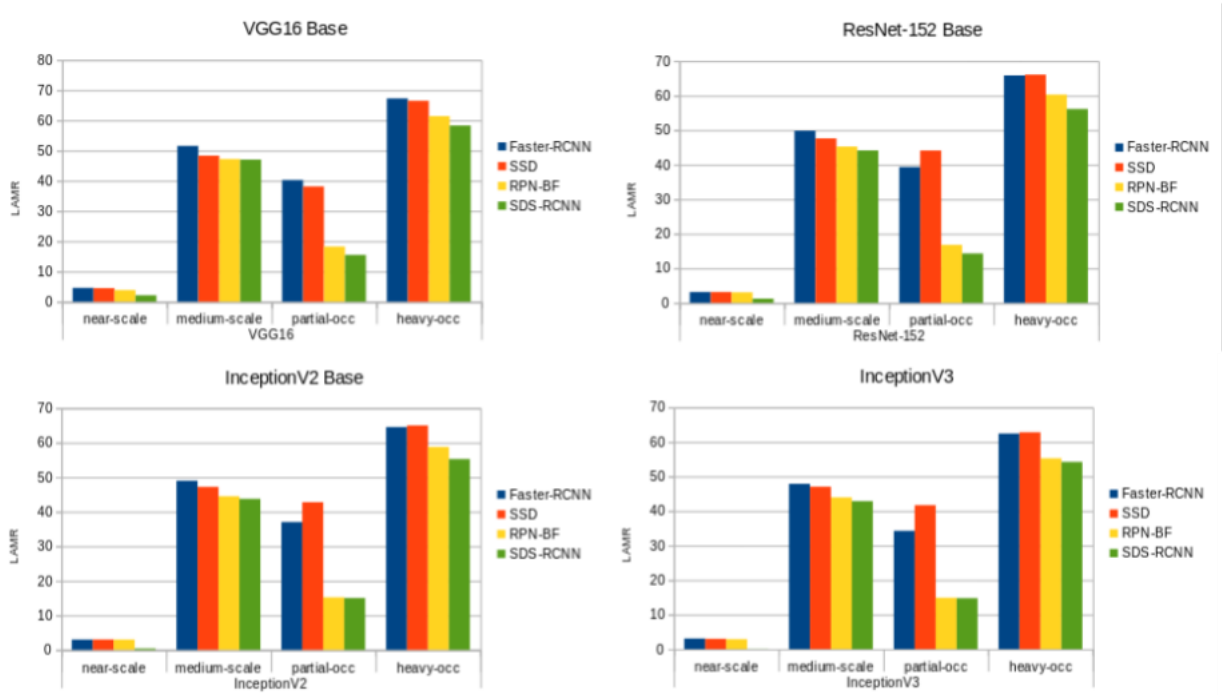
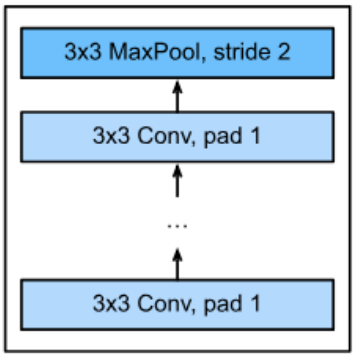


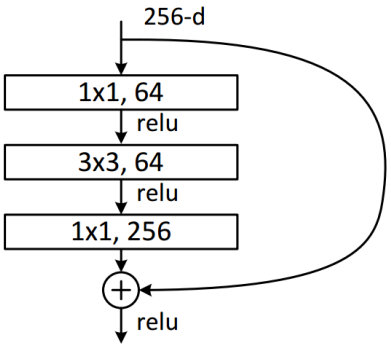
Figure 3.1: Scale-wise bar charts with a visual summary of figures in tables 3.1 through 3.4.

first block of VGG16, is able to extract features at a small scale (*due to a small receptive field*). Consider the first and third convolutional layers in a VGG16 block. The input to the third convolutional layer contains the information about the smaller image scale in a very indirect fashion. Furthermore, pooling operations between two blocks further diminish this information. Thus, the feature map output from the last layer of VGG16 has very little information about the small scale features of the input image. This explains the rapid degradation of LAMR as one moves from left to right (*columnwise*) in table 3.1. While, due to pooling layers in all CNN architectures, there is a loss of scale information, VGG16 does not make any effort to counteract it. This however is handled by ResNet-152 and Inception-V2 and V3 networks through their specific architectures.

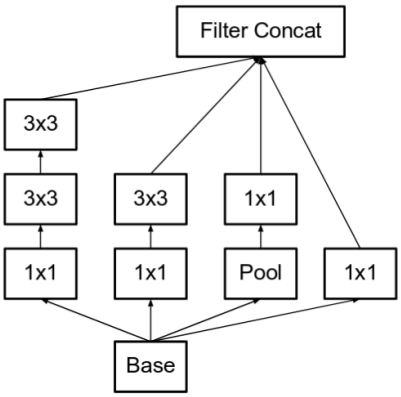
The fundamental building block of all architectures in the ResNet family is a "bottleneck residual block" (figure 3.2(b)). Multiple residual blocks are stacked interspersed with pooling layers to create the resnet architectures. The two 1×1 convolutions, first compress and then expand the feature map in the channel dimensions. By reducing the channel dimensions, the computational load of the 3×3 convolution in the middle is significantly reduced [67]. The feature map is then expanded across the channel dimensions to avoid loss of information fed to the next residual block. ResNet-152 employs residual connections which make it behave like an ensemble of multiple networks [160]. Residual



(a) VGG16 [140]



(b) ResNet-152 [66]



(c) Inception-V2 and V3 [147]

Figure 3.2: Building blocks of CNN architectures used in our analysis of the impact of base network on pedestrian detection performance.

connections also lead to a richer set of features getting generated. Due the residual nature, there is a more direct flow of information from lower layers to higher layers in the architecture.

The focus of inception family of architectures is on employing filters of multiple kernel sizes at the same layer depth so as to capture features of multiple scales easily. The building block shown in figure 3.2(c), shows this phenomenon. The primary difference between Inception-V2 and Inception-V3, is the use of batch normalization [75]. In terms of scale handling, inception family is a much better class of networks than the resnet family. Due to parallel branches processing information on multiple scales, there is a much greter retention of small-scale image features. In the case of ResNet, there are no parallel branches and hence, the flow of small-scale information is substantially less.

More pronounced improvements in medium-scale and partial occlusions show that Resnet and Inception networks are better at dealing with these aberrations. Performance

improvements with heavy occlusions are comparatively less. Heavily occluded pedestrians are delineated by their context rather than visual appearance. It suggests focus on context as a more relevant factor for detection of heavily occluded pedestrians.

From the previous discussion we gather an important information – while a deeper network performs better than a shallower network, the architectural specifics of a CNN play an important role in its fine-grained performance across scales and occlusion levels. It is important to note that, in practical instances, the selection of a CNN architecture is also guided by resource constraints such as available amount of processing memory. As an example, we present the case of two members of the inception family of architectures – InceptionV3 [75] and InceptionV4 [145]. InceptionV4 has a similar architecture as InceptionV3 but is deeper and has about 2 times the parameters as InceptionV3. As discussed in section 3.5.1.1, inception family of architectures has better scale handling compared to other architectures presented in section 3.5.1. Thus, on account of this and its deeper nature, InceptionV4 is a lucrative choice for use as a CNN architecture. However, as noted in a seminal review of CNN architectures [19], InceptionV4 has a higher inference time ($\sim 40\%$ more) and twice the power consumption as compared to InceptionV3. Furthermore, networks with large number of parameters are difficult to train or fine-tune when having a small dataset such as Citypersons. This brings out the conclusion that the choice of a network architecture is an ensemble of several decisions which go beyond their depth and architectural details. Thus in practice, it is fruitful to consider the practical constraints of memory, power consumption and speed requirements when considering a CNN architecture for use.

3.5.2 Convolution Techniques

3.5.2.1 à trous convolution

à trous convolution [69], increases the field-of-view of filters. This promotes features maps with a high spatial density. Such feature maps have been shown to improve results by a considerable margin in segmentation problems [21, 22]. à trous convolution is also used during inference by some pedestrian detection techniques such as RPN-BF [177].

à trous convolution is obtained by inserting *zeros* between non-zero elements of a filter kernel. Existing pre-trained filters can be used for à trous convolution without any re-training while re-training can also be employed.

In [177], it is reported that introducing à trous convolution with $rate = 2$, on the conv5_3 layer of VGG16, improves the miss-rate by nearly 4%. à trous convolution with $rate = 2$, means that one zero is introduced between every pair of non-zero elements in the filter kernel. The same trend is reported for the case when *boosted forests* are used for classification (miss-rate on à trous application reduces by nearly 5%).

à trous convolution can be a very effective technique employed in cases where feature resolution is too low for reliable classification. This is especially true for *small-scale*, when extracting features from the later layers of a network. Small-scale pedestrians are greatly reduced in resolution due to multiple pooling applications down the layers of a CNN.

à trous convolution also brings with it additional questions which need analysis for their resolution. To the best of our knowledge, the effect of employing à trous convolution in various layers has not been analyzed. Existing works (such as [177]), have used à trous convolution on the layers from which feature maps are sent to RPN. However, it is possible to employ it on other layers. The exact nature in which this impacts the feature quality is an open question and needs to be analyzed. The *rate* value is an equally important parameter. Higher *rate* values can reduce feature stride, but because of their very large fields-of-view may degrade the discriminative quality of features.

Overall à-trous convolution is a promising approach in the absence of other information aiding the training process (e.g- segmentation loss in [14]). The exact extent of its ramifications however need more work for resolution.

3.5.2.2 Depthwise Separable Convolution

During standard convolution in deep networks, a filter *kernel* during inference convolves with all the input feature channels. These convolution results are summed to create one feature channel of the output. In contrast, during depthwise separable convolution [25], each input feature channel is convolved with a separate *kernel*. The resulting convolutions are stacked depthwise, followed by a 1×1 convolution to generate one feature channel of the output.

Therefore, in standard convolution, a single filter *kernel* is tasked with mapping cross-channel as well as spatial correlations. The depthwise separable convolution approach, relaxes this constraint, by separating the process in two steps. While the first step involves a set of filter *kernels* performing channel specific spatial correlations, the second step involves 1×1 convolutions to perform cross-channel correlations.

Depthwise separable convolutions are used in the Xception architecture [25], to perform image classification. The reported results in [25], indicate marginal improvements. To the best of our knowledge, depthwise separable convolutions have not been utilized in pedestrian detection literature yet.

It is however, of interest to utilize depthwise separable convolutions on account of fewer parameters compared to standard convolutions. For an input with C_I channels, if an output with C_O channels is desired by using a filter kernels of size $k \times k$, normal convolution yields $C_I \times C_O \times k \times k$ parameters. With depthwise separable convolution for the same configuration, the number of parameters turn out to be $(C_I \times k \times k) + (C_I \times C_O)$,

which is lower than the previous number.

	Inception-V3									
	Reasonable		near-scale		medium-scale		partial-oc		heavy-occ	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
Faster-RCNN	11.03	11.08	2.37	2.38	44.97	44.98	35.09	35.14	62.4	62.42
SSD	11.54	11.74	2.90	2.98	44.69	45.07	43.01	43.17	63.88	64.18
RPN-BF	7.95	8.06	2.47	2.89	43.13	43.94	14.3	14.86	55.20	55.23
SDS-RCNN	6.44	6.5	0.04	0.04	42.77	42.84	14.53	14.74	53.86	54.21

Table 3.5: Impact of Depthwise Separable convolution (**w**) when replacing normal convolutional layer (**w/o**) in the last CNN layer in 4 pedestrian detection frameworks.

	VGG16									
	Reasonable		near-scale		medium-scale		partial-oc		heavy-occ	
	w	w/o	w	w/o	w	w/o	w	w/o	w	w/o
Faster-RCNN	12.63	13.34	3.8	4.09	48.12	49.78	39.02	40.14	67.01	68.10

Table 3.6: Impact of Depthwise Separable convolution (**w**) when replacing normal convolutional layer (**w/o**) in the last CNN layer in Faster-RCNN with VGG16 as the base architecture.

To study the impact of depthwise separated convolutional layers, we utilized the Inception-V3 based architectures for Faster-RCNN [130], SSD [102], RPN-BF [177] and SDS-RCNN [14]. The last CNN layer(s) in these detectors – which generate the feature map from which the features for final classification and bounding box regression are obtained – were replaced with depthwise separable convolutional layers. The kernel size and the number of output feature channels were kept the same as their non-depthwise counterparts. These frameworks were then trained in the same way as in section 3.5.1. Table 3.5 summarizes the results of using depthwise separable convolution as against their non-depthwise counterparts. Depthwise separable convolution does provide consistent improvements but the resulting improvements are small. These small improvements we believe are due to the sufficiently powerful feature representation power of the Inception-V3 base network. To verify this, we repeated the above experiment with the VGG16 base architecture for Faster-RCNN. The resulting performance in table 3.6 shows that improvements are more compared to those in table 3.5. This substantiates our understanding that the feature representation power of the base network architecture is one of the most prominent components of a pedestrian detection system.

We have not conducted a more exhaustive analysis of performance impacts when using depthwise separable convolution on other CNN layers. We reckon that by virtue of its lesser number of parameters, it is possible to garner increased speed from them. In terms of detection accuracy however, we do not expect a significant performance boost over

non-depthwise components. This is partly due to the observations in the original paper proposing depthwise separable convolution [25], which shows that the performance boost of replacing all convolutional layers with depthwise separable convolution in Inception-V3 is not significant.

3.5.3 Role of Convolutional Layer Selection in Pedestrian Detection

All *pedestrian detection* frameworks use convolutional base networks for feature extraction. Feature maps from one (e.g- Faster-RCNN [45]) or more layers (e.g- SSD [102]) are used for subsequent processing. CNNs being hierarchical, detect features with different semantic information at different layers. Early layers are associated with low-level features such as edges, while later layers encode high-level information not easily interpretable.

This hierarchical difference in the features extracted by different CNN layers can be used to detect different types of pedestrian instances more robustly. Pedestrians close to the camera have a lot of their details visible. Many of those details (e.g - *clothing*, *accessories* etc) vary widely among pedestrians. Higher level features are thus required to robustly encode the pedestrian-specific information for such instances. In the other extreme case, pedestrians very far away from the camera have fewer details visible. In the absence of motion information in still images, their contour is generally the only useful information available. It requires effective discrimination of their low-level information from the surroundings for robust classification. This observation has been made use of in MSCNN [15], to form a multi-scale system. A scale-wise division is made of a dataset during training and Region Proposal Networks (RPNs) attached to different layers process a certain scale-range of pedestrians as groundtruth samples pre-defined during training. Their approach released around the same time as [177], provides a similar miss-rate of 10% as [177] (9.6%) on the caltech reasonable test set. However, MS-CNN performs better on – *partially occluded pedestrians* (19% compared to 24% of [177]) and *medium scale pedestrians* (49% compared to 54% of [177]). Another recent work [158] shows that early layers are better at detecting small-scale and partially occluded pedestrians while later layers perform better for large-scale and unoccluded pedestrians. The performance achieved by [158], is 9.25% on the caltech reasonable dataset, which is better than [15], though worse than SDS-RCNN [14]. Although the late fusion reported in [158], showcases an approach to utilize multiple layers, it is less effective on account of training multiple heads. In chapter 4 we report experiments of utilizing various convolutional layers with respect to various scales and occlusion levels of pedestrians.

While the use of multiple layers for pedestrian detection is an intuitively appealing idea, its optimal usage needs further research. More specifically, there are two major issues surrounding the usage of multiple layers – a) How to select the layers to be modified

by a specific detection head ? and b) How to combine the detections from multiple layers ?. During standard techniques such as Faster-RCNN, the standard practice is to keep the initial two layers fixed. However with multiple layers simultaneously detecting pedestrians, the same layer can be modified by multiple detection heads simultaneously based on hierarchical features extracted at each layer. Lack of exhaustive work in this avenue indicates that a lot remains in this direction for successful utilization. We make detailed observations over this in chapter 4.

3.5.4 Anchor Parameters

Anchor boxes have two general parameters – a) aspect-ratio and b) scale. Anchors are created by transforming a base template which is usually a square of size ($height \times width = p \times p$). For an aspect ratio (*width/height*) α and scale s, the anchor is created as follows –

1. **Step 1** : Compute the anchor area $AR = p^2$.
2. **Step 2** : Compute the width of transformed anchor as $w_s = \sqrt{AR \times \alpha}$.
3. **Step 3** : Compute the height of the transformed anchor as $h_s = \frac{\sqrt{AR}}{\sqrt{\alpha}}$. Note that this ensures the area of the transformed bounding box is same as the area of the base anchor.
4. **Step 4** : Rescale the anchor by $w_s \times s$ and $h_s \times s$.

In the following subsections we look at the impact of these anchor parameters.

3.5.4.1 Anchor aspect ratios

In the caltech dataset, the mode of aspect ratios (*width/height*) of pedestrians is 0.41. In citypersons [179], annotations have been carried out while ensuring that bounding box aspect ratios are constant to 0.41. Moreover pedestrians in urban environments generally appear with specific profiles of aspect ratios. This information can be infused in the design process of anchors.

Table 3.7, shows the impact of setting the aspect ratios of anchors on results for caltech dataset. The performance figures show around 1-2% of improvement in both Faster-RCNN [130] and SSD [102].

Setting the aspect ratio to match the dataset allows for a richer set of anchors detected as *positive*. This ensures that the training is more diverse, because each anchor will now observe a specific view of a pedestrian. The impact however is also dependent on the quality of the *base network*. For a network like VGG16, the performance improvements

	Aspect Ratios	Reasonable	Near-Scale	Medium-Scale
Faster-RCNN	0.5, 0.6, 1, 2, 3	11.15%	3%	45.92%
	0.41	**10.05%**	**2.6%**	**44.49%**
SSD	0.5, 0.6, 1, 2, 3	12.01%	3.02%	46.62%
	0.41	**9.9%**	**2.24%**	**42.34%**

Table 3.7: Impact of anchor aspect ratio on Miss-Rate performance on caltech dataset with InceptionV2 as base network.

are generally higher. More refined base networks are better feature extractors and thereby can compensate for the lack of a very rich set of *positive* anchors.

It can also be observed that in table 3.7, we have not reported performance figures with respect to pedestrian occlusion. Occlusion of pedestrians is not impacted by anchor parameters in a clear manner. Solving occlusion problem is related to computing better feature representations. Anchors on the other hand are placeholders which serve to delineate different sub-regions of an image from one other.

3.5.4.2 Anchor scales

Understanding the heuristics to set the anchor scales is hard due to a number of reasons. Setting scales over a large range with the hope of covering all possible pedestrian heights can result in a large number of *negative* anchors. It also increases the computational cost. In SDS-RCNN [14], anchor scales are set to match the 25-350 pixel range in which majority of pedestrians in the caltech dataset occour. Using the same large range in standard Faster-RCNN [45], results in an increase of miss-rate by nearly 1.8%. This seems to be on account of semantic features infused in [14]. Unlike SDS-RCNN [14], Faster-RCNN does not utilize semantic segmentation as additional information. As a result, when large number of closely set anchor scales are used, there is an equally large number of negative anchors. This results in the classical problem of class imbalance which lowers the detection performance.

	Anchor Scales	Reasonable	Near-Scale	Medium-Scale
Faster-RCNN	0.5, 1, 2, 4	**10.05%**	**2.6%**	**44.49%**
	0.5 to 4 with step 0.1	11.3%	3.4%	50.2%
SSD	0.5, 1, 2, 4	**9.9%**	**2.24%**	**42.34%**
	0.5 to 4 with step 0.1	12.2%	2.8%	45.13%

Table 3.8: Effect of anchor scales on miss-rate performance on caltech-reasonable dataset. The base anchor size is 128×128. The base network is InceptionV2. The aspect ratio of anchors is 0.41.

Negative anchors may also overlap with pedestrian groundtruth; albeit with a small

overlap. A robust classifier is needed to handle the presence of large number of *negative* anchors. This is especially true when a very large range of anchor scales is used as a design parameter. Moreover, we feel it is unnecessary to use a large range of anchor scales. The regression layer for bounding box estimation, used in [55, 102, 47, 14] is able to handle aberrations due to imperfect overlap of anchors, as long as they are marked as *positive*. Table 3.8, shows the results of setting very fine-grained anchor scales. The performance degrades for both Faster-RCNN and SSD. For SSD the performance degradation is more pronounced. We reckon this difference in degradation further points towards the feature handling differences between Faster-RCNN and SSD as outlined in chapter 2.

3.5.5 Loss Function

Till date, the most remarkable improvement in detection performance on caltech dataset has come about by infusing semantic segmentation features in the detection pipeline. In SDS-RCNN [14], the groundtruth bounding box is assumed to represent a *pseudo*-segmentation mask. The logistic loss over an anchor with respect to this mask is infused in the loss function of the RPN as well as the BCN [14]. The qualitative and quantitative results show that this incorporation has led to an improvement by a margin of nearly 2% over RPN-BF [177].

There are other potential loss function refinements which have been introduced. One popular example is the focal loss of attention [98].

During training, for every anchor box during training, a softmax score representing its class can be obtained. Given that the true label for the anchor is t, we use the symbol p_t for the softmax score of being class t. Given this information, in [98], the focal loss for the anchor is defined as

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}log(p_t) \tag{3.2}$$

In equation 3.2, the term $-log(p_t)$, is the standard cross-entropy, which has been multiplied by a factor to generate the focal loss. As the value of p_t decreases, $FL(p_t)$, increases. Therefore, focal loss aims to force the optimizer to focus on difficult cases which are hard to classify. The values of γ and α are empirically chosen.

Table 3.9 shows the results of applying focal loss of attention to the existing loss functions of Faster-RCNN and SSD. Focal loss introduces a minor improvement in performance. These improvements are in line with the average quantitative improvements reported by the authors in [98]. However this improvement does not match the improvements introduced by the infusion of segmentation information by SDS-RCNN [14]. Segmentation information is much more specific and semantic in nature.

In addition to focal loss which is meant as a loss for classification problems, repulsion loss [166] has also been proposed which is meant for regression problems. Repulsion loss

	Loss function	Miss-Rate
Faster-RCNN	No Focal loss	10.05%
	Focal Loss	**9.97%**
SSD	No Focal Loss	9.9%
	Focal Loss	**9.7%**

Table 3.9: Impact of focal loss inclusion in the loss functions of Faster-RCNN and SSD. We choose $\alpha = 0.5$ and $\gamma = 2$ as in [98]. Anchor parameters are chosen as delivering best results according to tables 3.7 and 3.8. InceptionV2 is the base network.

	Loss function	Miss-Rate
Faster-RCNN	No repulsion loss	10.05%
	repulsion Loss	**8.72%**

Table 3.10: Impact of repulsion loss inclusion in the loss function of Faster-RCNN. Anchor parameters are chosen as delivering best results according to tables 3.7 and 3.8. InceptionV2 is the base network.

has been described in detail in chapter 2. In table 3.10, we outline the impact of repulsion loss on Faster-RCNN with InceptionV3 as the base network. We see a major performance boost of $\sim 2\%$ with the inclusion of repulsion loss. Localization is an important basis for our evaluation metrics. Repulsion loss hence is an important recently proposed loss function which serves to boost our quantitative performance on account of its focus on localization accuracy.

3.5.6 Role of Dataset Resolution in Pedestrian Detection

The resolution of a dataset is a key element in deep learning systems. Resolution differences between training and testing datasets often translates to degraded performance. This is especially true when the testing dataset is of a lower resolution.

When developing a system for use with a multitude of cameras with varying resolutions, it is important to do training or fine-tuning so as to minimize degradations caused by dataset resolution differences. Most pre-trained versions of pedestrian detection systems are trained on one specific dataset (*usually caltech pedestrian dataset*). To understand the role of dataset resolution we performed two complementary experiments – *E1* and *E2*.

In *E1*, we trained Faster-RCNN [45] with VGG16 as the base network on Caltech training set (*low resolution*). We then fine-tuned the trained system on the BDD100K training set (*high resolution*). The final system was then evaluated on two datasets – Caltech testing set and BDD100K validation set.

In *E2*, we changed the order of datasets during training. Thus, initial training was done on BDD100K training set followed by fine-tuning using Caltech training set. The

	Before Fine-tuning		After Fine-tuning		Difference (After - Before)	
	Caltech (Testing)	BDD100K (Testing)	Caltech (Testing)	BDD100K (Testing)	Caltech (Testing	BDD100K (Testing)
E1 (Caltech followed by BDD100K)	14.62	29.1	14.87	26.22	**+0.25**	**-2.88**
E2 (BDD100K followed by Caltech	15.14	27.33	13.43	27.10	**-1.71**	**-0.23**

Table 3.11: Impact of relative ordering of datasets of different resolutions when doing training and fine-tuning. It is better to first train a dataset with a high-resolution dataset and then fine-tuning it with a low resolution dataset. Caltech refers to caltech reasonable. BDD100K refers to pedestrians with heights$> 50px$. All figures are LAMR values

evaluations were then done as in *E1*.

In table 3.11, we compile the results of how the order of datasets in *E1* and *E2* impact the performance. The LAMR values were recorded for both *E1* and *E2* before and after fine-tuning for both the datasets. In *E1*, where a high resolution dataset was used after a low-resolution dataset, the performance on the low-resolution dataset (i.e caltech) degraded by a margin of 0.25%. In experiment *E2*, the performance on caltech however improved by about 1.71%. For BDD100K, which is a high-resolution dataset, the performance improved in the case of both *E1* and *E2*.

From the results in table 3.11, a number of important inferences can be made. A high-resolution dataset is more immune to degradations caused by fine-tuning when a low-resolution dataset is used. In *E2*, where fine-tuning was done using caltech, the performance on BDD100K, still improved albeit by a narrow margin of 0.23%. Caltech on the other hand, suffers degradation in *E1*, when BDD100K is used for fine-tuning. For the purpose of maintaining the performance, it is important thereby to use a high-resolution dataset for initial training, followed by fine-tuning on a low-resolution dataset.

3.6 What is the best pedestrian detector

In this section, we reflect upon our observations from the preceding sections and outline some guidelines which may prove to be helpful to a system designer.

3.6.1 *Base Network Architecture*

The most common use-case is to develop a detector which performs well on a specific custom dataset. At the outset, it is important to start with a deeper base network with architectural characteristics suitable for efficient multi-scale processing such as InceptionV3 [147]. Modern CNN architectures like InceptionV3 are deep and perform better, but usually have fewer parameters than an equivalent extension of classical architectures like VGG16 [141]. Therefore benefits can be elicited from them in terms of detection performance as well as runtime performance. It is well understood that the feature expressive

power of a CNN increases exponentially with depth [122]. Thus, a deeper network is more effective in modeling an object of interest in a more difficult environment. We observe from tables 3.1 through 3.4 that a simple change of base network architecture can provide improvements of upto 1%. While the selection of a base network architecture is not the only crucial design decision, it can be a good starting point for the design problem. There are several other interesting architectures and our analysis does not cover them – e.g Densenet [73] and Xception [25]. These architectures perform well and have fewer parameters than their counterparts in ResNet family of architectures. Their analysis and usage can be further useful.

3.6.2 Feature Map Resolution

Most network architectures employ pooling operations which increases the output stride of a feature map. While the à trous trick has been used with notable performance improvements, more work is needed in this direction. It is not always clear that which layers the à trous trick be applied to. Deciding the *rate* value is also a matter of experimentation at this stage. By applying à trous trick to a subset of convolutional layers, there are multiple ways to produce a feature map with a specific output stride. There needs to be greater work in this direction to fully exploit the à trous trick in detection problems. However, from the application of à trous trick in other works such as [177], and through our experiments we conclude that it may prove worthwhile to experiment with it, especially for small-scale pedestrians.

Simultaneous usage of multiple convolutional layers has been successfully employed in several works [15, 158]. While this approach is interesting, more work is needed to determine its impact on the feature expressiveness of a deep neural network.

3.6.3 Anchor Design

The choice of dataset specific anchor parameters is known to improve the performance as shown in RPN-BF. We have given quantitative figures for Faster-RCNN and SSD for the same in tables 3.7 and 3.8. It is notable that most detection techniques proposed following RPN-BF, such as SDS-RCNN [14], follow this direction. The limitation of this guideline is that only aspect-ratio can be appropriately selected by this approach. To the best of our knowledge, there is no quantitative or heuristic estimate for the fitting capacity of a bounding box regressor. Even a scale-imperfect anchor can be appropriately regressed by a regressor. Therefore it appears overwork to carefully study the scale-characteristics of a dataset to formulate the anchor scale parameters. Another latent difficulty therein is that, object scales are much more malleable than aspect-ratios. More work in these

directions can be helpful to suggest a more systematic way of deriving anchor parameters from dataset statistics.

3.6.4 Semantic Features

The current state-of-art detector – SDS-RCNN [14], makes use of pseudo-segmentation masks to improve proposal quality as well as final bounding box estimation. As shown in [14], the infusion of these weak segmentation based features is the most critical component of SDS-RCNN bringing about 3% of improvement on caltech-reasonable dataset. Therefore it is advisable to resort to infusing semantic features in a detection system design. Further work on better ways of infusing semantic features in different domains such as video streams can be of further interest to the research community.

3.6.5 Classifier Selection

Classifiers are a very critical part of system design. In [178], an exhaustive analysis of false-positive sources on Caltech [33] and KITTI [52] datasets reveals primarily two culprits – a) Double detections and b) vertical structures. Out of these vertical structures' misclassification signals at the inefficiency of existing fully-connected softmax based classifiers. In most cases, these vertical structures correspond to street-lamps, or doors of stores or even trees on sidewalks. As the visible details in pedestrians decrease (*small-scale* pedestrians), these problems become more pronounced. The improvements obtained by RPN-BF over Faster-RCNN by a change of classifier suggests that it may be more prudent to try out new classifiers. Pedestrians have a rather high intra-class variance, which gets more pronounced as the scale of a pedestrian increases. Decision trees are known to be good at handling intra-class variance. They are however prone to overfitting in the lack of careful selection of parameters. Often this is countered through pruning [38]. Therefore extensive hyperparameter optimization is needed when working with tree-based classifiers in pedestrian detection. The integration of tree-based learning in deep learning has been relatively poor. Recently several works [186, 184, 7, 84, 138] have appeared on introducing random forests as potential classifiers for different applications. Tree based classifiers are of specific interest because they are better at handling intra-class variance – a major component of pedestrian detection. Joint training of a tree-based classifier with existing pedestrian detection framework may also be of specific interest.

3.6.6 Post-Processing

As highlighted before, double detections are a principle source of false positives [178]. Multiple detections sharing the same region of space is a well known phenomenon since

the viola-jones detector [161]. These come about as a result of multiple sliding windows sharing a potential region of interest. Non-maximal suppression (NMS) is used to remove such ambiguous cases of multiple detections. NMS partitions all the bounding boxes into disjoint sets. Bounding boxes in each set are either averaged to obtain the final bounding box, else the box with maximum score is used as in the dalal-triggs detector [31]. This partitioning happens by the sorting of confidence scores of all bounding boxes. For each bounding box, other bounding boxes overlapping with it above a threshold are suppressed. The choice of this threshold is based on experiments and is non-intuitive. For cases involving closely separated objects, NMS generally delivers suboptimal results [12]. Recently new variations of NMS such as soft-NMS [12] and a trainable version [70] have been proposed. In our experiments we have used the traditional version of NMS. The present results are generally mixed for the new variations, with soft-NMS reporting an improvement of 3% in mAP of the 'person' class in Pascal VOC dataset, while the trainable version reporting a 0.4% degradation for the same category in MSCOCO dataset. It would be of interest to work on this front. As pointed out in [178], solution to this problem can significantly boost the performance of current pedestrian detection systems.

3.7 Conclusion

Past research in pedestrian detection has been primarily followed with respect to benchmarking on very specific public dataset configurations (e.g- *caltech reasonable* [34], with predefined constraints on heights and occlusion). In contrast, in industrial applications, pedestrian detection systems are applied with live video stream as input, where well-defined constraints like those of *caltech-reasonable* are not applicable. As such, live video stream in industrial applications present a more general set of scenarios which are not covered by the principal public datasets for pedestrian detection (e.g-*caltech* [34]). In this chapter we have outlined a set of potential guidelines for improving the performance of current pedestrian detection systems. These guidelines can benefit the use-cases of – a) Benchmarking on existing public datasets and, b) Pedestrian detection in live video stream inputs in industrial applications.

With respect to benchmarking on existing public datasets, evaluation protocols are often dataset specific, thereby making thorough cross-dataset analysis difficult. We have used a consistent protocol (see section 3.4) for our quantitative analysis, thereby providing a consistent estimate of the performance of existing techniques across pedestrian *heights* and *occlusion*. This analysis shows (tables 3.1 through 3.4) that existing techniques perform very well for detecting *near-scale* (i.e unoccluded pedestrians close to the camera). For *near-scale* pedestrians the average miss-rate (see section 2.5) is less than 4%, and with the present state-of-art technique approaches near perfection (miss rate of 0.04%).

The performance for *medium-scale* pedestrians however shows a major disparity and the average miss-rate hovering in the range of $40\% - 50\%$. This exhibits that more work is needed to improve the robustness of pedestrian detection systems for handling inputs a wide diversity of pedestrian heights. A similar trend exists with respect to *occlusion*, where heavily occluded ($35 - 80\%$) pedestrians are often $30 - 40\%$ harder to detect than less occluded ($< 35\%$) pedestrians. Combining these observations together, it is also seen that on the *caltech-reasonable* subset, while performance has improved to 7.6%, it is well below the average human-level performance of 0.88% [178]. Our quantitative analysis encompassing the selection of *base architecture*, *anchor design*, *use of multiple layers* and use of *à-trous convolution*, provide guidelines which serve to improve the performance of existing pedestrian detection systems. These guidelines applied to a novel pedestrian detection system construction can serve to provide a good starting point from where more advancements are possible. More notably, we show that with dataset-specific anchor aspect ratios miss-rates for medium-scale pedestrian detection can be improved by $4 - 5\%$.

Live video stream inputs encompass the aforementioned limitations of pedestrian detection systems, and add additional challenges related to cross-dataset generalization, varying resolutions, varying illumination and weather conditions. We have used the recently released BDD100K [174] dataset which more adequately models these conditions (e.g-weather and illumination) than previous datasets such as caltech. Our experiments (table 3.11) with caltech and BDD100K datasets enable a useful guideline about cross-dataset generalization. More specifically, initial training on *high resolution* dataset followed by fine-tuning on a *low-resolution* dataset preserves greater cross-dataset generalization than the other way around.

Several important research questions have also turned up during the course of this analysis. In our work, we have indicated various relevant research directions which serve to provide good grounds for further developments in pedestrian detection research. As an example, our experiments with setting fine-spaced anchor scales (table 3.8) shows that it does not necessarily improve the performance. We have instead shown that it is related to the robustness of the classifier and use of alternative classifier choices alongwith the analysis of their performance is more desirable. Even methods with previously demonstrated improvements such as à-trous convolution can benefit from more analysis about its applications in pedestrian detection.

Runtime performance is another relevant element, especially in industrial applications such as *autonomous vehicles*, due to their safety-driven requirements. We have not made an exhuastive analysis of runtime performance. Runtime performance can be affected by multiple optimizations in hardware as well as underlying software implementations and is better suited for a separate study. An important connection of runtime performance with deep learning comes in the form of quantization of existing networks. Quantization refers

to converting the weights of a pre-trained system from $32-bit$ floating point precision to a $16-bit$ or $8-bit$ floating point precision. These quantizations reduce the inference time, thereby improving runtime performance. However, this trades-off with the actual performance figure. Moreover, the improvements in runtime also depend upon the specifics of a hardware used. An exhaustive analysis of these runtime features can be of significant interest especially for industrial or other real-time applications.

Chapter 4

A Multiple layer RPN approach to Pedestrian Detection

Contents

4.1 Introduction

In this chapter, we consider the problem of *scale* and *occlusion* handling for pedestrian detection. Successful detection under occlusion implies ability to detect from a potentially small portion of a pedestrian being visible. Successful detection of small-scale pedestrians implies the ability to detect when a pedestrian is contained in a small pixel area in an image. Hence, the two problems are related in a complementary manner. As seen in chapter 1, these problems are quite well aligned with the recent interests in autonomous vehicles. Successful detection of far-scale pedestrians can assist the vehicle in making safety maneuvers well ahead in time, thereby promoting a safer traffic environment. The same is true for surveillance systems in high security environments like airports and ports [40].

We elaborate a pedestrian detection system for handling the aforementioned problems. We consider Faster-RCNN based detection framework as a basis for this design. This is motivated by the fact that Faster-RCNN framework represents the crucial components of a broad family of object detection techniques including RPN-BF [177] and SDS-RCNN [14], which are used for pedestrian detection. For our design, we show that lower layers detect small-scale and occluded pedestrians better than later layers; which are better at detecting large-scale and unoccluded pedestrians. We take cue from these conclusions and present a system design which performs at par with the state-of-art pedestrian detection systems.

4.2 Hierarchical nature of CNN features

CNNs are cascade structures and hence have an inherent hierarchy spread across its layers. This hierarchy is apparent in the features learnt by CNNs once trained. The nature of CNN features are analyzed in a multitude of ways such as deconvolutional layers [176] and t-SNE visualization [35]. Another way which is more commonly employed to understand the global nature of different convolutional layers is to visualize the feature maps produced by different layers. Such a general visualization confirms that the nature of features learnt by different convolutional layers is very different. Figure 4.1 shows the feature maps generated by VGG-16 [140] network pre-trained on ILSVRC-2012 dataset. The top-row shows the feautre maps corresponding to 4 randomly chosen filters in the first layer of VGG-16, while the bottom row shows the same for 4 randomly chosen filters in the last layer of VGG-16. The features in the top-row appear to capture low-level details such as edges, and contrast. In comparison, it is difficult to interpret the nature of details in the bottom row of figure 4.1. This observation has a long-standing precedence since the first large-scale work on training a CNN [86]. This phenomenon is apparent in other deep learning mechanisms such as RBMs, where lower-layers capture low-level details

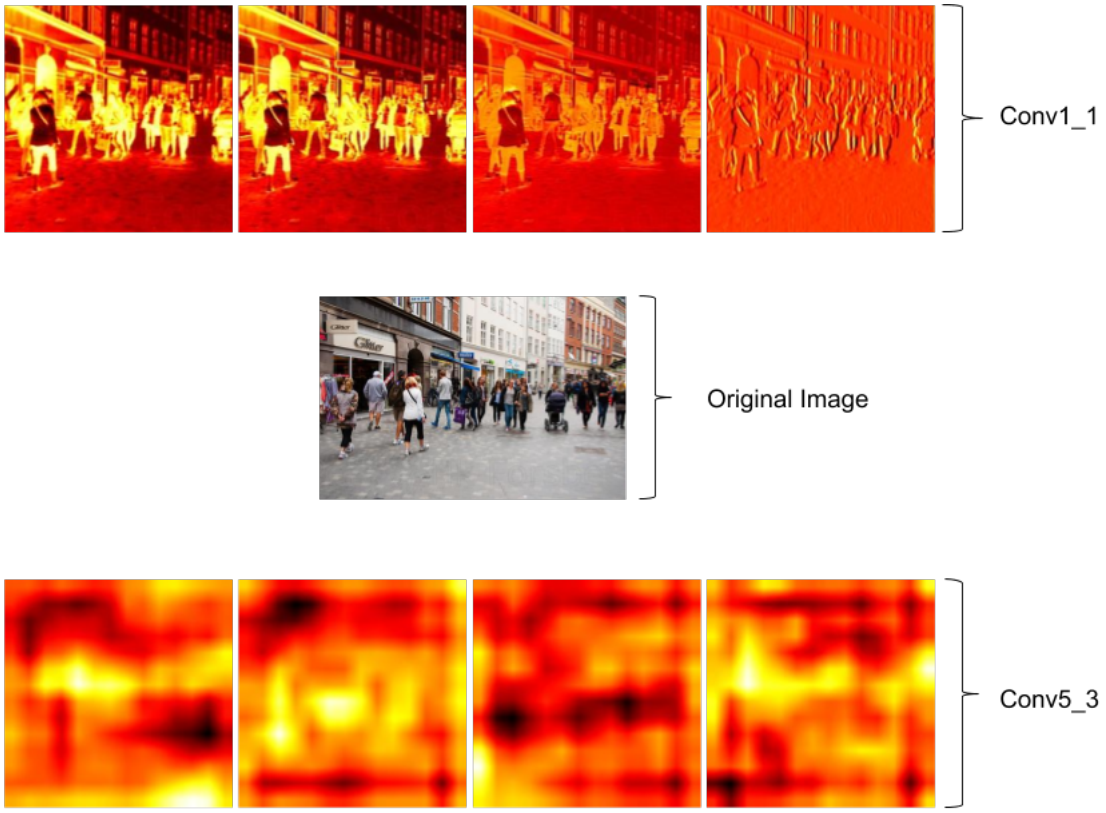


Figure 4.1: Visualization of feature maps from two different layers in VGG-16 network trained on ILSVRC-2012 dataset.

and higher layers successively capture high-level details such as faces and complex curves [92].

This hierarchical nature of CNNs evokes an interesting question with respect to occlusion and scale handling : *What is the nature of features captured by different CNN layers across scales and occlusion levels ?*

In the subsequent sections we attempt to obtain a quantitative estimate of the effectiveness of different convolutional layers in handling pedestrians across *scales* and *occlusion* levels. These estimates allow us to forge a system which is capable of simultaneously utilizing multiple convolutional layers to detect pedestrians across scales and occlusion levels.

4.3 Datasets and Evaluation Metrics

4.3.1 Datasets

In this work we utilize the Caltech pedestrian dataset [34] for benchmarking. Availability of large number of benchmarks [177, 14, 180] on it, makes it a vital dataset for studying our work and its impact.

The resolution of Caltech dataset images is 640×480 and has been captured from a moving vehicle without any corrections for vehicle pitching [34]. The original annotations in Caltech suffered from alignment problems [47]. We utilize the improved annotations [47] in our work for benchmarking. Following [47] the training images taken at interval of 3 frames (caltech10x-train) are used for training. The testing images taken at interval of 30 frames (caltech1x-test) are used for testing. We have also evaluated our method on the reasonable subset (height > 50px and occlusion < 0.35).

Evaluation on the reasonable subset involves evaluating on the reasonable annotations from the caltech1x-test set. New annotations of [47] are available only for these subsets (caltech10x and caltech1x).

4.3.2 Evaluation Metrics

We use log-average miss rate as proposed in [34] for the evaluation of our proposed approach. This average miss-rate is computed for a range of *false positives per image* (FPPI) in $[10^{-2}, 10^{0}]$.

4.4 Layer-wise analysis of CNN layers' effect on scale and occlusion

As features pass between convolutional layers separated by pooling layers, they undergo feature and scale transformation. Intuitively, it suggests that small-scale and partially occluded pedestrians can be captured well by the lower CNN layers. To look into this effect quantitatively, we design a system as exhibited in Figure 4.2. For our study, we use the data flow depicted in figure 4.2 using dashed green lines, i.e, no concatenation is performed. It uses VGG16 [140] as a base network which is also used in Faster-RCNN [45]. We model a detection framework after Faster-RCNN, by extracting features from a subset of convolutional layers. VGG16 is divided into convolutional blocks. Within each block, feature maps are of the same shape and they are transformed into half their size by the pooling layer between consecutive blocks [140]. We extract features (f_i^{conv}) from the last convolutional layer within each block (thus $i \in \{1, 2..5\}$. Each f_i^{conv} feeds into a separate region proposal network (RPN) [45]. A RPN passes the feature map through one or more

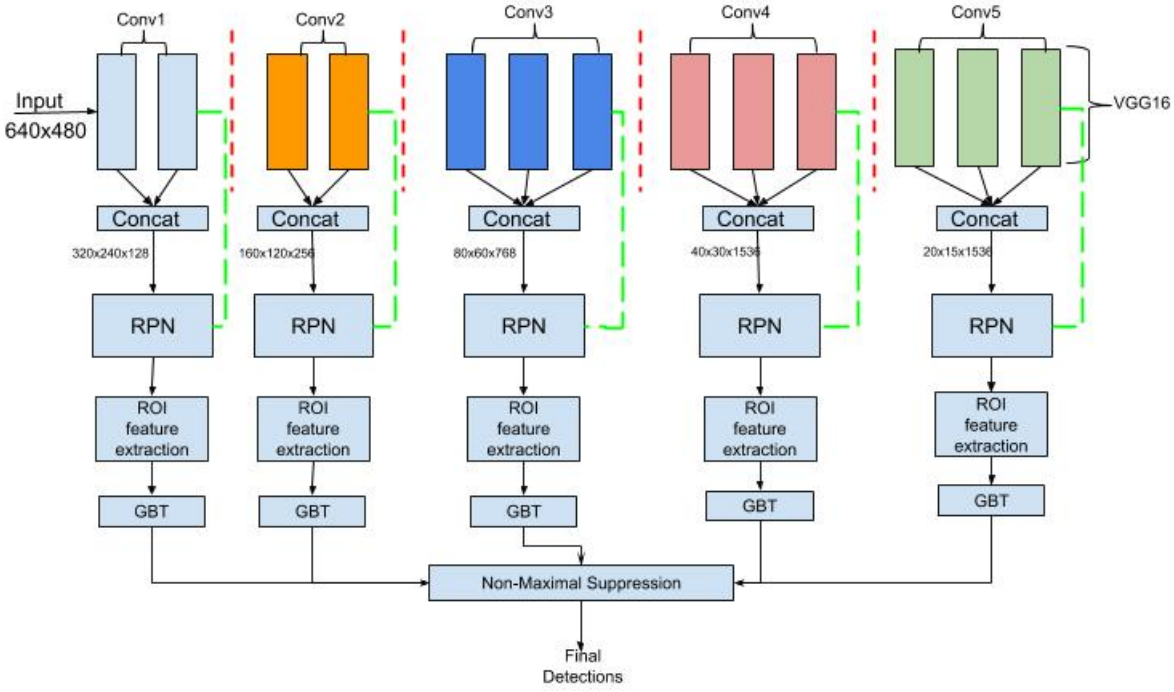


Figure 4.2: Block Diagram of the proposed pedestrian detection system. The data flow for the study in section 4.4, is indicated using dashed green lines. We later improve upon this and the final system's data flow is shown in solid black arrows. The red dashed lines in the diagram refer to the location of pooling layers in VGG16 [140], where the feature map changes size.

convolutional layers resulting in a feature map (f_{map}^{RPN}). It then slides a set of template bounding boxes (*anchors*) over f_{map}^{RPN}. An anchor is a template bounding box which is translated through a feature map. It is centered at each pixel and for the resulting area of the feature map it determines if that area contains a region of interest [45]. Anchors' intersection over union (IoU) with groundtruth bounding boxes is used to determine the presence of positive proposals during training (IoU$> 0.7 \implies$ +ve and IoU$< 0.3 \implies$ -ve, $0.3 <$IoU< 0.7 don't contribute to RPN training) [45]. Anchors are created for varying scale and aspect ratios. Due to the upright nature of pedestrians, pedestrian specific systems [177, 14] consider anchors encompassing several scales but limited to a constant aspect ratio of 0.41 (*mode of aspect ratios of pedestrians in caltech dataset*). Using two sibling fully connected layers, RPN performs two tasks on these anchors – a) proposal classification and b) proposal bounding box regression. In Faster-RCNN, positively classified proposals are further refined by a set of two sibling fully-connected layers for final object-level classification and regression. In a related work of RPN-BF [177], the authors show an improved performance using boosted forests. We use gradient boosted trees as classifiers, to do the final object level classification. At the end we use soft non-maximal

suppression [41] for getting final detections

In our experiments we use one convolutional layer (*kernel size*: 3×3, *stride*: 1, *padding*: 1, *num-filters*: 512) in each RPN. More convolutional layers for early layer feature maps may be beneficial but they require much GPU memory. Our base VGG16 network is pre-trained on the imagenet dataset. We train the design of figure 1, in two stages. In the first stage, we train the RPNs, using stochastic gradient descent with the same loss function (eqn 4.1) as proposed in [45].

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \frac{\lambda}{N_{reg}}\sum_i L_{reg}(t_i,t_i^*)p_i^* \tag{4.1}$$

In equation 4.1, quantities with a *star* (p_i^*, t_i^*) refer to groundtruth and those without it refer to the anchor which overlapped them with an IoU>0.7. $\{p_i\}$ is the label of i^{th} anchor (+ve or -ve), $\{t_i\}$ is the vector of bounding box coordinates of i^{th} anchor, L_{cls} is the cross-entropy loss, L_{reg} is the smoothed-L1 loss, λ is a scalar constant (set to 1.0 in our implementation). N_{cls} is minibatch size and N_{reg} is total number of anchors. Following the RPN training, we select the top 1500 ROIs based on RPN scores and extract RPN features from them. These serve as input to gradient boosted trees (GBT). We train the GBTs using the XGBoost framework. We set the maximum depth of each tree as 6, and the maximum number of trees as 1024. GBTs are used only for classification. Coordinates predicted by the regression layer of RPN are not further regressed by GBTs.

To evaluate the impact of the convolutional layer connected to i^{th} RPN (RPN_i), we detach all the other RPNs from the non-maximal suppression. We use the caltech10x training set [47] for training and caltech1x-testing set [47]. For occlusion studies we also use the complete caltech test dataset for testing using old annotations due to wider range of occlusion levels present in them.

4.4.1 Effect of CNN layers on scale and occlusion based detection

Pedestrian height(in pixels)	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Fused
>80	4.83	4.97	3.88	2.17	**2.15**	3.27
$50-80$	**9.17**	9.54	9.40	10.48	10.68	10.43
All (height >50)	12.83	13.12	11.34	11.2	**10.95**	10.16

Table 4.1: Log-Averaged miss rate for pedestrians of different heights by different layers in Caltech-reasonable (*test*) dataset.

Occlusion(% occlusion by area)	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Fused
0 (No occlusion)	10.6	10.4	9.7	9.68	**9.64**	9.79
$1-35$	**22.3**	24.75	25.2	25.5	26.1	25.3
$65-80$	**67.2**	69.9	71.2	73.5	76	75.4
All ($0-80\%$ occlusion)	**68.4**	70.1	72.7	75	77.2	76.3

Table 4.2: Log-Averaged miss rate for varying occlusion levels by different layers in Caltech-complete (test) dataset. For the Caltech-complete we have used the old annotations (pedestrian height > 50 pixels.)

Table 4.1 shows the log-average miss-rate for different pedestrian heights in the caltech-reasonable test-set. The column titled "fused", refers to the configuration with all RPNs contributing to NMS. For larger heights, better performance is achieved by the later layers, while the opposite is true for smaller heights. Small-scale pedestrians are mainly discriminated by their contours and other low-level features. Near-scale pedestrians have greater amount of visual detail, which varies with context such as clothing. Thus, their detection requires more semantic features which are captured by higher CNN layers. Moreover, due to a sequence of pooling layers, feature resolution decreases with CNN layer depth. For small-scale pedestrians this leads to their features reduced to sub-pixel accuracy. This is another factor behind a lower accuracy of detection of small-scale pedestrians in later CNN layers.
Table 4.2, shows the performance of different layers based on pedestrian occlusion. We test the system on the complete caltech dataset. To keep the study concise and tractable, we limit ourselves to pedestrians with a minimum height of 50 pixels. Extremely occluded pedestrians (*occlusion* $>$ *80%*) often require specialized approaches and we keep them out of our study. Table 4.2, points to the conclusion that lower CNN layers capture occlusion better than higher layers. This is expected as high-level of occlusions imply a less amount of visible pedestrian area, which is compromised by pooling layers. In the caltech dataset, high levels of occlusion are primarily observed in medium-to-small scale pedestrians [34]. We consider the trends of tables 4.1 and 4.2, fairly indicative and frame our design from these conclusions.

4.4.2 Pedestrian Detection System Design

Following the inferences from section 4.4, we design our pedestrian detection system to leverage multiple convolutional layers explicitly and simultanously. Our system design is similar to the one we used in our study (figure 4.2), with some modifications. The system level modifications include –

1. Concatenating all convolutional layers within a block before feeding the correspond-

ing RPN.

2. Using a dense system of pedestrian specific anchors.

3. Use of a modified loss function for RPN.

In addition, we experiment with two fusion approaches to utilizing multiple convolutional layers and determine the better of the two.

4.4.2.1 Feature Map Concatenation

As shown in figure 4.2, we concatenate feature maps from convolutional layers in each block of VGG16. VGG16 feature maps in each block have the same dimensions. Thus they can be concatenated without any overhead of resizing. Due to hierarchical character of CNN features, this also enhances the feature diversity for the input to the RPN.

4.4.2.2 Anchor Design

Layer	Scales
Conv1	[12, 512, 32]
Conv2	[10, 256, 32]
Conv3	[8,128, 32]
Conv4	[8, 128, 16]
Conv5	[8, 128,16]

Table 4.3: Anchor scales chosen for different layers. The notation [A,B,C] in the second column refers to minimum scale as A, maximum scale as B with a step-size of C (all in pixels).

Some preliminary bounds are available to determine the minimum detectable size of an object for an anchor of identical scale as the object [42]. However their assumption requires us to have cover all possible scales of objects which is impractical for large datasets like caltech. To mitigate the risk of choosing a set of fixed anchors, we consider a more dense set of anchors (*large range of scales with small step sizes*) as compared to [177] and [14]. Our anchor construction is detailed in table 4.3. All our anchors have a fixed aspect ratio of 0.41 (*width/height*). Anchors which fall outside a feature map perimeter are eliminated from computations. A dense set of anchors though increasing computational complexity, helps in improving detection of positive (*pedestrian*) region proposals.

4.4.2.3 Loss Function

The loss function for RPNs in [45, 177], do not take into account the object dimensions. In our work we give weight to the observation that pedestrians can appear at a wide variety

of scales on account of their varying distances from a camera. To this end, we modify the RPN loss function as follows–

$$L(\{p_i\},\{t_i\}) \triangleq \frac{1}{N_i}\sum_i [I_i^1\{\frac{L_{cls}(p_i,p_i^*)+\zeta L_{reg}(t_i,t_i^*)}{f(h_i)+\varepsilon}\}+ \eta I_i^0 L_{cls}(p_i,p_i^*)] \tag{4.2}$$

In equation 4.2, I_i^1 is an indicator function which is 1 if the i^{th} anchor is a pedestrian candidate proposal and 0 otherwise. I_i^0 is an indicator function which is 1 if the i^{th} anchor is a non-pedestrian candidate proposal and 0 otherwise. L_{cls} is the log-loss over two classes (*pedestrian* vs. *non-pedestrian*). L_{reg} is the regression loss and it is clear from equation 4.2, that it is activated only for positive proposals. ζ and η are scalar constants. ζ intuitively denotes the balance between the classification loss and the regression loss. η denotes the balance between the true-positive(TP) classification and true-negative(TN) classification. In our experiments we set both ζ and η to 1. We normalize our loss function by the total number of anchors in the RPN (N_i). Other symbols (p_i,p_i^*,t_i,t_i^*) are the same as in equation 4.1. In our loss function $f(h_i)$ denotes the running accuracy of correct RPN classification for a pedestrian of height h_i. Running accuracy is a metric which keeps an updated track of the accuracy through the iterations of training. The running accuracy is computed as follows-

- **Step 1:** Get all unique heights $H \triangleq \{h_1, h_2, ..., h_N\}$ in the dataset.
- **Step 2:** Initialize a dictionary D with elements of H as keys. For each h_i in H, $D[h_i] \triangleq (CC_i, TC_i)$. Here, $(.,.)$ is the notation for a tuple of two numbers. CC_i, is the cumulative count (*across iterations*) of correct RPN classifications for a pedestrian of height h_i. TC_i is the cumulative count (*across iterations*) of the number of times, a positive proposal overlapping with a pedestrian of height h_i has passed through the proposal classification layer.
- **Step 3:** The running accuracy $f(h_i)$ is then defined as : $f(h_i) \triangleq \frac{CC_i}{TC_i}$

Running accuracy is just an online version of accuracy of prediction. The parameter ε is a small scalar constant to avoid infinity, when $f(h_i) = 0$.

The incorporation of $f(h_i)$, makes the loss function in equation 4.2, scale aware. For pedestrians of certain heights, which are not getting detected by RPN, the loss function is penalized. For every unique height of pedestrians, $f(h_i)$ is maintained. During the training process, feature weights in the network are constantly updated. Some of these updates may improve detections of certain heights while deteriorating other pedestrian heights. Keeping a continuous track of accuracy during training helps in stablizing the network towards a balance of detection of all pedestrian heights. This is in stark contrast

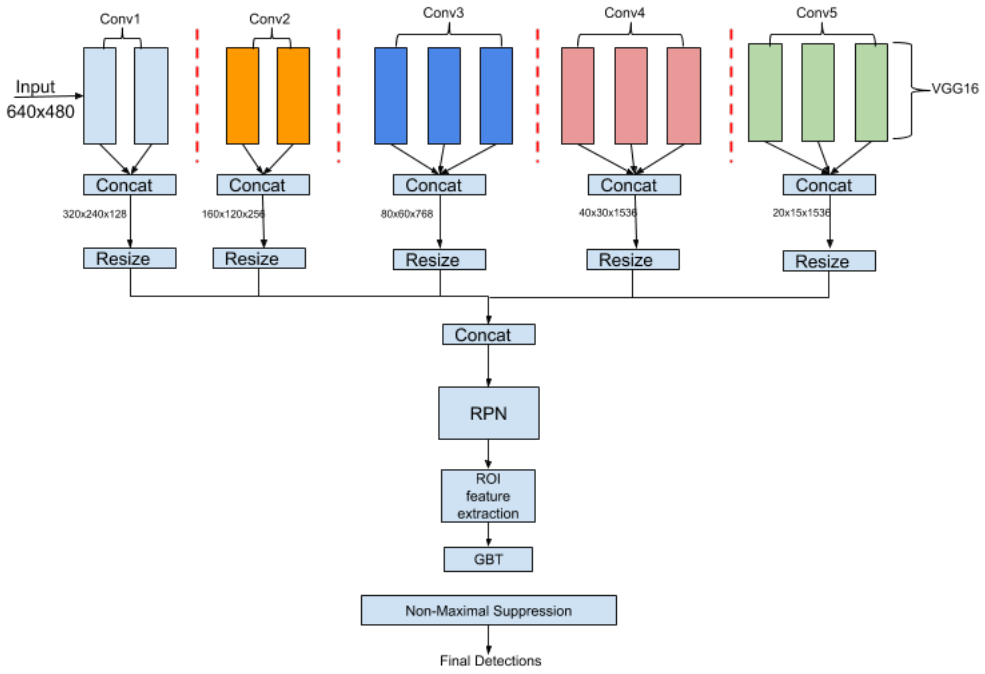


Figure 4.3: The proposed pedestrian detection system with early fusion.

to [45, 177], where the network is not made to give any weightage to the pedestrian height.

For bounding box regression we use the same smoothed L1-loss and bounding box parametrization as adopted in [45].

4.4.2.4 Fusion Approaches

In figure 4.2, detections produces by a number of RPN+GBT heads are merged together and post-processed using non-maximal suppression. This approach of putting the information from multiple convolutional layers together is called *late fusion*. Another approach called *early fusion* is shown in figure 4.3. In early fusion, we resize the feature maps fro multiple layers to a common size and concatenate them together. This is followed by a single RPN+GBT head. Compared to the late fusion, early fusion is simpler in design and computationally more efficient due to smaller number of operations involved due to less number of RPN heads. We have experimented with both approaches and we show the results in the next section.

4.5 Experiments and Results

4.5.1 Training

We train each RPN using the SGD optimizer for 30 epochs with a learning rate of 0.001 and a momentum of 0.99. The learning rate was decreased by a factor of 10 after every 10 epochs. A total of 4 Nvidia Titan X (Maxwell) GPUs; each with 12 GB of memory; were utilized during training. We use the caltech 10x training set and caltech 1x testing set in our experiments as in section 4.4. During the training process, the convolutional layers of the base network are also modified. Based on our experiments, we found that for the case of late fusion, the RPNs should be trained starting from the lowest convolutional layers and down to the highest. Moreover a RPN should be allowed to modify only the convolutional layers, whose feature maps are concatenated to supply the input to the RPN. This helps in making the training process more stable.

4.5.2 Results

Subset	**Log-Average Miss Rate (Testing)**		**Testing Set**	**Training Set**
	Late Fusion	**Early Fusion**		
Caltech Reasonable	9.25%	**8.84%**	Caltech-reasonable(1x-Test)	Caltech-reasonable (10x-Train)
Caltech-All	46.2%	**44.37%**	Caltech-all (1x-Test)	Caltech-all (10x-Train)

Table 4.4: Log-averaged miss rate over different subsets of caltech[46]. The testing and training subsets are shown in the 3^{rd} and 4^{th} columns. New caltech annotations of [46] are used for training and testing.

Method	**Caltech-Reasonable(1x) Test Set**
Faster-RCNN[45]	15%
RPN-BF[177]	9.58%
RPN-BF (Using only RPN)[177]	14.8%
SDS-RPN[14]	9.63%
SDS-RCNN[14]	7.6%
MS-CNN[16]	9.95%
FDNN[36]	8.65%
Ours	**8.84%**

Table 4.5: Comparison with other works with performance on Caltech-reasonable(test set) of caltech1x

Heights	Log-Average Miss Rate (Caltech-All Testing)	
	Late Fusion	Early Fusion
>80px	3.25%	3.11%
50-80px	12.66%	11.25%
<50px	64.20%	64.17%

Table 4.6: Miss-Rates for different pedestrian height ranges in the caltech-all testing (1x-test) set. This includes all occlusion levels. New annotations from [46] are used for training and testing.

In table 4.4 we show the performance of the proposed system on two variations of the caltech 1x testing set. In the first row, we show the performance on the reasonable subset (*height* > 50 px and *occlusion* $< 35\%$). In the second row results are shown for all annotations in the caltech 1x testing set. Table 2.1 shows a comparison of our performance vis-à-vis other recent methods; revealing a comparable and competitive performance of our approach. Table 4.6 shows the height-wise performance of the proposed approach over the caltech-all test dataset.

From tables 4.4 and 4.6, we see that early fusion provides a small but definite performance improvement over late fusion. During early fusion, detection is performed based on information of all the convolutional layers put together. On the contrary, during late fusion, detections are obtained based on information in individual layers, followed by the coalescing of the results. Apart from performance benchmarks in tables 4.4 and 4.6, this qualitative argument demonstrates that early fusion is a better and direct way to harness multiple convolutional layers simultaneously.

In our implementation of early fusion we have used resizing of the feature maps from different layers to the fixed size. One of the limitations of this implementation approach is that due to bilinear interpolation for resizing, training takes a little longer, as has been documented in [74]. The use of à trous convolution can mitigate this behavior. We have not utilized à trous convolution in this work because its use in VGG16 is limited as we explain next. Without changing the structure of the VGG16 network drastically, the only reasonable à trous value to use in it is 2. This à trous convolution can be carried out right after each pooling layer, in addition to setting the pooling stride to be 1. This setting allows one to have feature maps before and after the pooling operation to be the same. Therefore, in order to completely avoid explicit resizing, yet to have all feature maps to be of the same size, the only approach is to use à trous convolution after each pooling operation which leads to each feature map of the same size as the input image. While this is theoretically possible, it is computationally difficult to achieve because of limited memory of computational devices. The total number of feature map channels (*when considering*

all the feature maps in VGG16) is 8192, and therefore when using an input image of size $P \times Q$, the total amount of memory needed to store all the feature maps (*using* 32*-bit floating point numbers*) is ~ 10 GB. In addition weights in RPN and convolutional filter weights have to be stored as well. In our experiments we have found such a configuration to be untenable. In our proposed approaches in chapters 5 and 6 we have used ResNet class of networks and have utilized à trous convolution therein to perform early fusion of multiple layers.

4.6 Analysis

Our work demonstrates that combining multiple convolutional layers can improve pedestrian detection. We have experimented with two approaches to utilizing multiple convolutional layers simultaneously –*early fusion* and *late fusion*. Our experiments demonstrate that early fusion is better than late fusion on accounts of both detection accuracy and detection speed. Our experiments have also demonstrated that different layers have differing affinities for scale and occlusion. It is hence, beneficial to explore approaches to make individual layers more scale and occlusion specific, where a layer is specialized to detect pedestrians in a specific scale or occlusion range. We consider this a future objective of our work. We have also explored the use of running accuracy metric in the RPN loss function. While it has helped in faster training, its precise impact needs to be verified by ablation studies. Inclusion of $f(h_i)$ directly in denominator can give big jumps to the loss function thereby reducing the stability of training. Exploring other weightage functions is also an important part of our future work.

Another major takeaway of this work is a quantitative record of the impact of various convolutional layers on detection of pedestrians of varying scales and occlusion levels. To the best of our knowledge, while this varying behavior of different layers has been known and acknowledged for long, its precise quantitative study on pedestrian detection has never been carried out before. Therefore, this work serves as an important guide for future design approaches for pedestrian detection.

Chapter 5

A Spatial Attention Approach to Pedestrian Detection

Contents

5.1 Introduction

In this chapter we describe our contributions towards designing a pedestrian detection with a high detection accuracy and high detection speed at the same time. We begin by describing the speed and accuracy trade-off in object detection. This is followed by a brief look at some related literature relevant to the ideas presented in this chapter. In deep learning, the speed/accuracy tradeoff is a consequence of technical differences between single-stage and two-stage detectors. We have looked at these technical differences in sections 2.2.2 and 2.2.1 in chapter 2. Here, we briefly summarize those differences and utilize them to motivate an important concept called "processing targets". The problem of designing a pedestrian detection with high speed and high detection accuracy is then formulated in terms of selection of processing targets. We take a brief look at different methods for achieving it and then introduce our usage of pseudo semantic segmentation to select the processing targets. This is followed by a more exhuastive description of the entire system. We make use of several observations from the preceding chapters and gradually describe the design of the system as a whole. We close with a quantitative and qualitative discussion of our experiments and results.

5.2 Revisiting the Speed/Accuracy Tradeoff

The accuracy of a pedestrian detector is the fundamental metric for measuring its effectiveness in detecting pedestrian instances. In practical applications, the overall effectiveness of a detector is often described by the accuracy coupled with inference speed [9]. The concept of inference speed describes the number of images processed by the detector per unit time. It is often expressed in *frames per second* (fps). The frames correspond to RGB images being fed to the detector. These frames may correspond to distinct images coming from different sources or frames coming from a video stream.

The traits of detection speed and accuracy are understood to be in trade-off with respect to each other [74]. Detectors exhibiting higher accuracy often perform more intensive computations which renders them with a low inference speed, and vice versa. Often, in such cases the trade-off has to be balanced by careful experimentation to choose appropriate hyperparameters during inference. For instance, in Faster-RCNN [130], the number of proposals to be selected is continually adjusted and experimented with to determine the minimum number of proposals which lead to decent accuracy and speed [74]. A different approach which is also adopted is network quantization; instead of using 32-bit floating point representation of network weights, 16-bit or even 8-bit representation is used. This is either done *offline* [96] – quantize a pre-trained network without retraining, or *online* [77] – retraining a network with a dedicated quantization approach to learn quantized

weights directly. However, this approach always leads to some loss of accuracy and can be complex. For instance, in case of *offline* quantization when finetuning, one has to fine-tune the 32-bit representation followed by its quantization. In the case of *online* quantization, the finetuning needs to follow complex mechanism of quantization where approximation strategies need to be assessed [82] to balance the *speed/accuracy* trade-off.

This encourages us to look for formulating new architecture(s) and systems which enable us to take advantages of a detector with simultaneous high accuracy and high inference speed. In this chapter, we discuss a pedestrian detection system which leverages semantic segmentation features for fast and accurate pedestrian detection. The proposed detector is lightweight and performs less computations than conventional detectors, thereby achieving high inference speed. The less computations as will be seen, are performed while focusing on potential pedestrians, thereby achieving high accuracy.

In the next section, we describe the fundamental traits of existing detectors by categorizing them as –*single-stage* and *two-stage* detectors. This categorization as will be seen, exhibits the *speed/accuracy* trade-off of conventional detectors.

5.3 Related Work

We limit our focus to deep learning based pedestrian detection systems. Most contemporary pedestrian detection systems are derived from Faster-RCNN [130], SSD [102] or YOLO [127]. Of these, Faster-RCNN is most commonly used as the basis for building pedestrian detection systems [177, 14, 16, 108, 113, 181, 95], on account of better detection accuracy than *one-stage* detectors. Pedestrian detectors based on *one-stage* detection systems include [171, 103, 36, 49, 129, 90, 120, 109]. Our treatment of related work focuses on *speed/accuracy* trade-off in contemporary pedestrian detectors and delineating details which offer the cue for balancing this trade-off; thereby setting the basis of our contributions. We have noted in chapter 2 that existing pedestrian detectors have primarily been derived from general-category detectors and can be categorized into two-stage and one-stage detectors. Figure **??** in chapter 2 summarizes the relative performance of one-stage and two-stage detectors vis-à-vis speed and accuracy. In the next two paragraphs we briefly cover a summary of some major pedestrian detectors categorized into one-stage and two-stage detectors. We then review some works which utilize semantic segmentation to improve pedestrian detection. In our proposed approach, semantic segmentation has been used in a novel way to improve detection performance and detection speed simultaneously, and is described in detail later in section 5.7.

***Two-stage* pedestrian detectors :** Approaches extending Faster-RCNN to pedestrian detection include use of tree-based classifiers [177, 159], use of multiple CNN layers

[16, 159], use of additional information such as optical flow, segmentation and different color channels [108, 14], use of different networks for processing different scales of pedestrians [95] and novel loss terms such as repulsion loss [166] for improved localization. These extensions improve upon the generic Faster-RCNN detector for pedestrians by an order of $5-7\%$ LAMR. However a comparable improvement in inference speed is not observed. Often these extensions invoke increased system complexity thereby requiring more *floating point operations per second* (FLOPs), which lowers the inference speed. Generally the performance of *two-stage* detectors varies from $8-14$ fps, while that of generic VGG16 based Faster-RCNN detector lies in the range of $7-10$ fps.

All *two-stage* pedestrian detectors use region proposal network (RPN) for proposal generation. As mentioned before and illustrated in figure **??**, the features for proposal generation in RPN are generated without intra-anchor feature pooling. These proposals are often poorly localized [177] and require a further classification and regression stage [177, 14, 16] for improved performance. All the *two-stage* detectors utilize intra-anchor feature pooling after the proposal generation. These pooling operations are carried out over a large number of proposals to minimize the miss-rate. As a result, the inference speed of *two-stage* detectors are limited by the number of processed proposals in addition to system complexity.

One-stage pedestrian detectors : *One-stage* pedestrian detectors are based on SSD [102] or YOLO [127]. The performance of generic SSD and YOLO detectors on pedestrian detection is significantly lower than that of Faster-RCNN. Their extensions to pedestrian detection include multi-step training of SSD [103], use of late fusion of multiple networks to refine the pedestrian candidates generated by SSD [36], recurrent networks for incorporating context [129] and use of skip connections in YOLO [90]. These extensions have improved their performance vis-à-vis their generic counterparts. The recently proposed ALF-net follows the ideas of cascade-RCNN [18], but over SSD [102]. ALF-net achieves an impressive 4.5% miss-rate on caltech-reasonable dataset while operating at ~ 20 fps. This performance is still lower than the performance of generic SSD ($48-60$ fps). Other *one-stage* pedestrian detectors [119, 90, 120, 109] report their runtime performance in the range of $20-25$ fps which is substantially lower than their generic counterparts. This reduction is primarily the result of added system complexity. For example [36] performs late fusion of multiple CNN networks, each of which operates upon the pedestrian candidates generated by a SSD which is pre-trained to generate pedestrian proposals. At the same time, *one-stage* detectors share the lack of intra-anchor feature pooling which fails to provide as relevant pedestrian features as *two-stage* detectors.

Use of semantic segmentation for pedestrian detection : The use of semantic segmentation in a deep learning setting for pedestrian detection was used in F-DNN [36]. The masks in [36] are predicted by a separate network trained for semantic segmentation and then used as a post-processing cue to remove invalid detections. This however makes it difficult to tune F-DNN for datasets like caltech pedestrians [34] which do not come with groundtruth segmentation masks. This deficiency is answered by SDS-RCNN [14]. SDS-RCNN uses the pedestrian bounding box in the training data to construct a pseudo-segmentation mask. A pixel-wise cross entropy term to classify background from pedestrian instances augments the standard loss function of the RPN, thereby forcing the output feature map of RPN to focus better on pedestrian instances. The RPN features are then used for classification and regression by a second network. This multi-task approach though very promising suffers from the limitations discussed before for *two-stage* pedestrian detectors. Furthermore, semantic segmentation has been used with the objective of improving the detection accuracy without any focus on utilizing it for improving the inference speed. In contrast, our proposed approach shows that the use of semantic segmentation based on pseudo-segmentation masks naturally leads to a mechanism to reduce the number of anchors to be processed by as much as 97%. This is the key to invoke intra-anchor feature pooling without suffering a runtime performance setback.

Our approach utilizes the ideas of SDS-RCNN [14] but without using a RPN. We prune most of the anchors away and use a combination of visible and full body bounding boxes to select positive and negative anchors which are feature pooled for final classification and regression. This approach therefore leverages the best of both worlds – *two-stage* (intra-anchor feature pooling favoring detection accuracy) and *one-stage* (reduced computations favoring inference speed).

5.4 Fundamental traits of single-stage and two-stage detectors

The case of two-stage and single-stage detectors has been described in detail in sections 2.2.1 and 2.2.2 respectively. In this section we briefly recall their characteristics with respect to feature handling. These characteristics as we will see have a major bearing on the processing speed and detection accuracy.

5.4.1 Two-stage detectors

Two-stage detectors use a fixed kernel size to probe a feature map using convolutions. The feature map is generated by a CNN and is tiled with hypothetical *anchor boxes*. Anchors can be of varying sizes and hence a fixed kernel cannot probe the entirety of features covered by an anchor box. Thus this kind of feature probing is an approximation. This

feature probing is used in *region proposal network* (RPN) to classify a given anchor as *positive* or *negative* during training. This, results in a reduction in the number of anchors to be processed for obtaining the final detections. The selected *positive* anchors are then passed to the second stage which performs intra-anchor pooling using ROI-pooling [130] or ROI-align [43] to obtain features covering the entirety of an *anchor box*.

5.4.2 One-stage detectors

In comparison, a one-stage detector such as SSD [102] or YOLO [127] bypasses the intial region proposal stage (corresponding to the first stage of two-stage detectors such as Faster-RCNN [130]). Instead one fixed kernel size is used to probe the features and perform final classification and regression. Let us assume that A confocal anchors are located at each location in a feature map F. Let F_p be the feature vector extracted by the kernel at a location p in F. The feature vector F_p then represents the features using which the classification and regression of all A anchors at p is done. Since different anchors can have different sizes and aspect ratios, this approach is sub-optimal owing to the lack of intra-anchor feature pooling.

From a computational perspective, one-stage detectors are faster due to the lack of intra-anchor feature pooling. The price for this high inference speed is paid in terms of detection accuracy. Intra-anchor feature pooling is a slow operation owing to the fact that tensors are not stored in a contiguous manner in the memory. Intra-anchor feature pooling involves extraction of a subset of a tensor and this involves repeated fetches through the memory in a non-contiguous manner leading to the operation getting slower.

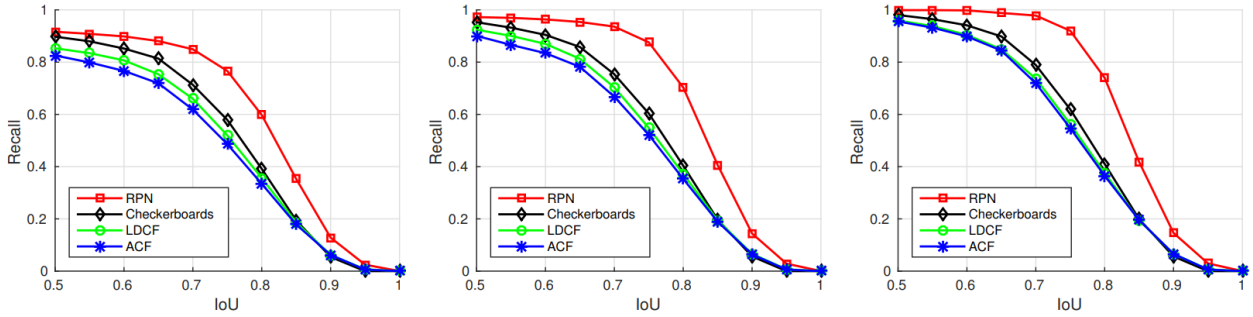
This leads to the insight that the feature handling mechanism of a detector has a coupled and opposing impact on detection accuracy and speed. The fundamental question to ask here is – *how can a balance be struck between two-stage and one-stage detectors* ? In the remaining sections we will see an approach which leverages semantic segmentation in order to reduce the number of anchors. Reduction in the number of anchors to be processed implies fewer intra-anchor pooling operations thereby improving the inference speed.

5.5 On the number of processing targets for convolutional detectors

We use the term “processing targets" to jointly refer to RPN proposals in Faster-RCNN-based detectors and the receptive fields of feature map cells processed by one-stage detectors like SSD [102] and YOLO [127]. The number of “processing targets" is directly

proportional to the number of convolutional operations for – *classification* and *regression* of a bounding box.

This offers an alternative insight into the *speed vs. accuracy* performance of single-stage and two-stage detectors. Let S^1 and S^2 respectively be the number of processing targets for one-stage and two-stage detectors respectively. Single-stage detectors such as SSD, perform $\mathcal{O}(S^1)$ convolutional operations while two-stage detectors such as Faster-RCNN perform $\mathcal{O}(S^2)$ convolutional operations. The cost of these convolutional operations is different though – $\mathcal{O}(S^2)$ operations are performed alongwith feature pooling operations thereby making them more costly. $\mathcal{O}(S^1)$ operations do not involve feature pooling and hence are considerably less costly and hence faster. In general $S^1 < S^2$, but due to feature pooling $\mathcal{O}(S^1)$ takes lesser time than $\mathcal{O}(S^2)$ (see table 5.1 for an example). As a reminder, as stated before, lack of feature pooling leads to less relevant features being captured which impacts the detection fidelity adversely.



(a) 1 proposal (b) 4 proposals (c) 100 proposals

Figure 5.1: Impact of number of RPN proposals on the recall of pedestrians. Shown for 4 techniques. Of these except RPN [177], the others are non-deep learning based techniques. As the number of proposals is increased, the recall is stablized over a large range of *intersection-over-union* with groundtruth bounding boxes. The dataset used here is the caltech-reasonable test set.

The above discussion concludes that the number of processing targets should be small for maintaining high detection speed. In order to maintain high detection accuracy, the locations of processing targets vis-à-vis should correspond to the locations of objects. This can be quantified by considering all object locations as the *positive* class, and all other locations as the *negative* class. Then based on the overlap between the processing targets and the positive class, the precision and recall for the processing targets can be determined. This kind of quantitative analysis is useful only for two-stage detectors due to a proposal generation stage which performs classification and bounding box regression of anchors. For a single-stage detector using $N \times P$ feature map, there are a total of NP processing targets and no bounding box regression is performed for anchors. There a precision-recall

analysis of processing targets for single-stage detectors is not very useful.

Thus we conclude that, the number and quality of processing targets is an important key to maintain a balance between detection accuracy and detection speed. A number of non-deep learning techniques also limit the number of processing targets by the use of proposals (section 2.2.1), and show a similar trend where the number of proposals impacts the detection accuracy and speed. However, deep learning techniques being computationally more intensive show this impact in a rather clearer manner. An illustration appears in figure 5.1, when detecting pedestrians in the caltech-reasonable test set. For increasing number of processing targets, the recall is considerably stablized for larger ranges of *intersection-over-union* with the groundtruth bounding boxes.

Network	**Number of Processing Targets**	**Inference/sec**
FRCNN (VGG16)	600	7
SSD (VGG16)	1939	48

Table 5.1: A comparison of the number of processing targets for Faster-RCNN [130] and SSD [102] with VGG16 [140] as the base network. For Faster-RCNN the figures correspond to 600 object proposals being selected for processing by the second stage. For SSD, the number of processing targets correspond to the network topology as shown in [102]

Reducing the value of P while encompassing all relevant pedestrians is therefore an important problem. We take up a semantic segmentation approach to reduce the number of processing targets (P). As will be seen in subsequent sections, this approach is a two-pronged approach for improving the detection accuracy while maintaining a high detection speed. This is possible because using the semantic segmentation approach, we select a minimal number of relevant processing targets. This eliminates the majority of *false positives* thereby improving the ensemble detection performance. At the same time a small value of P makes the system amenable to increased inference speed.

5.6 Reducing the Number of Processing Targets

As noted before, reducing the number of processing targets essentially means

There is very little to no prior information available about the location of objects in an image. For instance, assuming a natural scene with pedestrians in it, there are some regions (*e.g:- top of the image usually corresponding to open skies or high rises*) of the image which are less likely to have pedestrians. However, it is untenable to utilize this information in a quantitative setting because it is possible to have other image instances where

this is not true at all. For example, a surveillance camera mounted in an indoor setting covering a hallway is likely to break from such statistical assumptions. Nevertheless, such information has been utilized in the past through approaches such as GIST [116, 32] and visual codebook generation approaches trying to model shape and spatial context using feature histograms [164]. The primary objective of these early efforts was to reduce the number of regions to be processed while ensuring a low rate of false negative detections. Interestingly empirical studies on the use of such prior information in object detection such as [32] show that such prior information is hurtful in cases where different types of objects share similar context (*e.g:-cats and dogs, near and far-scale pedestrians*). Such early empirical studies have largely led to object specific appearance models being more favored.

Other approaches to reduce the number of processing targets include using a cascade of classifiers as in the AdaBoost setting [17, 9] and using object proposals [177]. The use of the cascade of classifiers reduces the number of processing targets in the sense that early stages remove discard most of the image regions, while successive stages process only regions not discarded by any of the previous stages. Boosting based classifiers are difficult to implement and train in a deep learning setting for reasons explained in section 2.3.5. Object proposal techniques select a small set of image regions which are likely to contain an object of interest. In the case of pedestrian detection this implies regions which are likely to contain pedestrians. Object proposals can be envisaged as saliency maps reflecting the liklihood of an image region as containing an object of interest.

In this work, we utilize semantic segmentation as the avenue for reducing the number of processing targets. Semantic segmentation has been used in the past [14] for improving pedestrian detection. However this improvement was motivated by improving the feature map fidelity for pedestrian detection and not improving the balance between detection speed and accuracy. In the next section we describe the details of utilizing semantic segmentation to reduce the processing targets.

5.7 On Semantic Segmentation to Reduce Processing Targets

Semantic segmentation is used to generate pixel-level labels for objects in a given image. Generally pedestrians occupy a small portion of the image space and as such semantic segmentation of pedestrians is a sparse representation of the pedestrians in the image space. Traditionally processing targets in two-stage detectors are obtained by performing a 2-class classification (*object* and *background*) as in RPN [130]. As discussed in section 2.2.1, this classification does not involve intra-anchor pooling and hence is similar in mechanism to that followed in *single-stage* detectors such as SSD. In the lack of a sparse representation of the image space, this leads to the generation of proposals which visually do not corre-

Figure 5.2: Top 35 proposals (**green**) generated by RPN [130] for an image in the caltech-reasonable test set. The **red** boxes correspond to groundtruth bounding boxes for the labelled pedestrians.

spond to actual pedestrians (hence *false positives*). As an example, the illustration in figure 5.2 shows the top 35 proposals generated by RPN. Generally, a much larger number (*e.g: 300-600*) proposals are selected for the second stage of Faster-RCNN. As can be seen, there are far too many proposals compared to the number of groundtruth instances of pedestrians in the image. In the lack of a mechanism to ignore most of these proposals, valuable time and computational resources are spent on processing these proposals. Furthermore, this also keeps open the door of possibility that a classifier is unable to classify all the proposals with perfect precision and perfect recall. In other words, working with large number of proposal regions presents the danger of increased false positive detections.

To utilize semantic segmentation, we consider a pseudo segmentation mask, henceforth referred to as **PSM** for brevity. It is worth noting that most of the publicly available datasets for pedestrian detection do not come with a pre-annotated segmentation mask for each pedestrian. A PSM refers to the bounding box of a pedestrian considered as a segmentation mask. Figure 5.3 presents some examples of PSMs. The PSM is not an accurate segmentation mask for it is obtained from an inexact representation of a pedestrian by considering the bounding box as a mask. However, it has been used successfully in [14] and our experiments as will be seen reveal that it offers an excellent way to reduce the number of processing targets. We utilize the PSM in a pixelwise classification setting. We utilize a 1×1 convolutional layer to pixelwise classify a feature map into pedestrians and non-pedestrians. During training, this classification is guided by the PSM. Figure 5.4

Figure 5.3: **Left :** An image with pedestrian bounding boxes. **Right :** The pseudo segmentation mask for the image on the left.

shows a set of examples of probability maps obtained after pixelwise classification of a feature map whilst guided by PSM.

Figure 5.4 confirms the relevance of using PSM as stated before. The probability maps (**bottom row**) demonstrate a high degree of affinity for pedestrian locations, even for small-scale pedestrians. Anchors at locations corresponding to lower probability need not be processed thereby eliminating a large fraction of anchors from computations.

In this section we described the utility of semantic segmentation in the form of PSM for pixelwise classification. We next come over to the question of integrating PSM with a complete pedestrian detection pipeline.

5.8 Proposed Pipeline for Fast Pedestrian Detection

To describe our proposed pipeline we will take a step-wise approach where we will describe the individual components of the system alongwith the rationale for their design choices. This is helpful in piecing together the global view of the pipeline. Figure 5.5

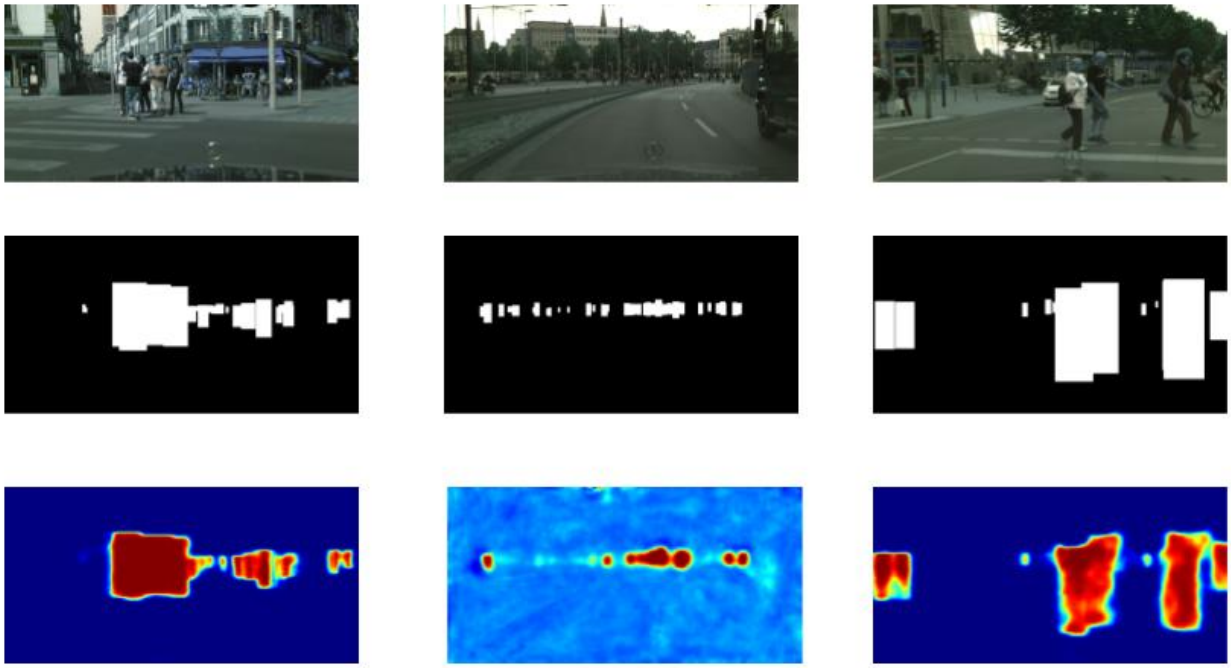


Figure 5.4: Pixelwise classification probability of image regions to be pedestrian.

summarizes the block diagram of the proposed approach.

5.8.1 Input

As in any object detection system, the input to our system consists of a minibatch $\mathcal{B}$ of images. All the images in our minibatch have the same size $H \times W \times C$ (*height* $\times$ *width* $\times$ *channels*).

Generally in nearly all practical applications encompassing surveillance and autonomous vehicles, the output resolution of a camera is fixed apriori. Therefore, no explicit resizing of images or frames is needed to pack multiple images in a minibatch. Hence, the concern of distortion occouring as a result of explicit resizing is not of gravity in practical situations. Nevertheless, in situations where images have to be resized to create a minibatch ($\mathcal{B} > 1$), no problem is posed if $|\mathcal{B}|$ is used for initial training of the target network. However, caution must be excercised by ensuring that in all subsequent training or fine-tuning of the target network, the minibatch height and width is not changed. Generally, in frameworks like SSD [102] or YOLO [127], where $|\mathcal{B}| > 1$ is used, various image dimensions are experimented with, by considering them as hyperparameters. Larger dimensions are preferred [16] on account of greater visual clarity of objects (*especially small objects*). However, larger image dimensions as input lead to more computations thereby

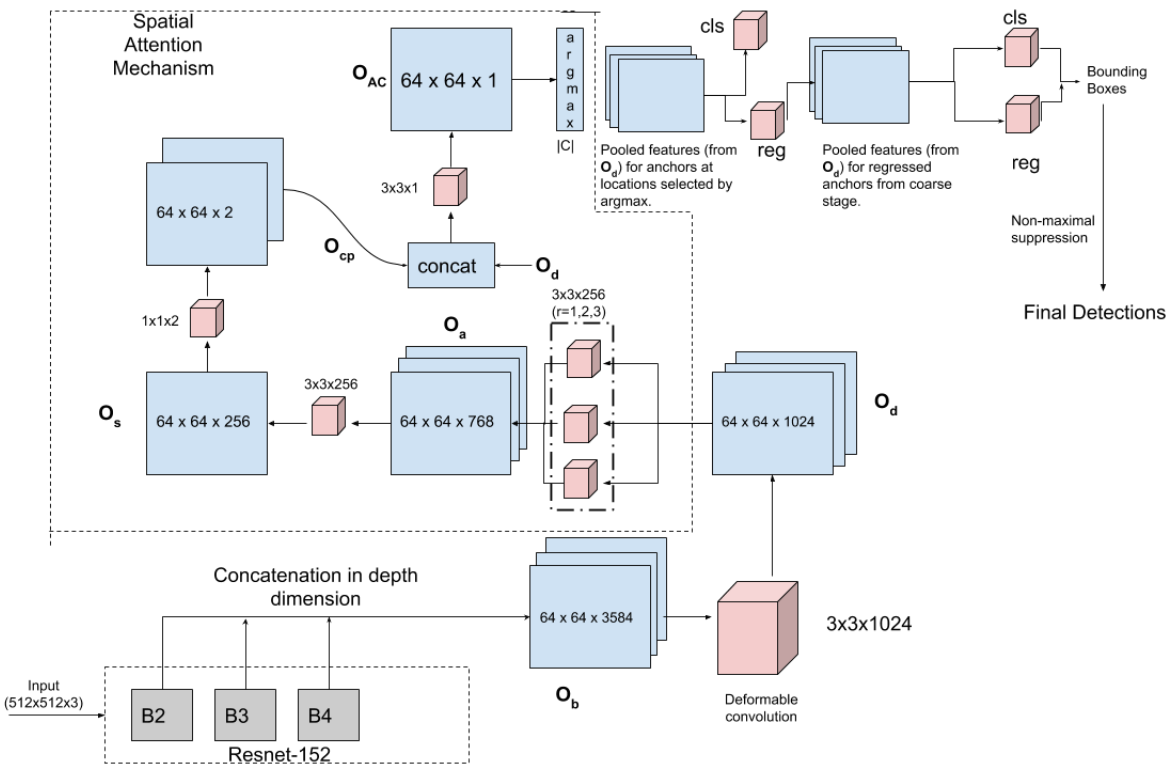


Figure 5.5: Block diagram of the proposed approach.

decreasing the inference and training duration. Therefore, the general practice is to use experiment with input image dimensions within a pre-specified range.

In our proposed system, we utilize $\mathcal{B}$ of dimensions $512 \times 512 \times 3$.

5.8.2 Base Network

The base network is a CNN architecture pre-trained on a dataset like imagenet [**?**]. We utilize Resnet-152 [67] as the base network in this work.

A summary of architectural details of various members of the resnet family is presented in figure 5.6. It can be seen that each one of these architectures can be grouped into 4 blocks (*conv2_x, conv3_x, conv4_x and conv5_x*. Successive blocks are separated by a max-pooling layer. In the rest of this chapter, we follow the following naming convention for referring to blocks. We consider the row in figure 5.6, corresponding to $convP\_x$ as $Block(P-1)$. Thus, $conv2\_x$ corresponds to $Block1$ and so on.

Considering the fact that CNNs are hierarchical structures, various layers extract features of varying complexity from an input image. To inculcate increased feature diversity, we utilize feature map outputs from $Block2$, $Block3$ and $Block4$. It can be seen in figure 5.6, that output feature map dimensions are different for these blocks. While bilinear in-

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5.6: Architectural details of various members of the ResNet family. Courtesy of [67].

terpolation can be utilized for resizing them to the same dimensions, it is not a preferred approach as it leads to unstable training during backpropagation as has been observed in [144]. We instead utilize à trous convolution to solve this problem. We set à trous rate for the convolutional filters in blocks 2 through 4 as 2. This results in an output feature stride of 8. For $\mathcal{B}$ of dimensions $512 \times 512 \times 3$, this results in feature maps of dimensions 64×64, which are concatenated along the channel dimension, to obtain a feature map ($\boldsymbol{O_b}$) of size $64 \times 64 \times 3584$.

5.8.3 Deformable Convolution

The operation of convolution in deep learning involves sliding a $m \times n$ filter over a feature map and computing its correlation with the feature map values for each sliding location. As a passing remark worth noting for its relevance, convolutions in deep learning do not involve flipping the filter as is done in convolutions studied in signal processing literature [110]. Figure 5.7 shows an example. From figure 5.7, we see that at each sliding location, the filter kernel shape is congruent to the sliding location.

The operation of deformable convolution [30] introduced the concept of offset locations in convolution. For every sliding location of the filter kernel, a separate branch computes the offset values. These offset values denote the horizontal and vertical shifts for each feature map location. The correlation is computed with feature map values at locations determined by these shifts. Example of offsets are shown in figure 5.8. The separate branch consists of a convolutional filter and is described in [30]. The offsets might

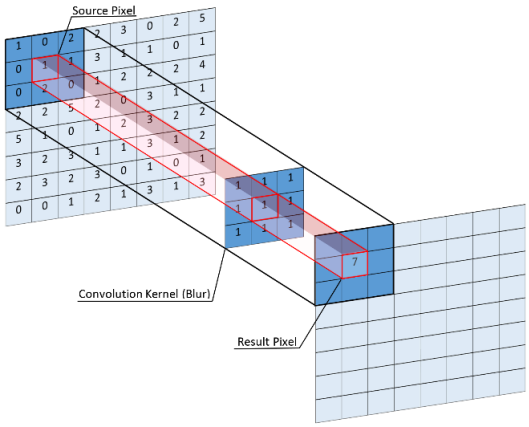


Figure 5.7: Standard convolution process in CNNs. The convolutional kernel computes correlation with feature map values for every sliding location. The correlation result becomes the feature map value for the center point location of the filter in the output feature map.

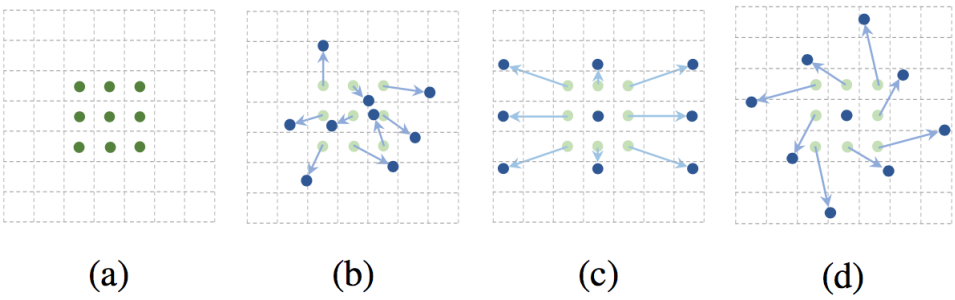
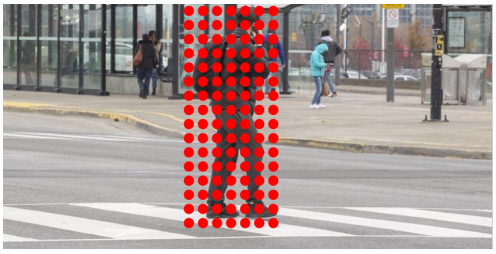


Figure 5.8: **a**: The sliding location of the filter kernel, **b,c and d**: Various examples of possible offset locations. The offsets are applied to each location at the sliding position of the kernel. The correlation is computed between the filter kernel and the feature map values at the locations after applying offsets (in blue).

result in non-integer locations in the feature map. To counteract this, bilinear interpolation of feature map values are used to find the feature map value for non-integral locations and they are used to compute the correlation.

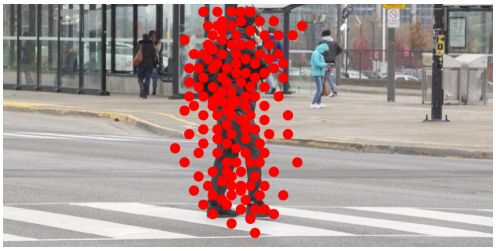
Deformable convolution owing to non-uniform sampling of feature map is able to learn more object centric and shape specific features compared to normal convolution. In figure 5.11, we show an illustration of sampling locations of both normal convolution (figure 5.9) and deformable convolution (figure 5.10) for one specific pedestrian. As can be seen deformable convolution has sampling locations which are more specifically glued to the pedestrian as compared to normal convolution.

We utilize 1024 filters of kernel size 3×3 with stride of 1 for deformable convolution in the proposed system. This convolution with $\boldsymbol{O_b}$ results in an output feature map $\boldsymbol{O_d}$ of



.5

Figure 5.9: Normal Convolution



.5

Figure 5.10: Deformable Convolution

Figure 5.11: Sampling locations shown for one pedestrian with **a): Normal Convolution** and **b): Deformable Convolution**. In both cases the convolution kernel is 3×3 with stride 1.

size $64 \times 64 \times 1024$.

5.8.4 Semantic Fully Convolutional Layer

In order to make use of semantic segmentation as explained in section 5.7, we make use of a set of parallel branches of fully convolutional layers which differ in their à trous rates. This structure is reminescent of à trous spatial pyramid pooling (ASPP) introduced in [22]. Figure 5.12 shows the details of the semantic segmentation module as used in our proposed system. The feature maps from the 3 branches are concatenated in the channel dimension to obtain $\boldsymbol{O_a}$ of size $64 \times 64 \times 768$. Another convolutional layer with 256 filters of size 3×3 acts on $\boldsymbol{O_a}$ to obtain a feature map of size $64 \times 64 \times 256$, which is then convolved with a convolutional layer with 2 filters of size 1×1 to obtain the a feature map $\boldsymbol{O_c}$ of size $64 \times 64 \times 2$, which we refer to as the pixelwise segmentation map.

The 2 channels of $\boldsymbol{O_c}$ correspond to *pedestrians* and *non-pedestrians* respectively. During training a pixelwise cross entropy loss is computed between $\boldsymbol{O_c}$ and PSM obtained from the training dataset. This trains the semantic segmentation module for predicting pedestrian locations in the input image.

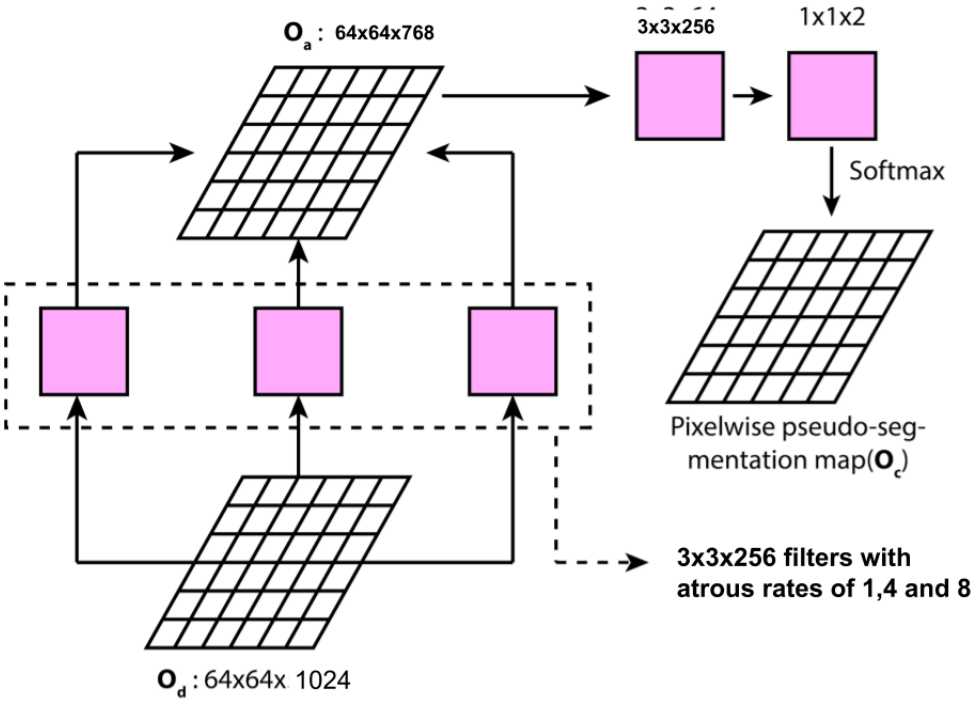


Figure 5.12: Semantic Segmantation module of the proposed system.

5.8.5 Anchor Location Selection

In existing anchor based detection frameworks such as Faster-RCNN [130], SSD [102] and YOLO [127], anchors are densely tiled over a feature map. In our approach the use of semantic segmentation, which provides a good estimate of pedestrian locations (figure 5.4), provides an opportunity to avoid tiling a dense set of anchors. Anchors may be centered only at locations with a high probability of pedestrians.

To achieve this, we make use of $\boldsymbol{O_c}$ as follows. Let $\boldsymbol{O_{cp}}$ be the channel of $\boldsymbol{O_c}$ corresponding to pedestrians. We concatenate $\boldsymbol{O_{cp}}$ with $\boldsymbol{O_d}$ in the channel dimension. The resulting feature map is processed by a convolutional layer with 1 filter of size 3×3 to obtain a feature map $\boldsymbol{O_{AC}}$ of size $64 \times 64 \times 1$. We infer a set of locations C where anchors are centered as follows :

$$C = argmax_{|C|}(\sigma(\boldsymbol{O_{AC}})) \tag{5.1}$$

where $\sigma(.)$ is the sigmoid function and $argmax_{|C|}(\boldsymbol{O_{AC}})$ is a function representing top $|C|$ locations in $\boldsymbol{O_{AC}}$. The parameter $|C|$ is a hyperparameter in our system. To facilitate backpropagation, we define the gradient of $argmax(.)$ as the identity function, which passes the gradients coming from the top layers to the bottom layers without modification. Anchors centered at C, are known as the candidate anchors in our work.

All the proposal anchors centered at locations defined by C, are subsequently used for

classification and bounding box regression.

5.8.6 Classification and Bounding Box Regression

In our work we perform classification and bounding box regression in two steps – *coarse* and *fine-grained*. For both steps, features pooled from the anchors located at C are used for classification and regression.

Coarse-Step
We pool features from the proposal anchors in $\boldsymbol{O_d}$. For pooling we crop the features inside the anchors and resize them to a fixed size followed by max-pooling by a 2×2 kernel with stride of 2. In our implementation the fixed size is 14×14. These features are fed to two sibling fully connected layers for classification and regression respectively.

Our regression function predicts – s_x (horizontal scale factor), s_y (vertical scale factor), t_x (horizontal translation) and t_y (vertical translation). Given an anchor with center (x_c, y_c), width as w and height as h, it is transformed as follows:

$$\begin{aligned} x_c^{new} &= x_c + t_x \\ y_c^{new} &= y_c + t_y \\ w^{new} &= w \times s_x \\ h^{new} &= h \times s_y \end{aligned} \tag{5.2}$$

In equation 5.2, variables on the left hand side are the transformed variables after regression.

Fine-grained Step All the anchors classified as pedestrians by the coarse-step classification, are transformed according to equation 5.2. The process of coarse-step classification and regression is then repeated using a different set of classification and regression layers, resulting in the final set of bounding boxes detected as pedestrians on the image.

In our work, since the fine-step of classification and regression, pools features from the regressed anchors of the coarse-step, it extracts features from an improved location. This helps in an improved prediction of bounding box coordinates. It is possible to cascade multiple such steps. However, in our implementation use of 2 steps was found sufficient for good results. Addition of subsequent steps increases the number of feature pooling operations – thereby decreasing inference speed.

5.9 Training

5.9.1 Loss Function

During training, all anchors overlapping with an $IoU > 0.5$ are selected as positive anchors, the remaining being negative anchors. In our proposed approach there are a total of 5 loss terms as enumerated below :

1. Spatial Attention loss (L_{att}) : This refers to the pixelwise cross-entropy between $\boldsymbol{O_c}$ and the groundtruth pseudo segmentation mask, averaged across all pixels. This cross-entropy is computed for 2 classes – *pedestrian* and *background*.

2. Coarse-step classification loss L_{cls}^{coarse} : The coarse stage classification loss is also a cross-entropy loss for the classification of a proposal anchor as *pedestrian* or *background*.

3. Coarse-step regression loss L_{reg}^{coarse} : For regression we use the smooth-L1 loss as used in [130]. Regression is performed only for proposal anchors classified as pedestrians. The regression loss is $L_1^{smooth}(p_i, p_i^*)$, where p_i denotes the transformation of the anchor box variables (s_x, s_y, t_x, t_y), while p_i^* denotes the same for the groundtruth box. These transformations are described in equation 5.3.

$$\begin{aligned} p_{t_x} &= log(\frac{t_x - x_c}{w_a}) \\ p_{t_y} &= log(\frac{t_y - y_c}{h_a}) \\ p_{s_x} &= log(s_x) \\ p_{s_y} &= log(s_y) \end{aligned} \tag{5.3}$$

   w_a and h_a are the width and height of the anchor respectively. The scaling is done assuming the scale of the anchor box as 1. Faster-RCNN during the RCNN stage, starts with proposal boxes which are already regressed. In our approach, there is a bigger difference between anchor and groundtruth bounding box dimension. We found that regression using s_x and s_y instead of width and height as in [130] lead to more stable training.

4. Fine-grained step classification loss L_{cls}^{fine} : Same as the formulation of L_{cls}^{coarse}.

5. Fine-grained step regression loss L_{reg}^{fine} : Same as the formulation of L_{reg}^{coarse}

6. Regularization losses L_R : We use l_2 regularization in all our convolutional layers. L_R is the sum of all the regularization terms in our detector.

Test Set	LAMR
Caltech1x (trained on caltech10x)	4.83
Caltech1x (trained on citypersons + caltech10x)	3.79
Citypersons-val (trained on citypersons train)	11.58

Table 5.2: Performance of the proposed approach on caltech1x and citypersons-validation set. ($|C| = 350$).

Method	Caltech1x	Citypersons-val
RPN-BF[177]	9.6	N/A
MSCNN[16]	10	N/A
SDS-RCNN[14]	7.6	N/A
GDFL[?]	8	N/A
SSD (Resnet-152)[102]	11	N/A
Rep-Loss[166]	4	13.1
Ours	**3.79**	**11.58**

Table 5.3: Comparative performance of the proposed approach with other pedestrian detectors.

The total loss function for training our detector is

$$L_{total} = \frac{\alpha L_{att}}{N} + \frac{L_{cls}^{coarse} + L_{reg}^{coarse} + L_{cls}^{fine} + L_{reg}^{fine}}{N_a} + \lambda L_R \tag{5.4}$$

In equation 5.4, α is a constant scaling factor, while N_a is the minibatch size (*total number of anchors*) and N is the batchsize of images. λ is the regularization factors (set to 0.00005 in all our experiments). Pedestrian bounding boxes in training data contain several background elements as well. Our experiments with $\alpha = 1$, suggest that it causes small-scale pedestrians to be missed, especially with large-scale pedestrians around. Since L_{att} is obtained by averaging over all the pixels, large-scale pedestrians contribute more to it. Scaling L_{att} by small value of α is found by us to be a better approach for improved detections. In all our experiments, $\alpha = 0.8$.

5.9.2 Implementation Details

In our implementation, we obtain the best results with $|C| = 350$. We use anchors with two aspect ratios $(0.41, 2.41)$ and 6 scales $(0.25, 0.5, 0.75, 1, 2, 4)$ resulting in 12 anchors at each location. All anchors have been generated with a base anchor size of 64×64. Thereby, $12 \times 60 = 720$ anchor features are the input to our coarse-stage classification.

For training, we utilized the stochastic gradient descent algorithm. We did a warm-up phase where the learning rate ($lr = 0.003$) for first $10K$ steps, after which lr was

Two-step (cls+reg)	Deformable Convolution	Early Fusion	LAMR
✓	✓	✓	4.83
✓	✓	–	6.22
✓	–	✓	5.81
–	✓	✓	5.67
–	–	–	7.19

Table 5.4: Summary of ablation studies on caltech-1x dataset. These ablations were performed for $|C| = 350$ and directly training the system on caltech10x-train.

reduced by a factor of 0.5 after every $10K$ steps. Data augmentation was used in our training experiments with random horizontal flip and random brightness and contrast adjustments. The batchsize used in our experiments was 2. Our proposed system was implemented in TensorFlow [1] and was run on a single NVidia V100 GPU with 32 GB of GPU memory.

5.10 Experiments and Results

5.10.1 Datasets

We experimented with the caltech-reasonable [34] and citypersons [179] datasets. For caltech-reasonable, we utilized the improved annotations provided by [178] for the $10x$ training set (42782 images) and for the $1x$ testing set (4024 images). For the citypersons dataset we used the original annotations from the authors [179]. For citypersons we trained on the training set (2975 images) and tested on the validation set (1525 images).

For both the datasets, we utilized only the reasonable subset annotations (height≥ 50 pixels, occlusion≤ 0.35) for training. We use *log-averaged miss-rate* (**LAMR**) [34] as the evaluation metric.

5.10.2 Results

5.10.2.1 Detection Accuracy

Our best results were obtained with $|C| = 350$, by training our system first on caltech10x training set followed by fine-tuning on the citypersons training set. Table 5.2 summarizes the results on caltech-reasonable and citypersons (validaton) datasets for different choices of training set. In table 5.3, we provide a comparison of our best numbers with several other pedestrian detectors. Table 5.3 shows that our system outperforms the other methods on both *caltech-reasonable* and *citypersons* validation sets. Methods such as [16, 166, 14] perform upscaling of input images prior to feeding them to their systems.

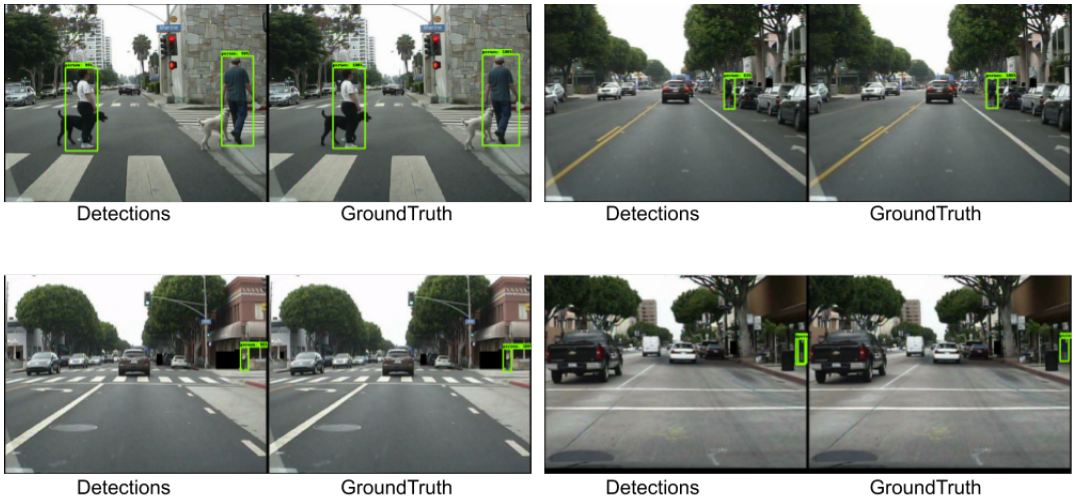


Figure 5.13: Some detections (**Left**) on the validation set of caltech-1x dataset compared with corresponding groundtruth (**Right**).

Method	Inference Speed (fps)
RPN-BF[177]	7
MSCNN[16]	8
SDS-RCNN[14]	5
SSD (VGG16)[102]	48
Rep-Loss[166]	N/A
Ours	11

Table 5.5: Comparative performance of the inference speed of various detectors. For our proposed detector, the inference speed corresponds to $|C| = 350$.

Upscaling has been shown in [16, 166] to improve performance by around $0.4 - 1.7$ LAMR points. Upscaling input images to higher sizes is likely to further improve the performance of our system.

5.10.2.2 Detection Speed

Our system operates at an inference speed of $\sim 11 fps$ under a batchsize of 2 images, for $|C| = 350$. This speed is comparable with the performance of most Faster-RCNN based systems. As table 5.7 shows, the inference speed decreases with an increase in the value of $|C|$. This is expected since the number of feature pooling operations is proportional to the value of $|C|$. Table 5.7 in conjunction with table 5.6 details the relative loss in detection accuracy for an increase in inference speed by decreasing the value of $|C|$. Even at $|C| = 100$, the performance obtained is quite competitive with the detection speed being at 20 fps. This shows that the proposed system offers considerable flexibility in terms of

|C|	LAMR	
	Caltech-1x	**Citypersons-val**
30	8.59	15.3
50	8.41	14.73
60	8.05	14.21
100	6.69	13.95
350	**3.79**	**11.58**

Table 5.6: Effect of varying the hyper-parameter $|C|$. The miss-rate for caltech-1x is based on model pre-trained on citypersons-train followed by fine-tuning on caltech-1x. The miss-rate for citypersons-val is based on model trained on citypersons-train.

|C|	Detection Speed (fps)
30	60
60	36
90	24
100	20
350	**11**

Table 5.7: Impact of varying the hyper-parameter $|C|$ on inference speed.

detection speed while maintaining competitive detection performance. We also observe from table 5.7 that, for lower values of $|C|$, the inference speed can be as high as 60 fps, which is higher than the speed of SSD [102] (with VGG16 as base network) (table 5.6). Feature pooling is a slower operation and is avoided by SSD for that purpose. However, for lower values of $|C|$, even with intra-anchor feature pooling, our proposed system performs fewer operations compared to SSD. This is on account of the fact that, SSD performs multi-scale computations which effectively increase its computations compared to our approach. However, for higher values of $|C|$, there are significantly more pooling operations which results in inference speeds more similar to other faster-rcnn based approaches.

5.10.3 Ablation Studies

In table 5.4 we summarize the ablation studies in our experiments. The use of deformable convolution assists in improved $\boldsymbol{O_{cp}}$ and provides a boost of ~ 0.9 LAMR points. The early fusion of multiple layers is responsible for a gain of $\sim 1.4\%$ in LAMR, thereby verifying the effectiveness of using multiple layers. To study the role of two-stage classification and regression, we switched to single-step classification and regression, which lowered our LAMR by 1.04%. Most of this loss in LAMR was observed for pedestrians with occlusion close to 0.35% (upper limit of occlusion in caltech-reasonable), and was owing to insufficient bounding box overlap. This is expected, as an anchor may not overlap fully with

a pedestrian under high occlusion conditions. Hence, it is difficult to estimate the full bounding box by the partial view of a pedestrian.

In table 5.6 we show the effect of varying the hyper-parameter $|C|$ which decides the number of anchors to be selected. Higher values of $|C|$ lead to improved performance, but at the cost of inference speed as it leads to more number of feature pooling operations.

5.10.4 Spatial Attention : Comparison with RPN

It is not straightforward to quantitatively compare spatial attention with RPN. Spatial attention module does not perform any bounding box regression and does not select all anchors at any specified location. Thus, although RPN and spatial attention in our work – both aim to reduce the search space for pedestrians, the underlying mechanisms are different. Traditionally RPN is evaluated by measuring the IoU of its proposal outputs with the groundtruth to compute the LAMR. In our work since no regression is performed on the anchors by the spatial attention, comparison using IoU is not fair.

For a fairer comparison, we trained Faster-RCNN with ResNet-152 on caltech10x and compared its performance on caltech1x with our approach for the configuration in the last row of table 5.4. In this case, the only difference between Faster-RCNN and our approach remains in our use of spatial attention. We gain 0.42% of performance over standard

Faster-RCNN	**Ours**
12.02	**11.6**

Table 5.8: Comparison between spatial attention and RPN on the basis of LAMR

faster-RCNN (table 5.8). This basic comparison serves to show that the performance of the proposed spatial attention module is comparable to the RPN performance.

5.11 Discussion and Conclusions

In nutshell, the proposed approach combines the best of single-stage and two-stage detectors. Intra-anchor feature pooling utilized in two-stage detectors is a better feature handling mechanism compared to sliding of a convolutional kernel over a feature map, which is followed in one-stage detectors. The former approach provides for separate features for each anchor while in the latter approach, all confocal anchors share the same feature. The proposed approach utilizes intra-anchor feature pooling thereby making it more robust compared to single-stage detectors. At the same time, it gets around the lack of detection speed in two-stage detectors, by reducing the number of processing targets. This operation is done using a basic semantic segmentation module, which not only provides the basis for

selecting relevant anchors for processing, but also decreases the risk of false-positive detections. Compared to many previous approaches like [14], where semantic segmentation has been used to improve the discriminability of feature maps, the proposed approach takes this methodology a step further and harnesses it for better detection speed. The proposed semantic segmentation module is an effective replacement of the RPN stage of Faster-RCNN [130] – the hallmark of two-stage detection frameworks. As against RPN, the proposed approach does not perform any regression over bounding boxes. This goes on to confirm that while the proposed system is not perfectly single-stage, due to its reliance on pedestrian probability map, it is not perfectly two-stage because bounding box regression is an integral part of two-stage approaches. Our proposed system is hence, a significant deviation from existing two-stage approaches.

Feature extraction strategies following the principle of early fusion for better feature diversity and deformable convolution for better object centric features ensures that the features feeding the semantic segmentation module as well as the final intra-anchor feature pooling module are well aligned with the discriminative task of separating pedestrians from non-pedestrian entities.

The effectiveness of the proposed system depends heavily on the choice of certain hyperparameters, the most important one of which is the the parameter C (section 5.8.5). Our experiments summarized in table 5.6, confirm that the choice of C inpacts the detection speed as well as detection accuracy. One limitation of the proposed approach is the lack of any analytical procedure to either pre-determine C based on dataset statistics, or to integrate it directly in the training process for its opimal selection. Higher values of C, though improve the detection speed, but they also degrade the detection accuracy. We hence, envision that it is of considerable research and practical interest to explore a more analytical approach to determine C rather than treating it as a free hyperparameter.

Another major hallmark of the proposed system is a 2-step process of classification and bounding box regression. While a similar approach has been adopted in "Cascade-RCNN" [18], their basic motivation is quite different from ours. In [18], the focus is on using a cascade of classifiers and regressors such that each is adept at detecting objects within a quality measure (*intersection-over-union*), which is better than the detection quality of its predecessor. The training process outlined in [18], reflects this motivation. Cascade-RCNN cannot be trained end-to-end. Each stage of the cascade must be trained before the subsequent stages can be trained. To the contrary, in the proposed approach, the two steps can be trained together and thus the entire proposed pipeline is an end-to-end trainable system. The primary motivation behind using a two-step classification and bounding box regression is to make up for a lack of bounding box regression process used in RPNs. Our ablation studies confirm (section 5.4), that the two-step process improves the detection performance by around 0.86%. While this improvement might seem small,

it goes on to show that from a practical perspective it is indeed possible to have multi-step classification and regression with the advantages of end-to-end training. This can be especially beneficial in competitive cases where even a minor improvement in performance is extremely desirable. Increasing the number of steps though is possible, it leads to increase in the number of feature pooling operations thereby causing the detection speed to suffer.

Chapter 6

Reducing anchors even more : Faster and Better

6.1 Introduction

In the previous chapter we studied the use of semantic segmentation based spatial attention to reduce the number of anchors and therefore reduce the computational complexity of pedestrian detection. The result of this approach was a faster pedestrian detector with a high detection accuracy. Can we push the barriers further and look for even faster and better pedestrian detectors ? The subject of this chapter is to simplify the design introduced in the last chapter for faster and better pedestrian detection. The fundamental tenet of this simplification is based on further reducing the number of processing targets while ensuring that the processing targets cover the pedestrian instances very well.

This further simplification is achieved by the introduction of an anchor selection layer. The proposed anchor selection layer performs a quick classification of anchors to determine which anchors are the most relevant. In this process, during training it is aided by the visible as well as the full body bounding boxes. As will be seen, this approach while minimizing the number of anchors to be processed, covers the pedestrian instances quite densely.

There are other simplifications introduced in this present work, which aid in speeding up the detection process. Although their elaborate description is deferred to the later sections, these simplifications involve – pruning of the deformable convolutional layer and use of a single-step classification and regression process.

This chapter is organized as follows. We begin by examining in detail the performance characteristics of the system introduced in chapter 5. This examination is then followed by a description of the system proposed in this chapter. Particular emphasis is paid on eliciting the differences from the system in chapter 5 and understanding their impact on detection

accuracy and detection speed. We then present results and analysis of the proposed system followed by a discussion which summarizes the proposed work and points out the potential avenues for further research.

6.2 Related Work

The background literature for this chapter is primarily shared with the background in chapter 5. Hence, in this section, we limit ourselves to the related literature which is directly linked to two of our main contributions in this chapter, namely the anchor selection layer and the use of visible and full body bounding boxes.

Techniques for Anchor Classification : The term "anchor classification" must be supplemented with a reference to the application context. For example, in a two-stage detector like Faster-RCNN [130], anchors are classified two times – a) in the RPN stage, anchors are classified as either representing a class agnostic object or background entity and, b) in the RCNN stage, the regressed versions of anchors representing objects in the RPN stage are further classfied into one of the object classes. To be precise, in the RCNN stage classification does not directly act on an anchor but rather on its regressed version. It is still worthwhile to refer to them as anchors because the classification done in the RPN stage is 2-class and hence is not suitable for final object detection. For all practical purposes, the positively classified anchors in the RPN stage, are still object candidates.

The classification of anchors in the RPN stage has been described in detail in chapter 2. The major takeaway needed for this chapter is the fact, that this classification is based on a fixed-size convolutional kernel. Anchors are usually of varying aspect ratios and scales and hene a fixed-size convolutional kernel is unlikely to fully cover the region inside an anchor. To the best of our knowledge, this limitation has not received any response in the existing literature. Most attempts at improving the classification of anchors has been through the use of post-processing techniques such as Greedy-NMS [56] which is adopted in HyperNet framework [83], or through position-sensitive ROI-pooling [29], or by using surrounding information around an anchor during classification as is done in MS-CNN [16]. While these attempts have proven to be effective, they do not address the fundamental limitation that a fixed-size convolutional kernel is unable to cover the entirety of an anchor. During training and inference, typically the number and size of anchors is fixed. Therefore this limitation is not hard to overcome. Our proposed anchor selection layer overcomes this limitation by using a multibranch set of convolutional layers. These convolutional layers are lightweight, with each layer handling only a fraction of the anchors. This makes this proposed approach amenable to better classification accuracy as well as faster performance.

Use of visible and full body bounding boxes : Public datasets for pedestrian detection, are often provided with full-body as well as the visible-body bounding boxes [34, 179]. Few pedestrian detectors utilize only the full body bounding boxes for training. In [182]; a spatial attention based work addressing detection under occlusionl; visible bounding boxes are used in addition to full-body boxes and body part annotations. Body part annotations are coarsely defined based on the other two bounding boxes. The basic idea to re-weight the intra-anchor feature maps with the 3 attention maps; one for each type of annotation. Though it is outperformed by other detectors [14, 16] on caltech reasonable subset, it indicates major improvements in cases with partial or heavy occlusion. This suggests that the use of visible part of bounding boxes aids in detection under occlusion. Intuitively, an anchor box which overlaps sufficiently well with both the visible and full body bounding box is a better candidate for regression. Regression of such an anchor box is easier owing to more complete information about the pedestrian inherent in it. Comparatively, complete dependence on full-body or visible bounding box may lead to a wide variance in the information inherent in the anchor.

We next take a look at the performance characteristics of the system presented in chapter 5. Specific focus is given to the computational complexity and performance details of some of the components of the system. Understanding these details allows us to make appropriate refinements thereby achieving improved detection speed without loss of performance.

6.3 Performance Characteristics

For simplicity we have reproduced the block diagram of the system proposed in chapter 5 in figure 6.1. The rationale of the design characteristics of this system have already been described in section 5.8. Our objective in this section is to understand the impact of various components and design decisions in figure 6.1 on detection speed and accuracy.

6.3.1 Deformable Convolutional Layer

Deformable convolutional layer is a major improvement over the standard convolution as the former computes correlation by sampling locations which are non-uniformly distributed over the feature map. This non-uniform sampling is made possible by a separate branch (*referred to as the offset branch*) which consists of a convolutional layer. Let us consider that a 3×3 deformable convolution operation with a stride of 1 is being employed. Recalling the details of the deformable convolution presented in section 5.8.3, this implies that a total of $3 \times 3 = 9$ offset points must be computed for each sliding location of the deformable convolutional filter. To facilitate this, the convolutional layer in the offset branch

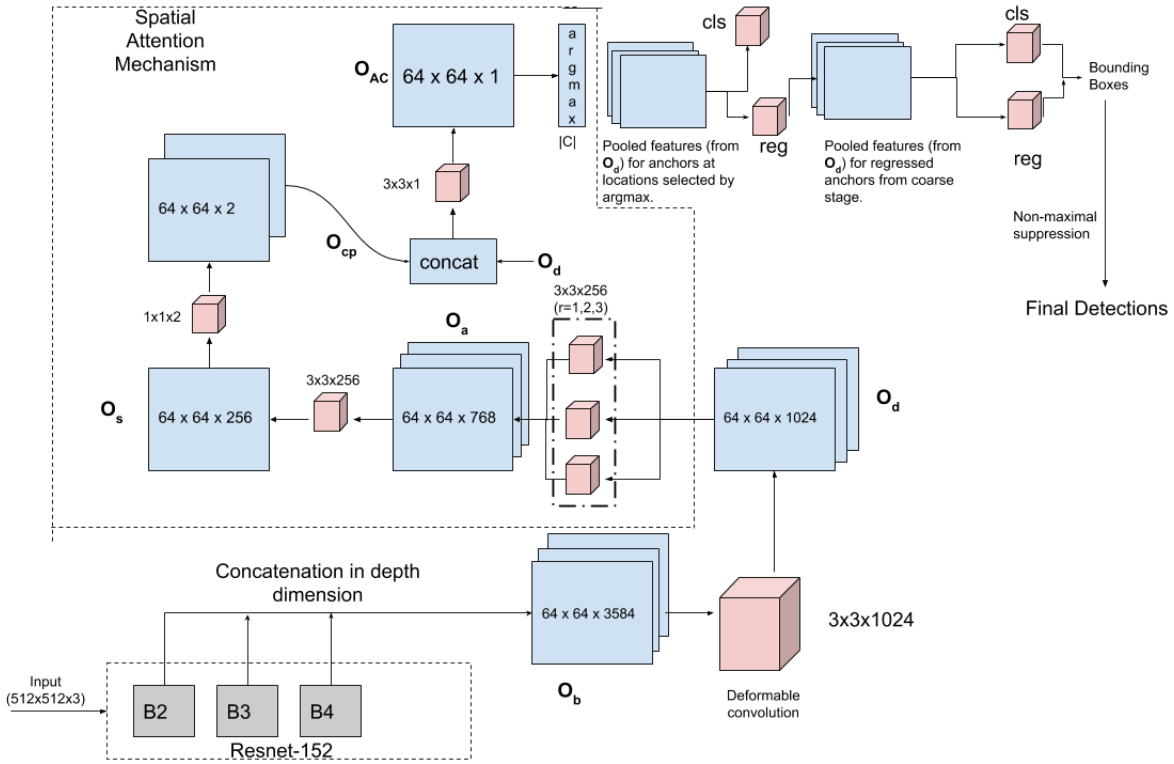


Figure 6.1: Block diagram of the approach proposed in chapter 5. Reproduced here for simplicity and comparison with the approach presented here in figure 6.2 .

has $2\times = 18$ channels (*the multiplicative factor accounting for the offsets in* x *and* y *directions and therefore* 2). As illustrated in figure 6.3, the convolutional layer output in the offset branch consists of the offsets for all the locations in the feature map and therefore is used to compute the deformable convolution output.

Computationally, this operation has a time complexity higher than that of normal convolution. Let N_i be the number of input channels to a convolutional layer, N_o be the number of output channels, $K \times K$ be the kernel size and $M \times M$ be the size of the output feature map. The time complexity of the convolutional operation can then be computed as in [64] to be:

$$\mathcal{O}(N_i \times K^2 \times N_o \times M^2) \tag{6.1}$$

It can be seen therefore that following the same notations as above, the time complexity of a $K \times K$ deformable convolutional operation is as follows

$$\mathcal{O}(N_i \times K^2 \times N_o \times M^2) + \mathcal{O}(N_i \times K^2 \times 2 \times K^2 \times M^2) + \mathcal{O}(I) \tag{6.2}$$

In equation 6.2, the last term ($\mathcal{O}(I)$) represents the time complexity of other operations

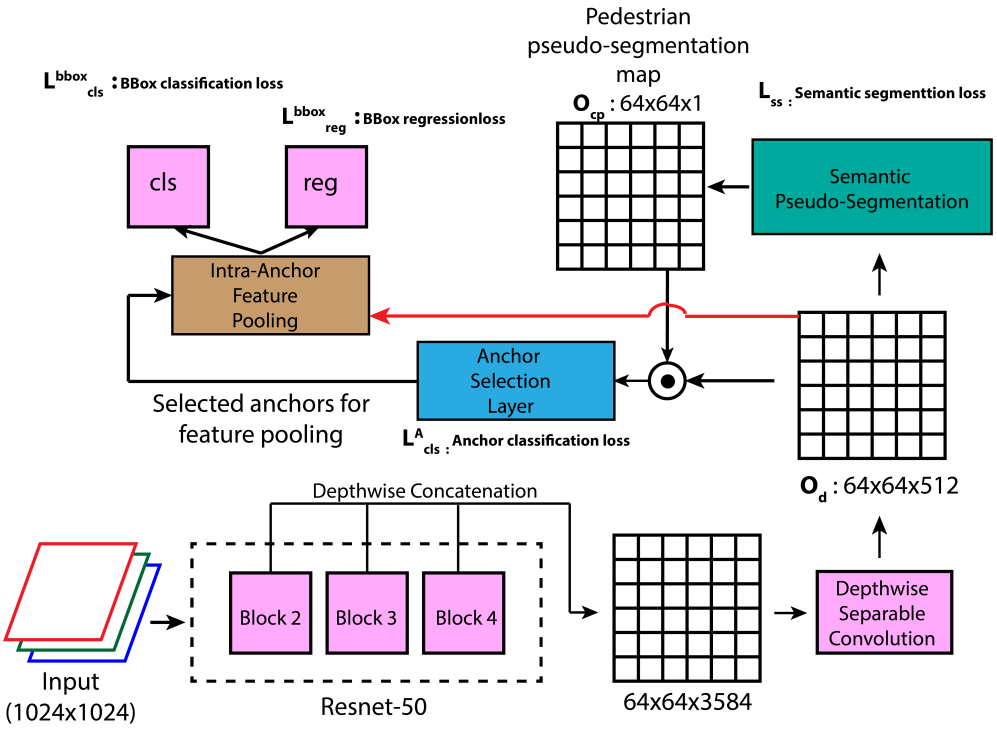


Figure 6.2: Block diagram of the proposed approach. This block diagram differs from figure 6.1 in terms of many refinements which allow for better inference without loosing detection accuracy.

such as computing bilinear interpolation from offset values.

Therefore, in applications such as high speed automotive applications, where near real-time performance is critical, deformable convolution can be a major computational limiting module. While we have discussed its time complexity, its space complexity is also greater than normal convolution on account of a separate offset branch and bilinear interpolation.

One of the major refinements in the proposed system of figure 6.2 is the replacement of deformable convolutional operation with depthwise separable convolution. Due to the early fusion approach adopted in our work, the number of output feature channels from the base network is very large. As seen in equation 6.1, the time complexity is proportional to the number of feature channels (*both input and output*). Therefore, a straightforward purging of the deformable convolutional layer is not desirable. We instead adopt a depthwise separable convolutional layer [25], which first processes the input feature map channel wise and then fuses the individual channel outputs using a 1×1 convolution to obtain the final output feature map. The time complexity of deformable convolution is substantially lower than that of normal convolution and hence is much more efficient than deformable convolution as well. Using the same notations as in equations

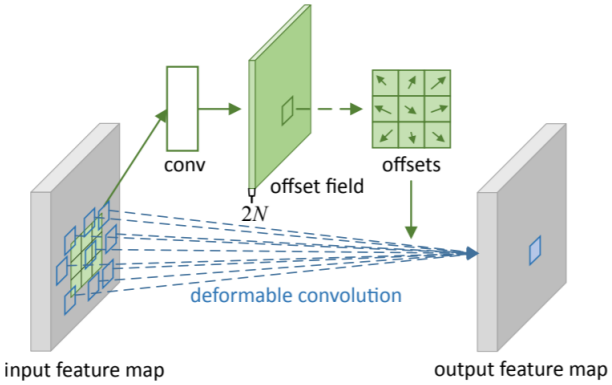


Figure 2: Illustration of 3×3 deformable convolution.

Figure 6.3: Illustration of offset calculation for a 3×3 deformable convolution operation.

6.1 and 6.2, the time complexity of deformabel convolutional operation is

$$\mathcal{O}(N_i \times K^2 \times M^2 + N_i \times N_o \times M^2) \tag{6.3}$$

Channelwise processing helps in better amalgamation of information in different feature channels. This is much better than normal convolution or even deformable convolution in the sense, that in those operations, all the feature channels are processed by the same convolutional filter. This is less desirable as the feature channels in the output of early fusion come from hierarchically different convolutional layers and hence contain information of varying nature.

6.3.2 Concatenation of pedestrian probability map and deformable convolution output

In figure 6.1, it can be seen that there is a concatenation along the channel dimensions of $\boldsymbol{O_d}$ and $\boldsymbol{O_{cp}}$. Elementwise multiplication in place of concatenation is another possibility that could be employed here. Concatenation in figure 6.1 was favored on following qualitative account.

In cases where the semantic segmentation module is in lack of perfection, i.e fails to localize very small pedestrians, a direct multiplication in place of concatenation would result in a feature map which is completely devoid of the information corresponding to very small pedestrians.

However, concatenation results in an added extra channel. While this does not incur a huge computational penalty, our observations reveal that semantic segmentation module through adequate training is indeed good at localizing nearly all pedestrians.

Dataset	IoU(0.5)	IoU(0.2)
Caltech1x-Test	0.89	0.97
CityPersons (Validation set)	0.93	0.98

Table 6.1: Intersection-over-union analysis of the performance of semantic segmentation module.

Table 6.1 summarizes the intersection over-union performance of the semantic segmentation module. To conduct this analysis $\boldsymbol{O_{cp}}$ was thresholded for all values $v \geq \{0.5, 0.2\}$ and their IoU computated against the **PSM** of the pedestrians from the dataset. This analysis suggests that the performance of semantic segmentation module is quite competitive. For a threshold value of 0.2, the IoU performance is near perfect. This suggests that the elementwise multiplication of $\boldsymbol{O_d}$ and $\boldsymbol{O_{cp}}$ does not rob away the small-scale pedestrian features, while their absolute strength might be affected for lower values in $\boldsymbol{O_{cp}}$.

Guided by these observations, we go about the route of elementwise multiplication (also known as *schur product* in the linear algebra literature). In figure 6.1, we use a 3×3 convolutional layer to further process the concatenated feature map. We purge this convolutional layer away and directly use the schur product output as the input to the anchor selection layer.

6.3.3 Two-Step classification and regression

The two-step classification and regression process adopted in figure 6.1, provides around 0.86% of performance boost compared to using only one classification and regression step. At the same time, its impact on computational time is significant as the number of intra-anchor feature pooling operations approximately double because of two steps. This increase in pooling operations double only approximately, as only the positively classified and regressed bounding boxes from the first step are used for feature pooling by the subsequent step. Due to the smaller magnitude of benefit reaped from using a two-step classification and regression process, we decide in favor of a single step classification and regression process. This results in significantly fewer pooling operations and therefore promotes faster training and inference.

6.3.4 Input Size

The input size to a network in object detection is of significant importance. Various existing pedestrian detection approaches such as MSCNN [16] and F-DNN [36] show that inference and training with increased image sizes results in performance boost of upto

$2-3\%$ [16]. We hence decide to use an input image size of 1024×1024. This input image size helps in improving the visibility of small-scale pedestrians in caltech dataet [34], while providing minimal distortion to the pedestrians in the citypersons dataset [179].

Having described the computational aspects of various system design approaches in figure 6.1, we now proceed to describing the details of the refinements introduced and compare the new architecture (figure 6.2) with that in figure 6.1.

6.4 Proposed Approach

Since the present approach is a simplification and refinement of the approach presented in chapter 5, for the purpose of brevity we limit ourselves only to some characteristic differences between the two approaches. We first outline the summary of the proposed approach followed by a detailed explanation of the anchor selection layer.

6.4.1 Summary

In this section, we describe the essential differences in the proposed pipeline (figure 6.2) from that in figure 6.1. The input minibatch is processed by a ResNet-50 base network and after early fusion is processed by a depthwise separable convolution operation. The semantic segmentation module (figure 5.12 and section 5.8.4) is used to generate a pseudo-segmentation map which is a pixelwise probability of pedestrian presence in an image. A schur product of $\boldsymbol{O_d}$ and $\boldsymbol{O_{cp}}$ is then fed to an anchor selection layer. The anchors selected from the anchor selection layer are used for classification and bounding box regression. The anchors are cropped and resized to 14×14 using bilinear interpolation followed by max-pooling to obtain a $7 \times 7 \times 64$ feature map which is flattened and fed to two sibling 1×1 convolutional layers for classification and bounding box regression. Softmax cross entropy is used for training classification (*pedestrian* vs. *non-pedestrian*), while the parametric formulation used in [130] is used in conjunction with smooth-L1 loss for bounding box regression. We use *non-maximal-suppression* (NMS) over the predictions to get final detections.

6.4.2 Anchor Selection Layer

We select the top C locations in $\boldsymbol{O_{cp}}$ for further processing. If N_A is the number of anchors at each location, this requires $C \times N_A$ anchors to be feature pooled. For high values of C (e.g:-1000), this is time-consuming and further requires a large memory footprint. Furthermore, not all anchors encompassing pedestrians are useful; as shown in figure 6.4, where only the magenta anchor overlaps sufficiently well with both full-body and visible part of the bounding box. An anchor overlapping well with both the full body and visible

Figure 6.4: An occluded pedestrian with full-body bounding box (**green**) and visible bounding box (**blue**). **Red** anchors are confocal with the **magenta** anchor. The **red** anchors do not overlap well with both the full-body and visible bounding box, while the **magenta** anchor has sufficient overlap with both.

part of the bounding box encapsulates information about the pedestrian and the occlusion and is thus more useful than other anchors. We therefore stipulate that based on the overlap criteria it is possible to further restrict the number of anchors to be processed. To handle test cases with no prior bounding boxes available, we use a classification strategy to learn good anchors, as described below.

The input to the anchor selection layer is $\boldsymbol{O_h}$, which is the schur product of $\boldsymbol{O_{cp}}$ and $\boldsymbol{O_d}$ (broadcasted across the feature channel). A set of N_A sibling classification branches are set up. Each classification branch serves the classification of anchors with a specific scale and aspect ratio. The i^{th} classification branch is constituted of a convolutional layer with 32 filters of size $h_i \times w_i$, followed by a $1 \times 1 \times 2$ convolutional layer, which is then followed by a softmax operation to determine the probability of an anchor to be *positive*. $h_i \times w_i$ is determined by the configuration used for generating anchors. An example is shown in figure 6.5.

Each classification branch during training is trained using focal loss for classification. for an anchor A, its IoU with the full-body bounding box B_F and visible bounding box B_V is computed and the class of the anchor is determined as *positive* or *negative* as follows

$$class(A) = \begin{cases} +ve, & \text{IoU}(A,B_V) \geq \alpha; \text{IoU}(A,B_F) \geq \beta \\ -ve & \text{otherwise} \end{cases}$$

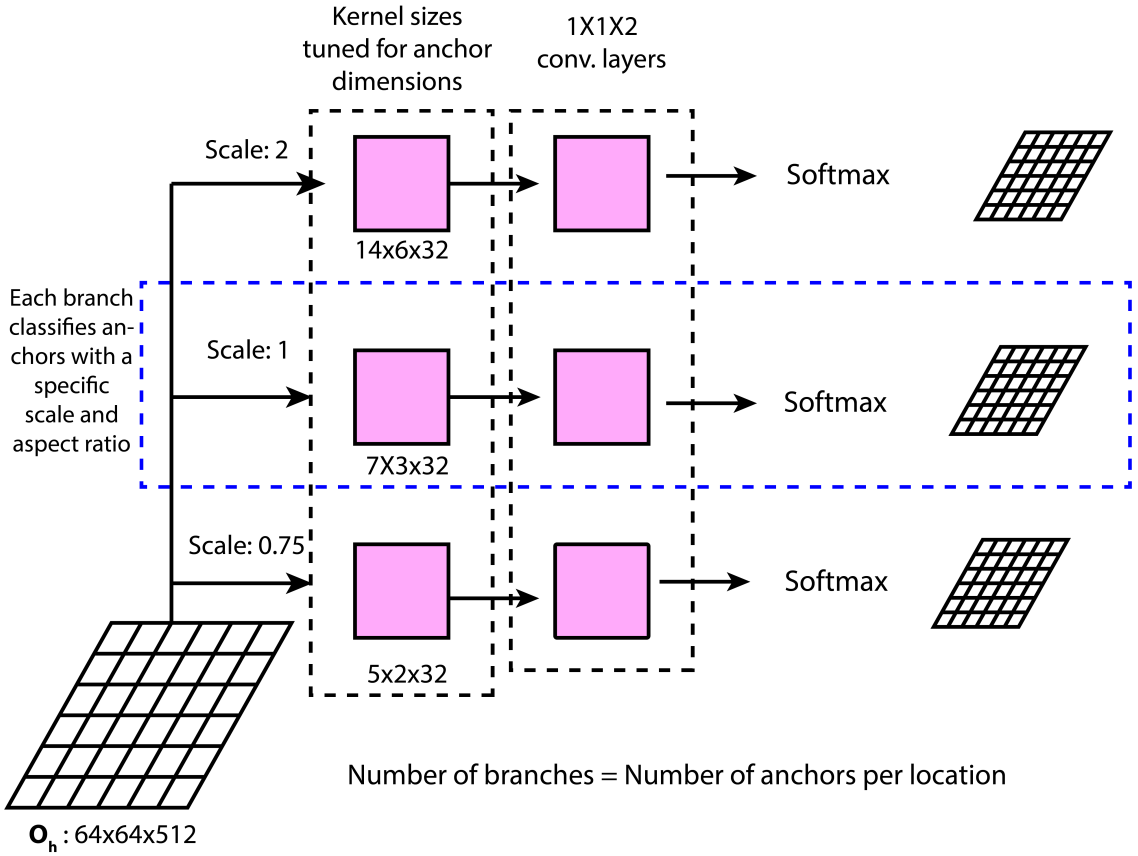


Figure 6.5: The anchor selection layer. For illustration it is assumed that all anchors have been generated by a base anchor of size 64×64 and have an aspect ratio of 0.41 (*width/height*). The anchor at scale 1 then corresponds to a box of size $\sim 100 \times 41$. For a feature stride of 16, a kernel of size 7×3 will cover the corresponding area of this box in the feature map. For other scale values, the kernel size can be similarly defined.

Behavior during training and testing : During training, we utilize both positively and negatively classified anchors, while only *positively* classified anchors are used during testing. Let n_{pos} be the number of *positive* predicted anchors during training. Then the number of negative anchors used is defined as follows :

$$n_{neg} = min(5 \times n_{pos}, M - n_{pos}) \tag{6.4}$$

where, M is a hyperparameter and is usually limited by the memory capacity of the computing device. During testing phase, we use only the *positive* anchors for classification and regression.

This difference in behavior is preferred so that during training, a large range of samples encompassing *positive* and *negative* examples are seen by classifier and regressor for robust learning.

6.5 Training and Implementation Details

Training set	Testing set	Testing set LAMR (%) (IoU : 0.5)	Testing set LAMR (%) (IoU : 0.75)
caltech-reasonable (train)	**caltech-reasonable (test)**	4.76	18.16
citypersons(train) + caltech-reasonable (train)	**caltech-reasonable (test)**	3.99	17.62
citypersons(train)	**citypersons (test)**	8.12	25.5

Table 6.2: LAMR of our proposed approach over caltech-reasonable and citypersons(val) datasets. Our best results on caltech-reasonable (test) are achieved through pre-training on citypersons(train) dataset.

We implemented our proposed approach on top of tensorflow object detection api [74]. We used stochastic gradient descent (SGD) with a momentum value of 0.9 as an optimizer. Our initial learning rate was set to 0.01 with gradients clipped to a value of 10.0. This warm-up phase lasted for $10K$ iterations after which the learning rate was decreased by a factor of 10 after every $30K$ iterations. Data augmentation was used for training in the form of – a) random horizontal flipping, b) random brightness adjustment and c) random contrast adjustment. We upscaled all images using bilinear interpolation to a fixed size of 1024×1024.

6.5.1 Loss Function

Our loss function can be written as

$$L_{total} \triangleq L_{ss} + L_{cls}^{A} + L_{cls}^{bbox} + L_{reg}^{bbox} + L_{2}^{reg} \tag{6.5}$$

where,

1. L_{ss} : Average pixelwise cross entropy term for semantic segmentation.

2. $L_{cls}^{A\ i}$: Cross entropy term for anchor classification. It is a sum of N_A terms, where N_A is the number of confocal anchors.

3. L_{cls}^{bbox} : Average cross entropy term for bounding box classification.

4. L_{reg}^{bbox} : Average smooth L1-loss term as defined in [130].

5. L_{2}^{reg} : Average of regularization terms elsewhere in the network.

6.6 Experiments, Results and Analysis

6.6.1 Datasets

We used caltech-reasonable [34] and citypersons [179] for validating the performance of the proposed approach. In the citypersons dataset's evaluation protocol, there are 4

	Caltech-Reasonable		**CityPersons**	
	Train	**Test**	**Train**	**Val**
Images	42,782	4024	2,975	500

Table 6.3: Summary of dataset size of caltech-reasonable [34] and citypersons [179] dataset.

Method	**Stages**	**LAMR**		**Speed**
		caltech-reasonable (test) (w/o CP pre-training) (CP pre-trained)	**citypersons (val) (trained only on CP)**	
Faster-RCNN [130]	2	12.10	15.4	7
SSD [102]	1	17.78 (16.36)	19.69	48
YOLOv2 [127]	1	21.62 (20.83)	NA	60
RPN-BF [177]	2	9.6 (NA)	NA	7
MS-CNN [16]	2	10.0 (NA)	NA	8
SDS-RCNN [14]	2	7.6 (NA)	NA	5
ALF-Net [103]	1	4.5 (NA)	12.0	20
Rep-Loss [166]	2	5.0 (4.0)	13.2	-
Ours	1.5	**4.76 (3.99)**	**8.12**	**32**

Table 6.4: Performance comparison of the proposed method with other methods for caltech-reasonable test set and citypersons validation set. The speed figures are in *frames per second*.

distinct categories of evaluation – *pedestrian*, *rider*, *sitting person* and *person (other)*. We cluster these 4 sub-categories are into one and refer to it as *pedestrians*.

6.6.2 Hyperparameter settings

Unless and otherwise mentioned, we use parameter settings as mentioned henceforth. Following [177, 14, 16], we use anchors with an aspect ratio of 0.41 (*average aspect ratio of bounding boxes in caltech and citypersons datasets*). We use 6 anchor scales ($\{0.25, 0.5, 0.75, 1, 2, 4\}$), generated from a base anchor of size $(64, 64)$. For anchor selection layer (sec 6.4.2), $\alpha = 0.3$, $\beta = 0.5$, $M = 2000$. $C = 1000$ in our work.

6.6.3 Results and Analysis

Table 6.2 summarizes the LAMR of our proposed approach on caltech-reasonable and citypersons datasets. Table 6.2 summarizes the comparative performance of our proposed approach with other pedestrian detectors vis-á-vis accuracy and speed on caltech-reasonable and citypersons datasets. We achieve state-of-art performance on both caltech-reasonable and citypersons datasets. Our best results on caltech-reasonable is obtained by first pre-training our model on citypersons training set, followed by fine-tuning it on caltech-reasonable training set. This amounts to 0.77% improvement in LAMR. From table 6.2, we see that our improvements on citypersons is much higher than on caltech-reasonable. The input to our model is 1024×1024, which results in higher distortion for caltech-reasonable images (640×480) than citypersons images (2048×1024). This lower distortion gives better training to our model and explains higher improvements on the citypersons dataset. We attribute the slight degradation in detection accuracy compared to the approach presented in chapter 5 (where LAMR on caltech-reasonable test set is 3.79%), to the use of a shallower base network, non-usage of deformable convolutional layer and non-usage of a cascade of classification and regression stages. However, this degradation is small (0.20% on the caltech reasonable and 0.93% on the citypersons validation set), while achieving a higher detection speed.

With an optimized implementation, our approach performs inference at ~ 32 fps, on input images of size 1024×1024, which is ~ 1.5 times faster than the next best speeds (~ 20 fps) achieved by [103]. It is to be noted that we report our inference speed only for the inference evaluation and do not factor in the time for any display or disk access. It is further notable that the speeds reported in [103] are for images of size 480×640, while for us all images are of size 1024×1024. This shows a major speedup attained by our approach vis-á-vis other competitive methods.

To the best of our knowledge, ours is the first *two-stage* detector achieving a high detection accuracy while also providing a high inference speed. Accuracy-wise our method competes closely with Rep-Loss [166] and ALF-net [103], while being faster than both of them.

The above improvements in inference speed and accuracy in our approach are largely possible through a conjunction of semantic segmentation layer and anchor selection layer. The semantic segmentation layer provides high quality segmentation maps, which assist in selecting a small set of locations at which all anchors are classified. The anchor selection layer then further reduces this count by classification. As a result, the number of anchors used during testing phase is proportional to the number of pedestrians in the image.

6.7 Ablation Studies

6.7.1 Comparison with Region Proposal Network

LAMR	
RPN	**Semantic Segmentation + Anchor selection layer**
15.18	**8.12**

Table 6.5: Ablation study of RPN vs. our approach using semantic segmentation and anchor selection layer on the citypersons(validation) dataset.

To know the impact of our first stage using semantic segmentation and anchor selection layers, we replace our first stage with standard RPN layer of Faster-RCNN [130], with the same anchor parameters as reported in section 6.6.2, and using top 300 proposals for processing. From table 6.5 shows that our first stage has a huge impact on performance. This performance boost speaks jointly of the improvements brought about by semantic segmentation and anchor selection layer. Moreover, our first stage has computational advantages over RPN as shown below.

In the RPN, for a feature map with 512 channels, and 6 confocal anchors, the classifier will admit $512 \times 2 \times 6$ parameters for *foreground/background* classification. The RPN regressor will admit $512 \times 4 \times 6$ parameters for 4 quantities regressed for each anchor box. This leads to a total of $512 \times (4 + 2) \times 6 = 18432$ trainable parameters. This does not include the parameter count of proposal filters and feature projection layer in RPN [130]. In contrast, the union of segmentation and anchor classification modules admit a total of 7048 training parameters in our approach. Thus, compared to a basic RPN with no extra proposal filter and feature projection layers, our approach admits 2.61 times less parameters than RPN. This makes learning easier for our system on smaller pedestrian datasets like citypersons [179] (2975 training images).

6.7.2 Impact of anchor selection layer

Removing the anchor selection layer (sec 6.4.2) from the pipeline, compels one to use all possible anchors at all the top C locations in $\boldsymbol{O_{cp}}$. Since, the total number of anchors one may use in a batch is limited by the memory limit of the computing device, large values of C (e.g:- 1000) were no longer admissible on our systems. In the proposed approach, the anchor selection layer, makes it possible to use large values of C, as classification lets one choose a set of positive and negative anchors during training (equation 6.4). We experimented with $C = 300$ (resulting in number of anchors as $300 \times 6 = 1800$). This dropped the LAMR on the caltech-reasonable dataset (after pre-training on citypersons) to 9.57%. For a fair comparison our proposed approach was also tried with $C = 300$,

resulting in a LAMR of 7.37. This corresponds to $\sim 2.20\%$ improvement brought upon by the anchor selection layer.

6.8 Conclusions

In this chapter we discussed an architectural refinement over the approach presented in chapter 5. The major refinement over the system in chapter 5 proposed in this chapter is that of an anchor selection layer. The anchor selection layer uses anchor specific kernels (*kernels with shape matching the anchor shapes*) to select anchors for further processing. A related refinement is that of the use of both visible and full body bounding boxes in terms of their overlap criterion with the anchors during training to guide the anchor selection layer. Other refinements such as the use of depthwise separable convolution in place of deformable convolution enable faster feature processing with lesser number of parameters. These refinements greatly improve upon the detection speed of pedestrian detection across compared to chapter 5. The detection accuracy suffers a minor breakdown and we infer it owing to the use of a shallower base network as well as non-usage of some computationally heavy layers such as deformable convolution and cascade of classification and regression stages.

Chapter 7

Conclusion and Future Work

In this thesis we focus on simultaneously improving the detection accuracy and detection speed of deep learning based pedestrian detection systems. The relevance of accurate and real-time detection is essential for autonomous vehicles especially when maneuvering at high speeds; and this forms the main motivation for our work. With a number of deep learning based detection systems being available; our work begins with the quantitative analysis of their accuracy and inference speed performance. This initial analysis presented in chapter 3 shows that one has at their disposal a large number of tools from deep learning, which can be employed to improve existing systems. As a part of our analysis we conduct experiments with a number of different refinements such as using various network architectures and using different types of convolutional layers. These refinements have primarily been proposed and validated on image classification problems and to the best of our knowledge, their systematic quantitative analysis has not been conducted before on pedestrian detection. Our initial analysis therefore, lends one a repertoire of information related to impact of various architectural refinements, as well as training and fine-tuning strategies, which can aid one in designing a custom pedestrian detection system. Though, we have primarily focused upon Faster-RCNN [130], SSD [102], RPN-BF [177] and SDS-RCNN [14], our analysis can be extended to any other pedestrian detection system on account of its generality. Refinements considered in chapter 3, such as base network architecture, loss function impact and impact of various types of convolutional techniques such as à trous and depthwise separable convolution are applicable to existing as well as future deep learning based pedestrian detection systems. This analysis has motivated the design of pedestrian detection systems proposed in the subsequent chapters of this thesis; both in terms of architectural refinements and training and fine-tuning strategies.

Detection of pedestrians across scales and occlusion levels is an important problem. In deep neural networks, pooling operations reduce feature map dimensions, thereby often reducing information of small-scale and highly occluded pedestrians to a sub-pixel level.

Furthermore, different CNN layers are known to learn semantically different features. This motivates us to focus on the important concept of convolutional layer selection which has been outlined in chapter 3. The topic of convolutional layer selection refers to – "The optimal approach to utilize multiple convolutional layers simultaneously for pedestrian detection.". The idea of utilizing multiple convolutional layers has been stressed several times in literature before, but to the best of our knowledge, their quantitative impact on pedestrian detection across scales and occlusion levels has never been done before. In chapter 4, we conduct this analysis and hence quantitatively demonstrate that early layers are more suitable for detecting small-scale and highly occluded pedestrians, while later layers are more suitable for detecting large-scale and unoccluded pedestrians. We also note in chapter 4, that tree based ensemble classifiers are better at handling classification problems with a large intra-class variance. In a previous work known as RPN-BF [177], boosted forests are used for classification of deep learning features. We compliment the approach of RPN-BF in two ways – a) *early* and *late* fusion of multiple convolutional layers which demonstrates that early fusion is a much better strategy than late fusion for both detection accuracy and detection speed and, b) use of gradient boosted trees (GBT) for classification. We show that the use of GBTs which utilize the concept of boosting to grow decision trees, improve the performance under early fusion over both Faster-RCNN [130] (by $\sim 5\%$ LAMR) and RPN-BF [177] (by $\sim 1.5\%$ LAMR). Although seemingly small, this improvement corresponds to correct detection of nearly 3800 previously undetected pedestrians. GBTs however are difficult to train jointly with CNNs and existing attempts at doing so have not received a major performance improvement in image classification problems [69]. Features first need to be extracted from a CNN followed by feeding them to a GBT – this transfer being difficult to achieve fastly due to implementation differences between modern GBT and CNN frameworks. This also causes the inference speed to go down to around $5 - 7$ fps. This detection speed is not reasonable when most cameras in use today operate at around 30 fps. For a vehicle travelling at high speeds, this low speed can pose dangers to both the vehicle and the pedestrians. This motivates us to further achieve a better balance between detection speed and accuracy, while utilizing multiple convolutional layers.

In response to our conclusions in chapter 4, take cue from the use of semantic segmentation in RPN-BF [14] and utilize that idea in chapter 5. However, unlike RPN-BF, we do not limit ourselves to improve the feature representation, but go on to reduce the number of anchors to be processed. We set the relevance of this idea in chapter 5, where we provide the details of the feature handling mechanisms of one-stage and two-stage detectors. More explicitly, we show that feature pooling is the primary trait which delivers better performance but slower speed to two-stage detectors such as Faster-RCNN [130], SDS-RCNN [14] and RPN-BF [177]. Semantic segmentation module in our proposed system

focuses on the pedestrians. This enables us to select a small set of regions which reduce the number of feature pooling operations in the detection stage. Our work differs significantly from other region selection based schemes like Faster-RCNN [130], which perform bounding box regression of anchor boxes. We only select a set of locations from where we select all anchors centered at those locations as regions without any bounding box regression. classification and bounding box regression is performed only in the second stage using a cascade of classification and regression modules. The proposed cascade is simpler than Cascade-RCNN [18] and can be trained end-to-end. With our approach to reduce the number of anchors to be processed, we obtain the state-of-art performance on caltech (3.79% LAMR) and citypersons dataset (7.19% LAMR) while maintaining a high inference speed (~ 20 fps). Usually modern detection systems use a large number of confocal anchors. Despite performing at the state-of-art level, the proposed approach can come under heavy load when a large number of confocal anchors are involved. Use of a cascade of two classification and regression modules implies that feature pooling is done two times over. Thus it is challenging to deploy the proposed approach on memory-constrained devices.

To overcome the limitations of the approach presented in chapter 5, we propose an anchor selection layer in chapter 6. The anchor selection layer further reduces the number of anchors to be processed. The anchor selection layer utilizes anchor specific kernel sizes for classification. This allows one to classify an anchor based on the entirety of information contained therein; but without using the costly operation of feature pooling. We further remove the cascade of classification and regression modules, thereby reducing the number of feature pooling to half. This proposed approach, allows us to obtain the highest inference speed of ~ 40fps while maintaining the state-of-art performance on both citypersons (8.14% LAMR) and caltech reasonable (3.99% LAMR) datasets.

While, we have been successful in settling the problem of balancing the trade-off between detection accuracy and detection speed, there are many impending challenges which need to be addressed. In a recent review [170], the human level performance for pedestrian detection on the caltech-reasonable test set is shown as 0.88% LAMR. Our current state-of-art results on the same dataset is 3.79% LAMR. This difference of 2.91% LAMR is significant to be bridged to ensure successful systems which perform at-par with human performance. Human beings are good at detecting pedestrians under poor illumination and weather conditions including wide ranges of scales and occlusion levels. Work in these directions is critical for bridging the gap to human level performance. The human level performance is an important objective for critical applications where human safety is directly involved such as in self-driving vehicles. One of the traditional bottlenecks in pedestrian detection research is unavailability of large scale datasets which reflect the wide range of illumination and weather conditions. Recently there has been some progress in this direction with the release of datasets such as BDD100K [174] and nightowls dataset

[112], which address these needs. Technically, poor weather and illumination lead to poor feature extraction to the lack of gradient information in such conditions due to poor visibility. Therefore an important future work in this direction is to evaluate the existing techniques on newer datasets like BDD100K and nightowls; thereby paving way to understanding the behavior of CNNs on pedestrian feature extraction. In practical instances, a detection system is expected to perform irrespective of illumination and weather conditions. Therefore work on determining and improving the generality of feature extracted from CNNs is an important direction of research.

Fine-tuning of an existing pedestrian detection system to new scenarios is another important challenge. We have covered this in chapter 3, and have made some important observations about the impact of image resolution on pre-training and fine-tuning. However, in practice other constraints may also apply. For example, when fine-tuning a system to images coming from a new camera; the new camera may have its own camera-specific aberrations (*due to different sensors*) and field-of-view. There does not appear to be an adequate study of fine-tuning strategies under these scenarios. With the growing proliferation of pedestrian detection systems in a world, where the number and quality of cameras is ever-changing, it is pertinent to deeply study these aspects of pedestrian detection systems.

Human beings make use of a multitude of information when detecting pedestrians in real life situations. Information about pedestrian motion, contextual information about the environment, action of a pedestrian and prediction of the immediate future behavior of a pedestrian are some common traits exploited during driving maneuvers in real life. Work on aforementioned traits has been and is being done, though in an isolated setting. As an example, it is vital to predict the future locations of other vehicles and pedestrians; something done effortlessly by human beings. While works such as social-LSTM [2], AMIR [131], P2F [162] exist on multi-target visual prediction, an amalgamation of these ideas with detection and action recognition is necessary to simulate the functionalities employed by human beings when driving in real environments. It is therefore, pertinent to work on multi-task systems for pedestrian detection, where information from multiple sources such as action recognition and future prediction are fused to make a final determination of pedestrian locations. Most of the contemporary pedestrian detection systems do not incorporate such multi-task solutions. Work in this direction can be facilitated by publicly available datasets with annotations for such traits. VIRAT video dataset [115], though not suitable for applications such as autonomous driving is an encouraging template dataset, which can be followed by potential future releases of similar dataset suited for autonomous applications. VIRAT dataset is labeled with human activities apart from pedestrian bounding boxes and is provided in video form, thereby making it possible to use motion information. Recently proposed EuroCityPersons dataset [13] having come

with annotations describing pedestrian motion orientation is a good step in this direction.

All the pedestrian detection analysis and proposals in this thesis related to the fully-supervised setting. In a fully-supervised setting, all the pedestrians in an image need to be annotated for proper learning. Collecting high-quality annotations of pedestrians is a costly endeavor, requiring a lot of time and training for the annotators. Furthermore, inter-annotator variance is an important challenge [89] when conducting large-scale annotations in computer vision. While availability of images and videos is an easy task today, their annotations is not. This has led to the development of several weakly and semi-supervised approaches for detection problems as well. In the semi-supervised scenario, only a limited number of pedestrians may be labelled in an image, instead of all. In the weakly supervised setting, there is no localization information for pedestrians, only an indication of their presence of absence. It is an attractive direction of research which can have major ramifications on the cost-effectiveness of pedestrian detection systems. Most of the existing work in these two settings have been carried out in general-category object detection. To the best of our knowledge, there is no existing literature on pedestrian detection which caters to weakly or semi-supervised settings. We consider this to be the next major helm of pedestrian detection research owing to its practical impact.

Bibliography

[1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016), pp. 265–283. (Cited on page 95.)

[2] ALAHI, A., GOEL, K., RAMANATHAN, V., ROBICQUET, A., FEI-FEI, L., AND SAVARESE, S. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 961–971. (Cited on page 120.)

[3] ALEXE, B., DESELAERS, T., AND FERRARI, V. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence 34*, 11 (2012), 2189–2202. (Cited on pages 16 and 17.)

[4] ALI, A., AND DAGLESS, E. Vehicle and pedestrian detection and tracking. In *IEE Colloquium on Image Analysis for Transport Applications* (1990), IET, pp. 5–1. (Cited on page 1.)

[5] ANGELOVA, A., KRIZHEVSKY, A., VANHOUCKE, V., OGALE, A., AND FERGUSON, D. Real-time pedestrian detection with deep network cascades. (Cited on page 16.)

[6] BAEK, J., HYUN, J., AND KIM, E. A pedestrian detection system accelerated by kernelized proposals. *IEEE Transactions on Intelligent Transportation Systems* (2019). (Cited on page 16.)

[7] BAI, S. Growing random forest on deep convolutional neural networks for scene categorization. *Expert Systems with Applications 71* (2017), 279–287. (Cited on page 57.)

[8] BELL, S., LAWRENCE ZITNICK, C., BALA, K., AND GIRSHICK, R. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2874–2883. (Cited on page 27.)

[9] BENENSON, R., MATHIAS, M., TIMOFTE, R., AND VAN GOOL, L. Pedestrian detection at 100 frames per second. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 2903–2910. (Cited on pages 76 and 83.)

[10] BENENSON, R., OMRAN, M., HOSANG, J., AND SCHIELE, B. Ten years of pedestrian detection, what have we learned? In *European Conference on Computer Vision* (2014), Springer, pp. 613–627. (Cited on page 39.)

[11] BLOMBERG, R. D., LEAF, W. A., AND JACOBS, H. H. Detection and recognition of pedestrians at night. *Transportation Research Circular 229* (1981), 17–21. (Cited on page 1.)

[12] BODLA, N., SINGH, B., CHELLAPPA, R., AND DAVIS, L. S. Soft-nmsâimproving object detection with one line of code. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017), IEEE, pp. 5562–5570. (Cited on page 58.)

[13] BRAUN, M., KREBS, S., FLOHR, F., AND GAVRILA, D. M. The eurocity persons dataset: A novel benchmark for object detection. *arXiv preprint arXiv:1805.07193* (2018). (Cited on pages 38 and 120.)

[14] BRAZIL, G., YIN, X., AND LIU, X. Illuminating pedestrians via simultaneous detection & segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 4950–4959. (Cited on pages 3, 4, 16, 17, 25, 26, 31, 33, 38, 39, 41, 48, 49, 50, 52, 53, 56, 57, 62, 64, 65, 68, 71, 77, 78, 79, 83, 84, 94, 95, 96, 99, 103, 112, 117 and 118.)

[15] CAI, Z., FAN, Q., FERIS, R., AND VASCONCELOS, N. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV* (2016). (Cited on pages 25, 50 and 56.)

[16] CAI, Z., FAN, Q., FERIS, R. S., AND VASCONCELOS, N. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision* (2016), Springer, pp. 354–370. (Cited on pages 6, 17, 26, 27, 28, 33, 71, 77, 78, 86, 94, 95, 96, 102, 103, 107, 108 and 112.)

[17] CAI, Z., SABERIAN, M., AND VASCONCELOS, N. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3361–3369. (Cited on pages 16 and 83.)

[18] CAI, Z., AND VASCONCELOS, N. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 6154–6162. (Cited on pages 31, 78, 99 and 119.)

[19] CANZIANI, A., PASZKE, A., AND CULURCIELLO, E. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678* (2016). (Cited on pages 26 and 47.)

[20] CAO, J., PANG, Y., AND LI, X. Learning multilayer channel features for pedestrian detection. *IEEE transactions on image processing 26*, 7 (2017), 3210–3220. (Cited on pages 27 and 29.)

[21] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence 40*, 4 (2018), 834–848. (Cited on pages 39 and 47.)

[22] CHEN, L.-C., PAPANDREOU, G., SCHROFF, F., AND ADAM, H. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587* (2017). (Cited on pages 47 and 90.)

[23] CHEN, X., AND GUPTA, A. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138* (2017). (Cited on pages xi, 6, 21 and 22.)

[24] CHENG, M.-M., ZHANG, Z., LIN, W.-Y., AND TORR, P. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 3286–3293. (Cited on page 16.)

[25] CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1251–1258. (Cited on pages 26, 39, 48, 50, 56 and 105.)

[26] COHEN, S. A., AND HOPKINS, D. Autonomous vehicles and the future of urban tourism. *Annals of tourism research 74* (2019), 33–42. (Cited on page 2.)

[27] COMBS, T. S., SANDT, L. S., CLAMANN, M. P., AND MCDONALD, N. C. Automated vehicles and pedestrian safety: exploring the promise and limits of pedestrian detection. *American journal of preventive medicine 56*, 1 (2019), 1–7. (Cited on page 2.)

[28] CORDTS, M., OMRAN, M., RAMOS, S., SCHARWÄCHTER, T., ENZWEILER, M., BENENSON, R., FRANKE, U., ROTH, S., AND SCHIELE, B. The cityscapes dataset. In *CVPR Workshop on the Future of Datasets in Vision* (2015), vol. 1, p. 3. (Cited on page 33.)

[29] DAI, J., LI, Y., HE, K., AND SUN, J. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (2016), pp. 379–387. (Cited on page 102.)

[30] DAI, J., QI, H., XIONG, Y., LI, Y., ZHANG, G., HU, H., AND WEI, Y. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 764–773. (Cited on page 88.)

[31] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)* (2005), vol. 1, IEEE Computer Society, pp. 886–893. (Cited on pages 4, 16, 32 and 58.)

[32] DIVVALA, S. K., HOIEM, D., HAYS, J. H., EFROS, A. A., AND HEBERT, M. An empirical study of context in object detection. In *2009 IEEE Conference on computer vision and Pattern Recognition* (2009), IEEE, pp. 1271–1278. (Cited on page 83.)

[33] DOLLÁR, P., WOJEK, C., SCHIELE, B., AND PERONA, P. Pedestrian detection: A benchmark. (Cited on pages 6, 7, 10, 32, 38 and 57.)

[34] DOLLAR, P. E. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence* (2012). (Cited on pages xvii, 34, 39, 58, 64, 67, 79, 95, 103, 108 and 112.)

[35] DONAHUE, J., JIA, Y., VINYALS, O., HOFFMAN, J., ZHANG, N., TZENG, E., AND DARRELL, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning* (2014), pp. 647–655. (Cited on page 62.)

[36] DU, X. E. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In *WACV* (2017), IEEE. (Cited on pages 71, 77, 78, 79 and 107.)

[37] ENZWEILER, M., AND GAVRILA, D. M. Monocular pedestrian detection: Survey and experiments. *IEEE transactions on pattern analysis and machine intelligence 31*, 12 (2009), 2179–2195. (Cited on page 39.)

[38] ESPOSITO, F., MALERBA, D., SEMERARO, G., AND KAY, J. A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence 19*, 5 (1997), 476–491. (Cited on page 57.)

[39] ESS, A., LEIBE, B., AND VAN GOOL, L. Depth and appearance for mobile scene analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (2007), IEEE, pp. 1–8. (Cited on page 32.)

[40] ET.AL, B. Learning object motion patterns for anomaly detection and improved object detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE. (Cited on page 62.)

[41] ET.AL, B. Improving object detection with one line of code. *arXiv preprint arXiv:1704.04503* (2017). (Cited on page 66.)

[42] ET.AL, E. Improving small object proposals for company logo detection. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (2017), ACM. (Cited on page 68.)

[43] ET.AL, K. H. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017). (Cited on page 80.)

[44] ET.AL, L. Feature pyramid networks for object detection. In *CVPR* (2017). (Cited on pages 27 and 28.)

[45] ET.AL, S. R. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS* (2015). (Cited on pages 38, 39, 50, 52, 54, 64, 65, 66, 68, 70 and 71.)

[46] ET.AL, S. Z. How far are we from solving pedestrian detection? In *CVPR* (2016). (Cited on pages xvi, 71 and 72.)

[47] ET.AL, Z. How far are we from solving pedestrian detection? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016). (Cited on pages 33, 42, 53, 64 and 66.)

[48] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K., WINN, J., AND ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International journal of computer vision 88*, 2 (2010), 303–338. (Cited on page 3.)

[49] FANG, L., ZHAO, X., AND ZHANG, S. Small-objectness sensitive detection based on shifted single shot detector. *Multimedia Tools and Applications* (2018), 1–19. (Cited on page 77.)

[50] FELZENSZWALB, P. F., MCALLESTER, D. A., RAMANAN, D., ET AL. A discriminatively trained, multiscale, deformable part model. In *Cvpr* (2008), vol. 2, p. 7. (Cited on pages 4, 10, 18 and 27.)

[51] FREUND, Y., SCHAPIRE, R. E., ET AL. Experiments with a new boosting algorithm. In *icml* (1996), vol. 96, Citeseer, pp. 148–156. (Cited on page 31.)

[52] GEIGER, A., LENZ, P., STILLER, C., AND URTASUN, R. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research 32*, 11 (2013), 1231–1237. (Cited on pages 33 and 57.)

[53] GEIGER, A., LENZ, P., AND URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 3354–3361. (Cited on page 32.)

[54] GERONIMO, D., LOPEZ, A. M., SAPPA, A. D., AND GRAF, T. Survey of pedestrian detection for advanced driver assistance systems. *IEEE transactions on pattern analysis and machine intelligence 32*, 7 (2010), 1239–1258. (Cited on page 39.)

[55] GIRSHICK, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448. (Cited on pages 17, 25 and 53.)

[56] GIRSHICK, R., DONAHUE, J., DARRELL, T., AND MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587. (Cited on page 102.)

[57] GLOROT, X., AND BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256. (Cited on pages 38 and 42.)

[58] GONG, Y., WANG, L., GUO, R., AND LAZEBNIK, S. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision* (2014), Springer, pp. 392–407. (Cited on page 27.)

[59] GROSHEN, E. L., HELPER, S., MACDUFFIE, J. P., AND CARSON, C. Preparing us workers and employers for an autonomous vehicle future. (Cited on page 2.)

[60] HARA, K., KATAOKA, H., AND SATOH, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA* (2018), pp. 18–22. (Cited on page 43.)

[61] HARDT, M., RECHT, B., AND SINGER, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240* (2015). (Cited on page 40.)

[62] HARMAN, R. K., AND PATCHELL, J. W. Perimeter surveillance system, Feb. 3 1981. US Patent 4,249,207. (Cited on page 2.)

[63] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2961–2969. (Cited on pages xi, 21, 22 and 23.)

[64] HE, K., AND SUN, J. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 5353–5360. (Cited on page 104.)

[65] HE, K., ZHANG, X., REN, S., AND SUN, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence 37*, 9 (2015), 1904–1916. (Cited on page 27.)

[66] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778. (Cited on pages 43, 44 and 46.)

[67] HE, K., ZHANG, X., REN, S., AND SUN, J. Identity mappings in deep residual networks. In *European conference on computer vision* (2016), Springer, pp. 630–645. (Cited on pages xii, 26, 45, 87 and 88.)

[68] HINTON, G. E. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619. (Cited on page 24.)

[69] HOLSCHNEIDER, M., TCHAMITCHIAN, P., KRONLAND-MARTINET, R., AND MORLET, J. The'algorithme a trous'. Tech. rep., 1988. (Cited on pages 47 and 118.)

[70] HOSANG, J. H., BENENSON, R., AND SCHIELE, B. Learning non-maximum suppression. In *CVPR* (2017), pp. 6469–6477. (Cited on page 58.)

[71] HU, J., SHEN, L., AND SUN, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7132–7141. (Cited on page 26.)

[72] HU, Q., WANG, P., SHEN, C., VAN DEN HENGEL, A., AND PORIKLI, F. Pushing the limits of deep cnns for pedestrian detection. *IEEE Transactions on Circuits and Systems for Video Technology 28*, 6 (2017), 1358–1368. (Cited on page 31.)

[73] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *CVPR* (2017), vol. 1, p. 3. (Cited on pages 26 and 56.)

[74] HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A., FATHI, A., FISCHER, I., WOJNA, Z., SONG, Y., GUADARRAMA, S., ET AL. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7310–7311. (Cited on pages 22, 72, 76 and 111.)

[75] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015). (Cited on pages 46 and 47.)

[76] IZADYYAZDANABADI, M., BELYKH, E., MOONEY, M., MARTIROSYAN, N., ESCHBACHER, J., NAKAJI, P., PREUL, M. C., AND YANG, Y. Convolutional neural networks: ensemble modeling, fine-tuning and unsupervised semantic localization for neurosurgical cle images. *Journal of Visual Communication and Image Representation 54* (2018), 10–20. (Cited on page 5.)

[77] JACOB, B., KLIGYS, S., CHEN, B., ZHU, M., TANG, M., HOWARD, A., ADAM, H., AND KALENICHENKO, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2704–2713. (Cited on page 76.)

[78] JACOBSTEIN, N. Autonomous vehicles: An imperfect path to saving millions of lives, 2019. (Cited on page 2.)

[79] JOHNSON, R., AND ZHANG, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems* (2013), pp. 315–323. (Cited on page 40.)

[80] KESKAR, N. S., AND SOCHER, R. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628* (2017). (Cited on page 40.)

[81] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). (Cited on page 40.)

[82] KJÆRGAARD, M. B., WIRZ, M., ROGGEN, D., AND TRÖSTER, G. Mobile sensing of pedestrian flocks in indoor environments using wifi signals. In *2012 IEEE International Conference on Pervasive Computing and Communications* (2012), IEEE, pp. 95–102. (Cited on page 77.)

[83] KONG, T., YAO, A., CHEN, Y., AND SUN, F. Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 845–853. (Cited on pages 27 and 102.)

[84] KONTSCHIEDER, P., FITERAU, M., CRIMINISI, A., AND ROTA BULO, S. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1467–1475. (Cited on pages 30 and 57.)

[85] KORNBLITH, S., SHLENS, J., AND LE, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 2661–2671. (Cited on page 5.)

[86] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. (Cited on pages 7, 26, 28 and 62.)

[87] KRUITHOF, M. C., BOUMA, H., FISCHER, N. M., AND SCHUTTE, K. Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers. In *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII* (2016), vol. 9995, International Society for Optics and Photonics, p. 99950K. (Cited on page 5.)

[88] LAHOULI, I., KARAKASIS, E., HAELTERMAN, R., CHTOUROU, Z., DE CUBBER, G., GASTERATOS, A., AND ATTIA, R. Hot spot method for pedestrian detection using saliency maps, discrete chebyshev moments and support vector machine. *IET Image Processing 12*, 7 (2018), 1284–1291. (Cited on page 16.)

[89] LAMPERT, T. A., STUMPF, A., AND GANÇARSKI, P. An empirical study into annotator agreement, ground truth estimation, and algorithm evaluation. *IEEE Transactions on Image Processing 25*, 6 (2016), 2557–2572. (Cited on page 121.)

[90] LAN, W., DANG, J., WANG, Y., AND WANG, S. Pedestrian detection based on yolo network model. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)* (2018), IEEE, pp. 1547–1551. (Cited on pages 77 and 78.)

[91] LECUN, Y., AND CORTES, C. MNIST handwritten digit database. (Cited on page 24.)

[92] LEE, H., GROSSE, R., RANGANATH, R., AND NG, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (2009), ACM, pp. 609–616. (Cited on page 63.)

[93] LI, B., YAO, Q., AND WANG, K. A review on vision-based pedestrian detection in intelligent transportation systems. In *Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on* (2012), IEEE, pp. 393–398. (Cited on page 39.)

[94] LI, J., LIANG, X., SHEN, S., XU, T., FENG, J., AND YAN, S. Scale-aware fast r-cnn for pedestrian detection. *IEEE transactions on Multimedia 20*, 4 (2017), 985–996. (Cited on pages 27 and 28.)

[95] LI, J., LIANG, X., SHEN, S., XU, T., FENG, J., AND YAN, S. Scale-aware fast r-cnn for pedestrian detection. *IEEE transactions on Multimedia 20*, 4 (2018), 985–996. (Cited on pages 77 and 78.)

[96] LIN, D., TALATHI, S., AND ANNAPUREDDY, S. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning* (2016), pp. 2849–2858. (Cited on page 76.)

[97] LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K., AND DOLLÁR, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2980–2988. (Cited on page 27.)

[98] LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K., AND DOLLÁR, P. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence* (2018). (Cited on pages xvi, 53 and 54.)

[99] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision* (2014), Springer, pp. 740–755. (Cited on page 3.)

[100] LIU, J., ZHANG, S., WANG, S., AND METAXAS, D. N. Multispectral deep neural networks for pedestrian detection. *arXiv preprint arXiv:1611.02644* (2016). (Cited on page 16.)

[101] LIU, M., SHI, J., LI, Z., LI, C., ZHU, J., AND LIU, S. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics 23*, 1 (2016), 91–100. (Cited on page 26.)

[102] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., AND BERG, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision* (2016), Springer, pp. 21–37. (Cited on pages xvi, 6, 16, 18, 22, 24, 25, 28, 38, 42, 49, 50, 51, 53, 77, 78, 80, 82, 86, 91, 94, 96, 97, 112 and 117.)

[103] LIU, W., LIAO, S., HU, W., LIANG, X., AND CHEN, X. Learning efficient single-stage pedestrian detectors by asymptotic localization fitting. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 618–634. (Cited on pages 26, 27, 28, 29, 30, 77, 78, 112 and 113.)

[104] LIU, W., WANG, Z., LIU, X., ZENG, N., LIU, Y., AND ALSAADI, F. E. A survey of deep neural network architectures and their applications. *Neurocomputing 234* (2017), 11–26. (Cited on page 26.)

[105] LUO, P., TIAN, Y., WANG, X., AND TANG, X. Switchable deep network for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 899–906. (Cited on pages 4, 24 and 25.)

[106] LUO, P., WANG, X., AND TANG, X. Pedestrian parsing via deep decompositional network. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 2648–2655. (Cited on page 24.)

[107] MANEN, S., GUILLAUMIN, M., AND VAN GOOL, L. Prime object proposals with randomized prim's algorithm. In *Proceedings of the IEEE international conference on computer vision* (2013), pp. 2536–2543. (Cited on page 16.)

[108] MAO, J., XIAO, T., JIANG, Y., AND CAO, Z. What can help pedestrian detection? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 3127–3136. (Cited on pages 26, 27, 28, 29, 77 and 78.)

[109] MOLCHANOV, V., VISHNYAKOV, B., VIZILTER, Y., VISHNYAKOVA, O., AND KNYAZ, V. Pedestrian detection in video surveillance using fully convolutional yolo neural network. In *Automated Visual Inspection and Machine Vision II* (2017), vol. 10334, International Society for Optics and Photonics, p. 103340Q. (Cited on pages 77 and 78.)

[110] MOON, T. K., AND STIRLING, W. C. *Mathematical methods and algorithms for signal processing*, vol. 1. Prentice hall Upper Saddle River, NJ, 2000. (Cited on page 88.)

[111] MORI, H., CHARKARI, N. M., AND MATSUSHITA, T. On-line vehicle and pedestrian detections based on sign pattern. *IEEE Transactions on industrial electronics 41*, 4 (1994), 384–391. (Cited on page 1.)

[112] NEUMANN, L., KARG, M., ZHANG, S., SCHARFENBERGER, C., PIEGERT, E., MISTR, S., PROKOFYEVA, O., THIEL, R., VEDALDI, A., ZISSERMAN, A., ET AL. Nightowls: A pedestrians at night dataset. (Cited on page 120.)

[113] NEUMANN, L., ZISSERMAN, A., AND VEDALDI, A. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. (Cited on page 77.)

[114] NOH, J., LEE, S., KIM, B., AND KIM, G. Improving occlusion and hard negative handling for single-stage pedestrian detectors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). (Cited on page 25.)

[115] OH, S., HOOGS, A., PERERA, A., CUNTOOR, N., CHEN, C.-C., LEE, J. T., MUKHERJEE, S., AGGARWAL, J., LEE, H., DAVIS, L., ET AL. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011* (2011), IEEE, pp. 3153–3160. (Cited on page 120.)

[116] OLIVA, A. Gist of the scene. In *Neurobiology of attention*. Elsevier, 2005, pp. 251–256. (Cited on page 83.)

[117] OREN, M., PAPAGEORGIOU, C., SINHA, P., OSUNA, E., AND POGGIO, T. Pedestrian detection using wavelet templates. In *cvpr* (1997), vol. 97, pp. 193–199. (Cited on page 16.)

[118] OUYANG, W., AND WANG, X. Joint deep learning for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 2056–2063. (Cited on page 25.)

[119] OUYANG, W., ZHOU, H., LI, H., LI, Q., YAN, J., AND WANG, X. Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence 40*, 8 (2018), 1874–1887. (Cited on page 78.)

[120] PENG, Q., LUO, W., HONG, G., FENG, M., XIA, Y., YU, L., HAO, X., WANG, X., AND LI, M. Pedestrian detection for transformer substation based on gaussian mixture model and yolo. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)* (2016), vol. 2, IEEE, pp. 562–565. (Cited on pages 77 and 78.)

[121] PETERSON, R. D. Vehicle mounted surveillance system, Aug. 14 1990. US Patent 4,949,186. (Cited on page 2.)

[122] RAGHU, M., POOLE, B., KLEINBERG, J., GANGULI, S., AND SOHL-DICKSTEIN, J. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336* (2016). (Cited on page 56.)

[123] RAHTU, E., KANNALA, J., AND BLASCHKO, M. Learning a category independent object detection cascade. In *2011 international conference on Computer Vision* (2011), IEEE, pp. 1052–1059. (Cited on page 16.)

[124] RAN, Y., WEISS, I., ZHENG, Q., AND DAVIS, L. S. Pedestrian detection via periodic motion analysis. *International Journal of Computer Vision 71*, 2 (2007), 143–160. (Cited on page 16.)

[125] READING, I., WAN, C., AND DICKINSON, K. Developments in pedestrian detection. *Traffic engineering and control 36*, 10 (1995), 538–542. (Cited on page 1.)

[126] REDDI, S. J., KALE, S., AND KUMAR, S. On the convergence of adam and beyond. (Cited on page 40.)

[127] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788. (Cited on pages 16, 22, 24, 77, 78, 80, 86, 91 and 112.)

[128] REDMON, J., AND FARHADI, A. Yolo9000: better, faster, stronger. *arXiv preprint* (2017). (Cited on page 25.)

[129] REN, J., CHEN, X., LIU, J., SUN, W., PANG, J., YAN, Q., TAI, Y.-W., AND XU, L. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5420–5428. (Cited on pages 77 and 78.)

[130] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99. (Cited on pages xi, xii, xvi, 6, 11, 16, 17, 18, 21, 22, 23, 28, 29, 33, 38, 42, 49, 51, 76, 77, 80, 82, 83, 84, 91, 93, 99, 102, 108, 111, 112, 114, 117, 118 and 119.)

[131] ROBICQUET, A., SADEGHIAN, A., ALAHI, A., AND SAVARESE, S. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision* (2016), Springer, pp. 549–565. (Cited on page 120.)

[132] SCHMIDHUBER, J., AND HOCHREITER, S. Long short-term memory. *Neural Comput 9*, 8 (1997), 1735–1780. (Cited on page 4.)

[133] SCIEUR, D., D'ASPREMONT, A., AND BACH, F. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems* (2016), pp. 712–720. (Cited on page 40.)

[134] SCIEUR, D., OYALLON, E., D'ASPREMONT, A., AND BACH, F. Nonlinear acceleration of deep neural networks. *arXiv preprint arXiv:1805.09639* (2018). (Cited on page 40.)

[135] SERMANET, P. E. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR* (2013), IEEE. (Cited on page 25.)

[136] SHARIF RAZAVIAN, A., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813. (Cited on page 5.)

[137] SHRIVASTAVA, A., GUPTA, A., AND GIRSHICK, R. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 761–769. (Cited on pages 10 and 11.)

[138] SILVER, D., HUANG, A., MADDISON, C. J., GUEZ, A., SIFRE, L., VAN DEN DRIESSCHE, G., SCHRITTWIESER, J., ANTONOGLOU, I., PANNEERSHELVAM, V., LANCTOT, M., ET AL. Mastering the game of go with deep neural networks and tree search. *nature 529*, 7587 (2016), 484. (Cited on page 57.)

[139] SIMO-SERRA, E., TRULLS, E., FERRAZ, L., KOKKINOS, I., AND MORENO-NOGUER, F. Fracking deep convolutional image descriptors. *arXiv preprint arXiv:1412.6537* (2014). (Cited on pages 10 and 44.)

[140] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). (Cited on pages xii, xvi, 26, 38, 44, 46, 62, 64, 65 and 82.)

[141] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). (Cited on pages 43 and 55.)

[142] SINGH, B., AND DAVIS, L. S. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 3578–3587. (Cited on pages 6 and 7.)

[143] SUNG, K.-K. Learning and example selection for object and pattern detection. (Cited on page 10.)

[144] SZE, V., CHEN, Y.-H., YANG, T.-J., AND EMER, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE 105*, 12 (2017), 2295–2329. (Cited on pages 26 and 88.)

[145] SZEGEDY, C., IOFFE, S., VANHOUCKE, V., AND ALEMI, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017). (Cited on page 47.)

[146] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going deeper with convolutions. In

Proceedings of the IEEE conference on computer vision and pattern recognition (2015), pp. 1–9. (Cited on page 43.)

[147] SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J., AND WOJNA, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2818–2826. (Cited on pages 26, 43, 44, 46 and 55.)

[148] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013). (Cited on page 26.)

[149] TAJBAKHSH, N., SHIN, J. Y., GURUDU, S. R., HURST, R. T., KENDALL, C. B., GOTWAY, M. B., AND LIANG, J. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging 35*, 5 (2016), 1299–1312. (Cited on page 5.)

[150] TANG, S., ANDRILUKA, M., AND SCHIELE, B. Detection and tracking of occluded people. *International Journal of Computer Vision 110*, 1 (2014), 58–69. (Cited on page 10.)

[151] THORPE, C., HERBERT, M., KANADE, T., AND SHAFER, S. Toward autonomous driving: the cmu navlab. i. perception. *IEEE expert 6*, 4 (1991), 31–42. (Cited on page 2.)

[152] TIAN, Y., LUO, P., WANG, X., AND TANG, X. Pedestrian detection aided by deep learning semantic tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 5079–5087. (Cited on page 16.)

[153] TIAN, Y. E. Deep learning strong parts for pedestrian detection. In *CVPR* (2015). (Cited on pages 24 and 25.)

[154] TOMMASI, T., PATRICIA, N., CAPUTO, B., AND TUYTELAARS, T. A deeper look at dataset bias. In *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 37–55. (Cited on page 8.)

[155] TRAN, L., YIN, X., AND LIU, X. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1415–1424. (Cited on page 7.)

[156] TSUKIYAMA, T., AND SHIRAI, Y. Detection of the movements of persons from a sparse sequence of tv images. *Pattern Recognition 18*, 3-4 (1985), 207–213. (Cited on page 1.)

[157] UIJLINGS, J. R., VAN DE SANDE, K. E., GEVERS, T., AND SMEULDERS, A. W. Selective search for object recognition. *International journal of computer vision 104*, 2 (2013), 154–171. (Cited on pages 16 and 17.)

[158] UJJWAL, DZIRI, A., LEROY, B., AND BREMOND, F. Late fusion of multiple convolutional layers for pedestrian detection. *To appear in, 2018 International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (2018). (Cited on pages 50 and 56.)

[159] UJJWAL, DZIRI AZIZ, L. B., AND FRANCOIS, B. Late fusion of multiple convolutional layers for pedestrian detection. (Cited on pages 4, 17, 26, 27, 28, 30, 77 and 78.)

[160] VEIT, A., WILBER, M. J., AND BELONGIE, S. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems* (2016), pp. 550–558. (Cited on page 45.)

[161] VIOLA, P., AND JONES, M. J. Robust real-time face detection. *International journal of computer vision 57*, 2 (2004), 137–154. (Cited on page 58.)

[162] WALKER, J., GUPTA, A., AND HEBERT, M. Patch to the future: Unsupervised visual prediction. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2014), pp. 3302–3309. (Cited on page 120.)

[163] WALLACE, B. C., SMALL, K., BRODLEY, C. E., AND TRIKALINOS, T. A. Class imbalance, redux. In *2011 IEEE 11th international conference on data mining* (2011), IEEE, pp. 754–763. (Cited on page 10.)

[164] WANG, X., BAI, X., LIU, W., AND LATECKI, L. J. Feature context for image classification and object detection. In *CVPR 2011* (2011), IEEE, pp. 961–968. (Cited on page 83.)

[165] WANG, X., AND OUYANG, W. A discriminative deep model for pedestrian detection with occlusion handling. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 3258–3265. (Cited on page 25.)

[166] WANG, X., XIAO, T., JIANG, Y., SHAO, S., SUN, J., AND SHEN, C. Repulsion loss: Detecting pedestrians in a crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 7774–7783. (Cited on pages 26, 28, 29, 30, 53, 78, 94, 95, 96, 112 and 113.)

[167] WILSON, A. C., ROELOFS, R., STERN, M., SREBRO, N., AND RECHT, B. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems* (2017), pp. 4148–4158. (Cited on page 40.)

[168] WOJEK, C., WALK, S., AND SCHIELE, B. Multi-cue onboard pedestrian detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), IEEE, pp. 794–801. (Cited on pages 16 and 32.)

[169] XU, Y., XIAO, T., ZHANG, J., YANG, K., AND ZHANG, Z. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369* (2014). (Cited on page 27.)

[170] XUE, J.-R., FANG, J.-W., AND ZHANG, P. A survey of scene understanding by event reasoning in autonomous driving. *International Journal of Automation and Computing 15*, 3 (2018), 249–266. (Cited on page 119.)

[171] YANG, F., CHEN, H., LI, J., LI, F., WANG, L., AND YAN, X. Single shot multibox detector with kalman filter for online pedestrian detection in video. *IEEE Access* (2019). (Cited on page 77.)

[172] YANG, T., ZHANG, X., LI, Z., ZHANG, W., AND SUN, J. Metaanchor: Learning to detect objects with customized anchors. In *Advances in Neural Information Processing Systems* (2018), pp. 320–330. (Cited on page 41.)

[173] YASUTOMI, S., AND MORI, H. A method for discriminating of pedestrian based on rhythm. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)* (1994), vol. 2, IEEE, pp. 988–995. (Cited on page 1.)

[174] YU, F., XIAN, W., CHEN, Y., LIU, F., LIAO, M., MADHAVAN, V., AND DARRELL, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687* (2018). (Cited on pages 6, 32, 33, 38, 59 and 119.)

[175] ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012). (Cited on page 40.)

[176] ZEILER, M. D., AND FERGUS, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833. (Cited on pages 7 and 62.)

[177] ZHANG, L., LIN, L., LIANG, X., AND HE, K. Is faster r-cnn doing well for pedestrian detection? In *European conference on computer vision* (2016), Springer, pp. 443–457. (Cited on pages xii, 3, 16, 17, 25, 26, 30, 33, 38, 41, 47, 48, 49, 50, 53, 56, 62, 64, 65, 68, 70, 71, 77, 78, 81, 83, 94, 96, 112, 117 and 118.)

[178] ZHANG, S., BENENSON, R., OMRAN, M., HOSANG, J., AND SCHIELE, B. Towards reaching human performance in pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence 40*, 4 (2018), 973–986. (Cited on pages 39, 57, 58, 59 and 95.)

[179] ZHANG, S., BENENSON, R., AND SCHIELE, B. Citypersons: A diverse dataset for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 3213–3221. (Cited on pages xvii, 5, 6, 7, 10, 30, 32, 33, 38, 51, 95, 103, 108, 112 and 114.)

[180] ZHANG, S., BENENSON, R., SCHIELE, B., ET AL. Filtered channel features for pedestrian detection. In *CVPR* (2015), vol. 1, p. 4. (Cited on pages 16 and 64.)

[181] ZHANG, S., WEN, L., BIAN, X., LEI, Z., AND LI, S. Z. Occlusion-aware r-cnn: detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 637–653. (Cited on page 77.)

[182] ZHANG, S., YANG, J., AND SCHIELE, B. Occluded pedestrian detection through guided attention in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 6995–7003. (Cited on page 103.)

[183] ZHANG, X., CHENG, L., LI, B., AND HU, H.-M. Too far to see? not really!âpedestrian detection with scale-aware localization policy. *IEEE transactions on image processing 27*, 8 (2018), 3703–3715. (Cited on pages 6 and 7.)

[184] ZHOU, Z.-H., AND FENG, J. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835* (2017). (Cited on page 57.)

[185] ZITNICK, C. L., AND DOLLÁR, P. Edge boxes: Locating object proposals from edges. In *European conference on computer vision* (2014), Springer, pp. 391–405. (Cited on pages 16 and 17.)

[186] ZUO, Y., AND DRUMMOND, T. Fast residual forests: Rapid ensemble learning for semantic segmentation. In *Conference on Robot Learning* (2017), pp. 27–36. (Cited on page 57.)