TECHNISCHE UNIVERSITÄT
CHEMNITZ

Pedestrian detection in front of the ego vehicle using (stereo) camera in the urban scene: Deep versus Shallow learning approaches

Master Thesis
for
the fulfillment of the academic degree
Master of Science in Information and Communication Systems

Faculty of Electrical Engineering and Information Technology
Professorship of Communications Engineering
Technische Universität Chemnitz
November 2016

Submitted by:
Gurucharan Srinivas
Matr Nr.:357634

Supervisors:

Univ.-Prof.Dr.Gerd Wanielik
(Technische Universität Chemnitz)

M.Sc.Paulin Pekezou Fouopi
(Deutsches Zentrum für Luft- und Raumfahrt, Institut für Verkehrssytemtechnik)

Acknowledgement

This master thesis work has been carried out at the Institute of Transportation Systems, Deutsches Zentrum für Luft- und Raumfahrt e.V (DLR) and at the department of Electrical Engineering and Information Technology at TU Chemnitz, Germany.

In the first place, I thank M.Sc.Paulin Pekezou Fouopi, Wissenschaftlicher Mitarbeiter at DLR e.V for providing me this wonderful opportunity and for his constant guidance and support during my thesis work. I would like to acknowledge and extend my heartfelt gratitude to Prof.Dr.-Ing.Gerd Wanielik and Dr.-Ing.Ulrich Neubert for their valuable support to accomplish my thesis at Professorship of Communication Engineering, TU Chemnitz.

I express my gratitude to many colleagues and friends who have supported me with new ideas and suggestions to make my work better. I am grateful to my family for their best wishes which helped me to overcome all the hurdles during the entire thesis work.

Thank you

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit mit dem Thema “ Pedestrian detection in front of the ego vehicle using (stereo) camera in the urban scene: Deep versus Shallow learning approaches “ selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Weiter erkläre ich wörtliche und sinngemäße Zitate als solche gekennzeichnet zu haben.

_____

Ort, Datum

_____

Vorname, Name

Abstract

Object detection is crucial in the environment of autonomous driving and advance driver assistance systems for safely maneuvring vehicle in the urban traffic. Among the traffic participants we find pedestrians are the one who are most vulnerable and their safety is also crucial. Therefore, this work focuses on pedestrian detection in urban environment using the camera mounted on ego vehicle. The thesis aims at understanding and comparison of shallow and deep learning approaches for pedestrian detection, and two ensemble methods are proposed that combines the chosen deep and shallow method with the context-based classifier respectively. Firstly, an pre-trained deep architecture for object detection is combined with the context-based classifier. Whereas, in second method shallow approach is combined with context-based classifier. Further in the outlook of this work stereo data is used to minimize the detected false positives form the proposed ensemble deep approach. Prototyping of first proposed method is achieved using the CAFFE deep learning framework with Python interface, and the second shallow method is achieved using the well known computer vision library OpenCV with C++. The proposed method is trained, tested and evaluated on Caltech pedestrian dataset with different metric.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent times extensive progress in development of advance driver assistance systems is noticed and a camera sensor can be found in every car that is produced in last few years. Computer vision provides cost effective solution in developing Autonomous Emergency Braking (AEB) for vulnerable road users. Pedestrians are the one who are most prone to traffic accidents in urban environment. Detecting pedestrian will not only improve the safety, but also increase the ability to achieve fully autonomous self-driving car. Pedestrian in the urban environment appear with huge variance in pose, clutter background and distortion due to illumination effect. Recent advancement in Convolutional Neural Networks (CNN) has shown the ability to learn features inherently that tackle the pedestrian appearance variance. Several Shallow Learning (SL) approaches have been proposed that involves complex hand-coded feature engineering task to encode the pedestrian into discriminant features that enables to detect pedestrian in the scene. CNNs fail to capture inherently the contextual information that are relative to object, unless they are modelled explicitly during training procedure. Recent focus of work in deep learning is towards developing architectures that make use of context-information implicitly or model explicitly at the post-processing stage of the detector.

The main focus of this work is to combine the contextual information of pedestrian with chosen CNN object detector and shallow approach for detecting pedestrian in realistic and challenging onboard dataset [1]. First, a naïve Bayesian context-based classifier is learnt using training dataset. Secondly, two classifier fusion techniques are proposed. In the first fusion technique context-based classifier is combined with pre-trained CNN object detector. Whereas, in the second fusion technique context-based classifier is combined with SVM-HOG [2] detector pipeline. Finally, an outlook work is shown that uses stereo information to eliminate the false positives produced by fusing context-based classifier with CNN object detector. Training and evaluation of proposed methods is done on real traffic data [1].

Outline of the Thesis

Several essential aspects that are relevant to thesis work are discussed in each chapter separately. This section provides outline of thesis structure.

- **Chapter 2:** Related work towards pedestrian detection using deep and shallow Learning are discussed. Also a brief overview regarding available pedestrian datasets and used frameworks are given.
- **Chapter 3:** Explains the prerequisite knowledge on feature engineering task and classification procedure involved in deep learning and shallow learning. It provides the foundation to understand the proposed thesis work.
- **Chapter 4:** Explains the proposed concepts briefly.
- **Chapter 5:** This chapter describes the evaluation method employed to weight the proposed work.
- **Chapter 6:** Implementation details of the proposed methods are explained in detail with relevant graphical representations.
- **Chapter 7:** Obtained results of the proposed method are summarized with reference to evaluation metrics.
- **Chapter 8:** Finally, this chapter concludes the thesis work and provides proposal of future work that is based on this thesis.

Chapter 2

Literature review

2.1 Deep learning approaches for object detection

Popularity of pedestrian detection using the deep neural networks methods has been focused in recent years, their is lot of research work that needs to be addressed in this area.

ConvNet [3] was the first paper that made use of convolutional neural network to address the pedestrian detection problem. Convolutional sparse coding was used to initialize each layer of the designed network and the perform fine-tuning of entire network to perform detection. Last and second last layers features where used for detection. ConvNet [3] where trained on limited dataset and the length of the network was very small. Later AlexNet [4] an large convolutional neural network (CNN, ConvNet) was proposed which showed outstanding performance on classification and localization task on ImageNet 2012 competition. The details of localization task by AlexNet [4] was not described. Many authors have used sliding window approach to generate region hypothesis for detection and localization of given object in the image. In ImageNet 2013 competition challenge, Overfeat [5] addressed the multi-scale sliding window approach to detect and localize the object instance that appear at different scales buy wining 1^{st} in that years of challenge. This work emphazised the capabilities of ConvNets further in solving the task of object detection and localization.

The R-CNN: Regions with CNN features [6], neural network method addressed the detection and localization pipeline by over throwing the design of Overfeat [5] for object detection. The R-CNN used large ImageNet data set to pre-train the convolutional layers and then it was fine-tuned on PASCAL VOC [7]. A selective search [8] region proposal method was used to propose 2000 region hypothesis. High-capacity CNNs was applied to these proposed regions. Later a class-specific linear SVMs would classify the computed CNN features of each proposed regions. This allowed R-CNN to achieve a mean Average Precision (mAP) of 53.3% on PASCAL VOC detection dataset. R-CNN included a multi-stage training pipeline, it included three stage of training. In the first stage a ConvNet on object proposals is fine-tuned

using log-loss, in second stage the fine-tuned softmax layer is replaced with SVMs which act as object detector. In the last stage a bounding-box regressors is learnt. This training procedure is expensive in memory and even in time. Also, for each region proposed a CNN features are extracted which make object detection slow.

A revised Fast R-CNN [9] was proposed which eliminated the drawback of R-CNN [6]. Fast-RCNN detector pipeline accepts input image and region proposals generated using selective search [8]. Sequence of convolutional (conv) layer and max pooling layers produces conv feature map by acting on the input image. A region of interest (RoI) pooling layer is appended in between the first final conv layer and fully connected (fc) layer. This RoI pooling layer accepts proposed object proposals and extracts the dedicated feature vector from the conv feature map. Further each feature vector of proposed regions are fed into a series of fc layers which brach out at last into two output layers. One produces a softmax probability of the region and another layer acts as bounding box regressor. This redesigned pipeline allows Fast-RCNN (with VGG16 [10]) to train the network 10x faster than R-CNN. During testing the image is processed much faster and producing mAP 0f 66% on PASCAL VOC detection dataset.

The bottle neck included in region proposal task would limit the Fast-RCNN to work in real time. Region Proposal Network (RPN) is proposed in Faster-RCNN [11] that works on the computed convolutional features of the image for proposing regions. The RPN produces is trained end-to-end to produce potential object proposals. This significantly solves the bottle neck included in Fast-RCNN and R-CNN detector pipeline. Fast-RCNN using the RPN works at 5fps on a GPU. In ImageNet 2015 challenge task, Faster-RCNN achieved the 1^{st} place. In the recent times many CNN methods [12, 13] are proposed that eliminate the requirement of external region hypothesis generation. The YOLO [12] has very less recall on pedestrian detection and fails to detect objects that appear in dense.

Chosen method: CNNs require high end GPUs to train and inference them on the real platform. Therefore, this work focuses on employing pre-trained Fast-RCNN [9] detector pipeline to design proposed deep method. The pre-trained deep model of Fast-RCNN is available under the open-source with MIT License [14]. This available model is first trained on ImageNet dataset and later fine-tuned on PASCAL VOC dataset. Even though Faster-RCNN [11] perform better compared to Fast-RCNN [9], the Faster-RCNN includes end-to-end Region Proposal Network (RPN) that should be fine-tuned for our specific task. Due to unavailability of required GPU hardware for fine-tuning RPN and to have high degree of control over proposed regions, this work selects Fast-RCNN.

Frameworks: Handful of deep learning libraries are available under the BSD license. Caffe [15], Theano [16], and Torch [17] are some of the prominent deep learning libraries. Caffe [15] is the widely used framework when dealing on CNNs.

Core implementation of this framework is in C++ and provides binding to Python and MATLAB. Out of the existing deep framework Caffe is the most stable especially for image processing task. Caffe is highly modular framework and allows us to work in both CPU mode and GPU mode. It contains many off-the-shelf pre-trained base models like AlexNet [4], VGGnet [10], GoogleNet [18]. These pre-trained models allows to design and model detectors that eliminate costly procedure of training the deep models. The chosen Fast-RCNN [9] is implemented using Caffe framework, hence this work make use of Caffe with Python biding for prototyping proposed ensemble deep approach.

2.2 Conventional methods for pedestrian detection

Within the last years a number of several shallow approaches where proposed to tackle the problem of detecting pedestrian detection in automotive environment using the camera sensor. Histogram of Oriented Gradients (HOG) has a input feature descriptor as proven its ability to encode the pedestrian effectively. Using HOG directly as a input along with linear Support Vector Machine (SVM) as shown significant performance in detecting pedestrian on INRIA dataset [2]. After this work many variants of HOG have been proposed. Deformable parts model [19] also make use of HOG as input. The integral channel features [20] linearly transforms input channel resulting in multiple registered image channels; features such as HOG, Haar wavelets are computed using the integral channels. Colour provides essential texture information regarding the pedestrian, hence LBP [21] a variant of HOG feature captures texture information along with gradient information. These captured LBP features are provide as input to linear SVM for detection. The filtered channel features [22] transforms the input image into set of feature maps using filters. These feature maps are sum pooled into feature vector and fed into decision forest, which are learnt via Adaboost. Multi-cue onboard pedestrian detection [23] uses features like HOG, Haar, Oriented Histogram of Flow; cue-components are L2-normalized and concatenated, then MLPBoost classifier is learnt on these features to provide strong classification.

Chosen method and Framework: Considering the ambiguity in feature representation and complexity involved in shallow methods for pedestrian detection. A classic detector pipeline as in [2], which make use of HOG descriptor (with normalizing and pooling) as direct input for linear SVM [2] to perform pedestrian detection is consider in prototyping the proposed ensemble shallow approach. OpenCV [24] with C++ interface is used in this context of work for prototyping proposed shallow method for pedestrian detection. OpenCV is the widely available open source under BSD license.

2.3 Dataset

In supervised learning data is very crucial. In urban scene pedestrian appear in wide variety of pose and also they are influenced by many environmental variations. It is necessary to look in to openly available pedestrian datasets that capture these variations in large scale with additional features. Caltech [1], TUD-Brussels [23], ETH [25],Daimler pedestrian detection dataset [26] and INRIA [2] are some of the openly available pedestrian dataset. Among these datasets only INRIA pedestrian dataset is acquired using point and shoot camera which suffer from selection bias, whereas other mentioned datasets are acquired by means of camera mounted on the moving mobile platform.

The size of the Caltech dataset is twice large as compared to the other dataset that are mentioned (Table 1 from [1]). It contains approximately 250,0000 frames of video sequence recorded with $640 * 480$ resolution at 30Hz while driving through urban traffic environment. Caltech dataset is completely annotated with 350,000 bounding boxes and 23000 unique pedestrians. Temporal correspondence between annotated bounding boxes and occlusion labels are provided. The ETH [25] pedestrian dataset are recorded on a chariot with $640 * 480$ resolution at ≈ 14 frames per second. The dataset consists of single frame annotation, depth map for each image, and the calibration files. Size of the annotated training and test data is combined of $\approx 15k$. TUD-Brussels data is acquired from onboard camera mounted on driving car with resolution of $640 * 480$. The total dataset corresponds to 1776 annotated training and 1498 test pedestrians dataset. The Daimler [26] is the only dataset with non-colour images among all, and is only $\approx$10% magnitude of that of Caltech dataset.

The scale of pedestrians in the INRIA dataset is having median height ranging between 100-250 pixels. While most of the datasets (Caltech, ETH, TUD-Brussels and Daimler-DB) collected with the camera mounted on the mobile platform in driving environment has median height from 50-100 pixels. It accentuates the necessity to consider the detection of low resolution pedestrians in the driving environment.

Caltech dataset is the only dataset that has high detailed annotations with features including like colour image, video data and temporal correspondence between annotated bounding boxes and occlusion labels. Table 1 from [1] provides in depth details of Caltech dataset. Only *Set 05* of training dataset, and *Set 09* of testing dataset from Caltech pedestrian dataset is chosen for training, testing and evaluating the proposed pedestrian detectors. All pedestrian ranges (far, near and medium) with full visible region are taken into consideration during designing and testing the proposed methods. This work does not focus on occlusion handling, hence the occluded pedestrian regions are not taken into account.

Chapter 3

Fundamentals

3.1 Feature engineering

Feature representation are critical in object classification problem, there is a need for structuring the raw information into a distinctive features (like HOG, SHIFT, BOW....etc). A high level detail description of a features are necessary for classification and detection problems, this high level detailing comes at the cost of dealing with more data and complex design procedures. Feature engineering reasons this procedure of essential feature selection and generation that is necessity for the particular detection or classification problem. In the further subsection of this section, an procedure of generic featuring engineering process involved in shallow learning and inherent feature engineering involved in deep learning approaches are discussed.

3.1.1 Hand-coded Features

In shallow learning approach the feature engineering is a process that involves computer vision algorithms which indeed involves hand-coded procedure for feature representation. Selection and generation of features of a particular object is crucial as it describes the distinctive structure of the object during classification stage. Detecting human in the image is challenging because of the wide range of dynamic appearance and poses that they adapt. There is a need for robust feature set which counters the effects of dynamic illumination effects and cluttered background in the detection environment. From the work done by [2] it has been noticed *Histogram of Oriented Gradients* (HOG) descriptors can effectively generate features that are efficient to classify persons.

3.1.1.1 Histogram of Oriented Gradients(HOG)

HOG as a feature descriptor have proven its efficiency in the pedestrian classification in computer vision problems, hence this work utilizes the HOG descriptor as designed in [2] for pedestrian detection . Features are computed on the dense and overlapping grid of equi-spaced cells. Fig.1 gives us the ground explanation about HOG features that are generated. HOG descriptor defines the distribution of local intensity gradients and edges that characterises the shape and appearance of an

object in an image. In practice features using HOG descriptor is accomplished by dividing the image into small spatial regions called *cells*. Cells accumulates a gradient direction or edge orientation over the pixels in 1-Dimension histogram. Then the representation of features is given by the combined histogram entries of all cells. For the generated features to have the ability to counter illumination, shadowing ..etc, contrast-normalizing the local response is essential. Therefore "energy" (which is the local histogram) is accumulated over larger spatial regions called *blocks* and obtained energy is utilized to normalize all the cells in that particular block.

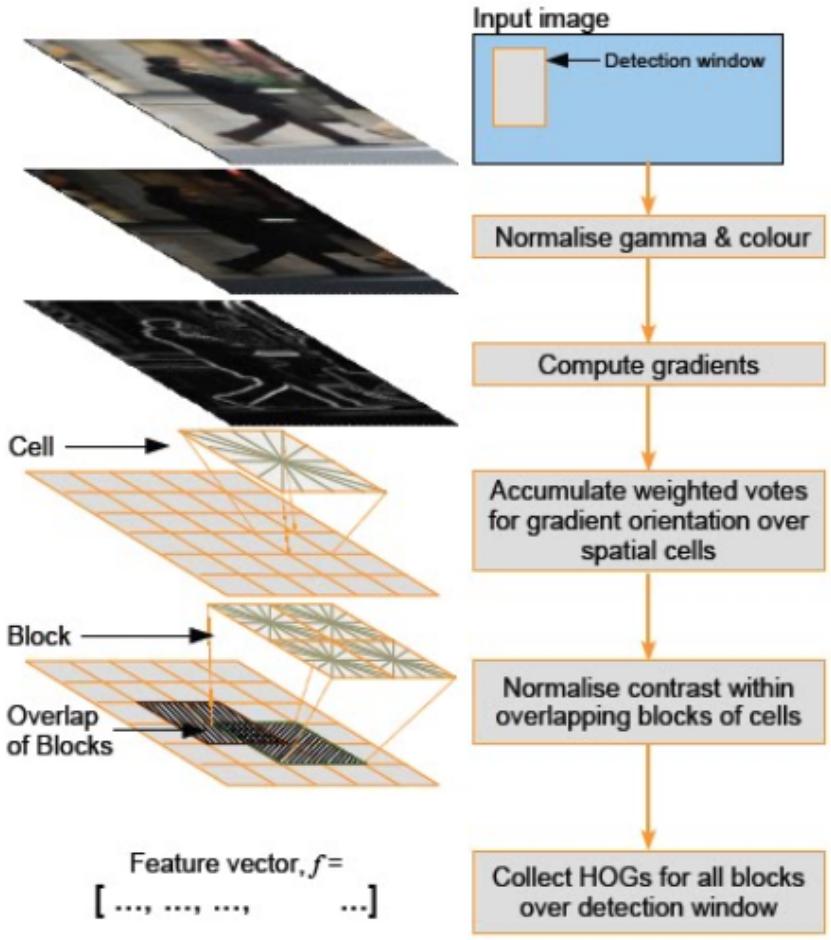


Figure 1: Graphical description of HOG descriptor feature extraction steps [2]

Normalizing gamma/colour: Usually working on the RGB and LAB colour space provides colour information which results in performance improvement of the overall detector, however as per the [2] showed using the grayscale had reduced in the performance. And performing gamma normalization with square root compression improves the detection performance by slight margin. This step in generation

of the features of an object had no significance because of the subsequent descriptor normalization.

Gradient computation, for grey images gradient is computed at each pixel. Whereas, for colour images gradient of a pixel is computed on all the three channels, and the gradient with highest norm is selected. Features that are computed with different masks and smoothing will results in significance performance of the detector. The work done by [2] suggest using the mask [-1,0,+1] and Gaussian smoothing σ=0 results in better performance of the detector. Therefore using smaller mask on the image results in finer details which are important and smoothing decrease the contrast of the edges in the image. Gradients provide information regarding the intensity of particular pixel changing in the given direction. This allows in finding the strong edges in that direction in which the gradient is computed. The gradient vector of a pixel at a particular location in the image can be computed by taking the partial derivative in x and y direction as given in below equation [27]:

$$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$$

Using the partial derivative function the gradient along x and y direction are computed as:

$$\frac{\partial I}{\partial x} = \frac{I(x+1,y) - I(x-1,y)}{2}$$

$$and$$

$$\frac{\partial I}{\partial y} = \frac{I(x,y+1) - I(x,y-1)}{2}$$

By using this gradient maps of an image, magnitude and orientation of a pixel (edge) is given as:

$$S = \sqrt{S_x^2 + S_y^2} \tag{3.1}$$

and

$$\theta = arctan(\frac{S_y}{S_x}) \tag{3.2}$$

Spatial/Orientation Binning: Each pixel based on its orientation of the gradient element contributes for its weighted vote for orientation. As noticed from fig.1, along the local spatial region called cells these votes are grouped into the orientation bins. The cells have the rectangular shape and the orientation binning are evenly spaced over 0°-180°, therefore these votes are function of the gradient magnitude. Hence only the magnitude is considered for better results, and also the number of orientation bins used are 9. According to [2] it shows increasing the number of orientation bins about 9 shows increase in performance and further increase has little

effect on the performance of the detector.

Block normalization and Descriptor Overlap: Grouping cells into large spatial regions called blocks and contrast normalising each block separately turns out to be effective to counter the local variations in illuminations and foreground-background contrast. Typically blocks are overlapped, hence each cell response contributes several components to the final feature vector (descriptor vector), therefore cells are normalized with respect to a different block. This leads to redundant information contribution of each cell, but as shown by [2] block normalization increases performance. Let **v** be a descriptor vector/feature vector that is unnormalized, $\|v\|_k$ be its k-norm and to avoid division by zero a small normalization constant ϵ is used. The *L2-norm* is given in equation (3.3) [28].

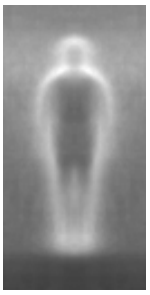
$$v \leftarrow \frac{v}{\sqrt{\|v\|_2^2+\epsilon^2}} \tag{3.3}$$

we use the *L2-Hys*; L2-norm followed by clipping and renormalizing from [29]. Using this for person class has been effective it can bee seen from the evaluation from [2].

Descriptor window size: The size of the descriptor is further discussed in the implementation chapter.

The above process depicts the operation and function involved in the HOG descriptor for feature generation and these generated features are used to train the classifier. In this work we use default linear SVM soft calssifer which is trained with SVMLight [30]. In the further sections of this chapter we explain the principle of *Linear Support Vector Machine* and their cases.

3.1.1.2 Visualization of HOG-descriptor computed features

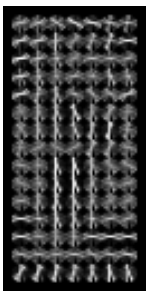


(a) Average gradient for positive images over INRIA dataset

(b) Positive SVM weights, where the block is centred on the pixel

(c) The negative SVM weights

(d) positive test image INRIA dataset



(e) Computed HOG descriptor on test image

(f) positive SVM weighted HOG descriptor

(g) HOG descriptor weighted by the negative SVM weights

Figure 2: Illustration of HOG descriptor and subsequent activations by the SVM weights [28]

The fig.2 gives an sample visualization of HOG descriptor computation on an INRIA dataset [2]. When an average gradient is taken on set of pedestrian, it is observed from the fig.2a it gives out the contour of pedestrian object. As noticed from the fig.2e, when an HOG is computed on the test image fig.10i the gradients of the edges are captured into orientation bins which in-turn encodes the contour of an specific object. Further as the computed HOG is weighted using the positive SVM weights it reveals the encoded object as noticed from the fig.2f.

3.1.2 Deep Learning

Deep learning is inspired by working principle of the biological neural network exiting in our human brains. The fig.3 shows the basic unit of the neuron. The input signal is brought by **dendrites** and the processed output signal is carried out by **axon**. The axons further branches out and connects to other dendrites via the synapses to other neuron unit. Fig.4 is the mathematical model of basic neuron unit. When looked into computations involved in the mathematical model, has the signal (e.g x_0) travels along the axon it will interact with strength (e.g w_0) of the synapses and then based on the influence of the synapses the product ($w_0 * x_0$) is carried by the dendrites to the cell body. Several other dendrites connected to cell body will carry this information to the cell, where they all get summed up. When the sum is greater than certain threshold the neuron will fire and send a spike along axon. The firing rate is modelled by the **activation function** f. The activation function that is employed in the VGGnet [10] (used base convolutional neural network model in the work) is the **ReLU** *Rectified Linear Unit*. The graphical representation of ReLU activation function is shown in fig.5, this activation function just thresholds at zero. The activation function [31] is given as, gather f(x)= max(0,x) gather

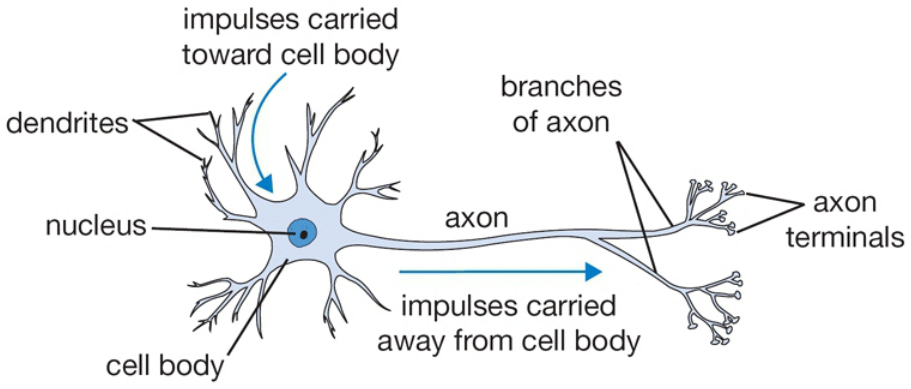


Figure 3: Basic biological neuron unit [31]

3.1.2.1 Convolutional Neural Networks(CNN, ConvNet)

Its important to understand the inherent feature engineering process that is involved in end-to-end CNNs. CNNs are the subclass of the deep learning, where to due it's

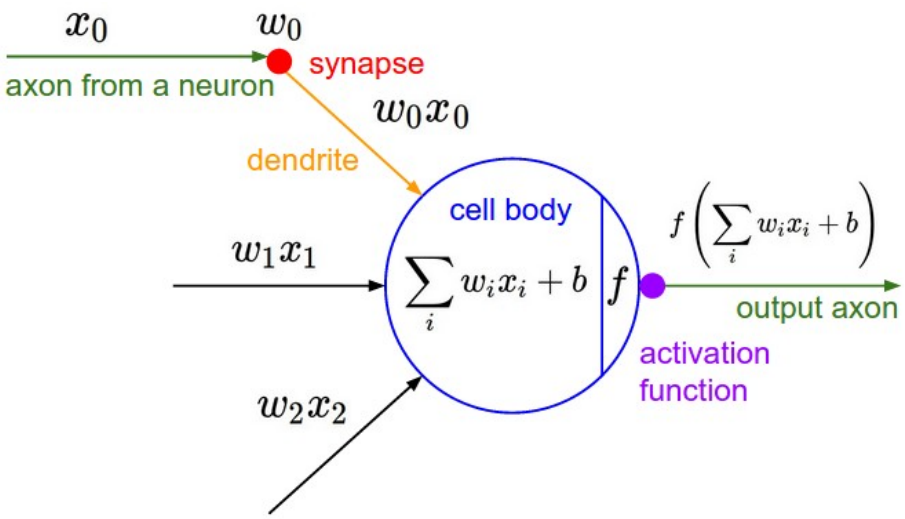


Figure 4: Mathematical model of neuron unit [31]

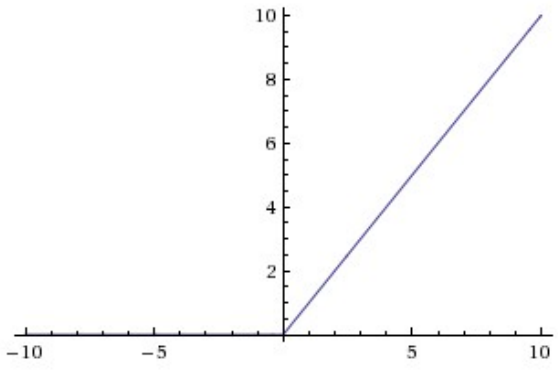


Figure 5: Rectified Linear Unit(ReLU) activation function [31]

significant mathematical operation *convolution* it derives the name *Convolutional Neural Networks*. The *Regular Neural Networks* take input as a single vector and transform it through the *hidden layers*. Each hidden layer is made up of a set of neurons and each neuron is connected to all the neurons in the previous layer, but is not connected to the neurons in the same layer which can be visualized in fig.6. The number of neurons in each layer corresponds to the number of weights in each layer. The final layer in the architecture is also fully connected and its responsible for the class score, this layers is also termed as *output layer*. *Regular Neural Network* does not scale well when dealing input as *full images*. Consider an image of size $600 * 500 * 3$ this would lead to a neurons of size $600 * 500 * 3 = 900000$ weights at each layer, eventually through the network architecture it adds up to large number of weights. Cleary it is visual that large number of parameters can results in *overfitting*.

3D volumes of neurons: Convolution Neural Network (see fig.7) modify the architecture of the regular neural network by having neurons arranged in 3 dimensions: *width, height, depth* (depth corresponds to third dimension of activation volume in each layer, not to the depth of the neural network architecture which can refer to total number of activation layers.) In CNNs each neuron is not connected in fully connected manner as seen in regular neural network, it is connected to just small region of the layer before. This modification in the architecture will significantly reduce the number of parameters needed to be learnt and reduce the risk of *overfitting*.

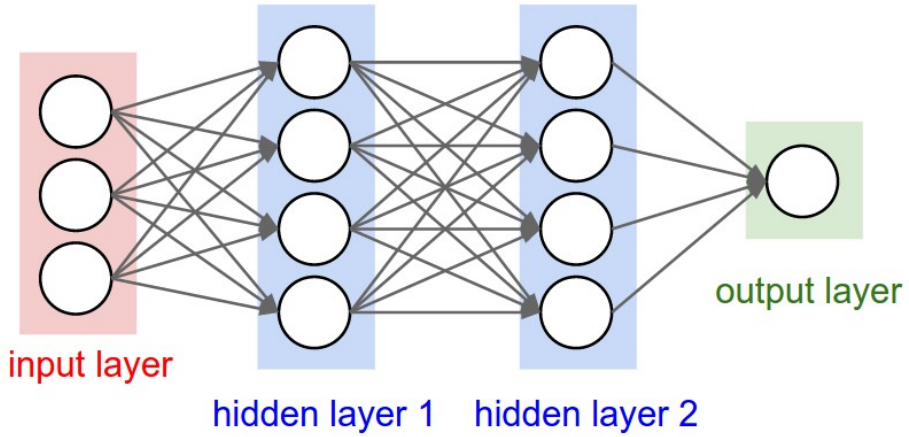


Figure 6: A threee layer *Regular Neural Network* [31]

Lets have a look into 16 layer VGG network architecture from [10], to understand the feature engineering process and classification process involved (classification process involved will be explained in next section). This work employes Fast-Region based Convolution Neural Network (Fast-RCNN) from [9] which make use of VGGnet architecture as the base network for object detection. The fig.8 gives the visualisation of general VGG-16 ConvNet model, based on required operation the final *fully connected* layer can have different dimension. For the purpose of explana-

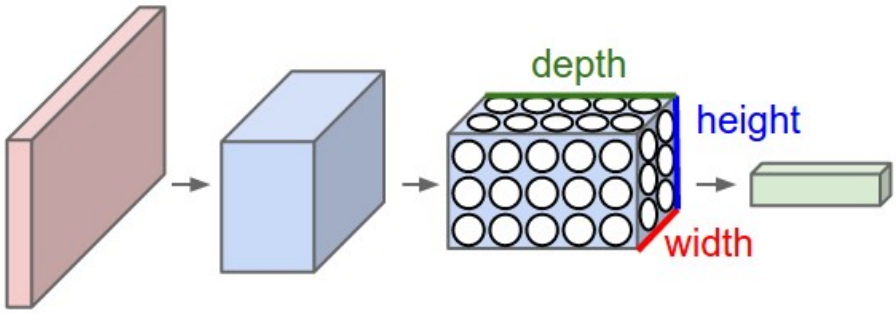


Figure 7: ConvNet arrangement of neurons in 3D [31]

tion general VGG-16 model is considered, which is trained and tested on Imagenet data set[32]. Therefore the *output layer* is reduce to 1*1*1000 = 1000. 1000 because the Imagnet has 1000 classes for classification challenge.

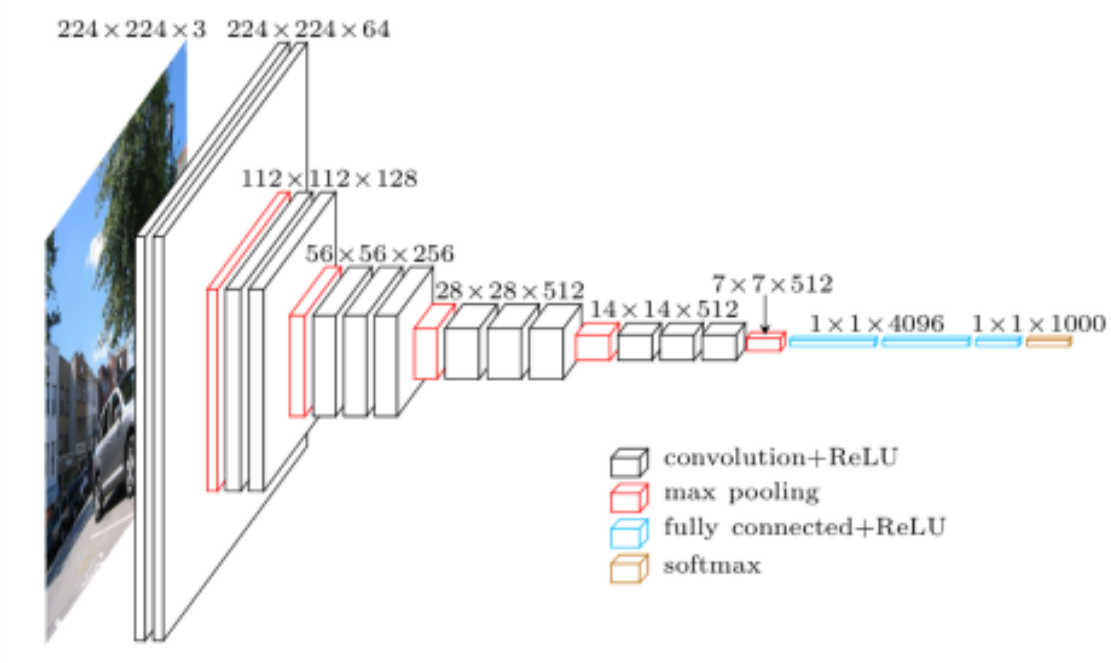


Figure 8: Layers involved in VGG16 ConvNet
Notes: In this thesis work a slightly modified VGG model is used which is available in [9], The VGG16 is consider to showcase the general used layers in ConvNet model.

General layers used to build ConvNet

As observed from fig.8 the ConvNet is made up of sequence of layers, where each layer will be responsible for transforming the volume of activations into another through a differential function. Main layers that are used in the building the ConvNet

are **Convolution layer, Pooling layer and Fully-Connected layer**. Before explaining the feature extraction process that is involved inherently in CNNs. It is important to understand the operation of different layers that are used repeatedly in building the ConvNet architecture.

Convolution Layer

Convolution layer parameter consists of a learnable filter or kernels, these kernels are small spatially (width and height) and perform operation over the depth of the input volume. In convolutional network terminology the first argument is referred to as *input* and the second argument is know as *kernel or filter* and the output is termed as *feature map*. Convolution operation over the image is 2-Dimension and can be given as (3.4) [33, 31].

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.4}$$

In VGG-16 CNN model uses the filter (K) size of 3*3, they use small spatial filed. During the forward pass, sliding of each filter across the width and height of the input volume is performed, and also dot products between the entries of the filter and the input at every position is computed. This gives rise to responses of that filter at every spatial position. Intuitively, filters get activated when they see some type of visual feature such as an edge of some orientation or a blotch of some colour on the first layer, or eventually entire patterns on higher layers of the network. These filter weights are learned over the training procedure to look for specific activations. This allows to have an entire set of filters in each CONV layer (e.g.64 filters in VGG Conv1 layer), and each of them will produce a separate 2-dimensional activation map. These activation maps will be stacked along the depth dimension to produce the output volume.

The neurons in the regular neural network will be connected fully with all other neurons of the before layer, but as the dimensionality of the input increases (incase of images) it is impossible to have full connection of the neurons with the previous layer neurons, therefore CNNs pocess only *local connectivity*. That is, the neurons look at particular region of the before input volume called ***Receptive field***. The fig.9 gives the visual understanding about receptive field. The output volume spatial arrangements are define by the hyperparameters: *depth, stride, zero-padding*.

Depth is usually an hyper-parameter that is defined by the number of *filters or kernels* that are used. Each filter looks at different characteristics of input. ***Stride*** specifies amount of sliding kernel across input, if larger the stride then it results in smaller the spatial size (width and height) of the output volume. To have the control over the spatial size (width and height excluding the depth dimension) we use the hyperparameter ***Zero-padding***. Zero's are padded around the input volume border, by doing this it allows to have the same spatial size of the input volume so

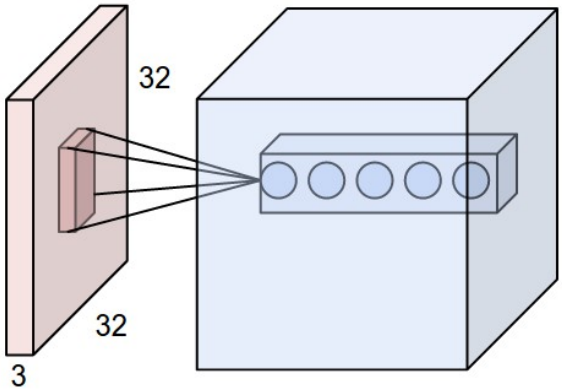


Figure 9: Neuron looking at the small spatial block of the before layer, this region is termed to be *Receptive field* of the neuron [31]

.

that height and width of the inout and output results in same.

Parameter Sharing reduces the number of parameters. For example consider the Conv1 layer with input *Image* size of [224*224*3], the kernel size f=3 (i.e $3 * 3$), stride S=2, depth K=64 and padding P=0 then the number of neurons are given by $(\frac{W-f+2P}{2} + 1)$ [31] which gives rise to $((224 - 3)/2) + 1 = 112$. Then the output of the convolution layer have $112 * 112 * 64 = 802,816$ neurons and each neuron as $3 * 3 * 3 = 27$ weights and 1 bias. Altogether, this sums up to $802,816 * 28 = 22,478,848$ parameters on the Conv1 layer of the ConvNet alone. It is clear that the number is very high. To reduce the number of parameters, one reasonable assumption is made use: the neurons are constrained in each depth slice to use the same weights and bias. By using this concept of parameter sharing the Conv layer has only 64 unique set of weights, this gives rise to a total of $96 * 3 * 3 * 3 = 2,592$ unique weights, or 2,656 parameters (+64 biases).

Pooling Layer

This layer functionality is to progressively reduce the spatial size of the representation of input and even reduce the number of parameters which also assist in controlling overfitting. Usually *Max* operation is employed independently over each depth slice of the input and gradually resizes it spatially. Pooling layer down-samples every depth slice in the input both in height and width, hence discarding redundant activations.

Normalization

To have some sort of inhibition scheme in CNN architecture *normalization layers* are used. From neurobiology we have the concept called *"Lateral Inhibition"*. Lateral inhibition is the capacity of the excited neuron to suppress its neighbour so that *local*

maxima are enhanced. In the VGG architecture they make use of *Local Response Normalization(LRN) layer* to employe the concept of inhibition. When we have ReLU neurons, using LRN normalization scheme is advantageous because ReLU neurons have unbounded activation and we need LRN to normalize that. This make feasible to have high frequency features with a large response. Performing normalization around the local space of the excited neuron makes it more sensitive and it will subdue the response that are uniformly large in any given local neighbourhood. LRN basically boost the neurons with relatively large activations by employing the concepts of *lateral inhibition* across the channels . Section 3.3 in [4] gives in-depth understanding regarding the operation of LRN when using the ReLU neurons.

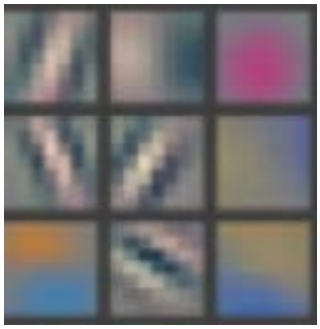
Fully connected layer

This layer is similar to the layers in the *Regular Neural Network*, where a single neuron in the layer is connected to all the. From the fig.8 the final fully-connected layer (fc layer) reduces to $1 * 1 * 1000$, because the number of classes that we have in ImageNet [32] is 1000 classes. On top of this layer, from the fig.8 we see the *softmax layer* which is responsible for probabilistic score for each class that is relative to the given input data.
The *Linear classification* of features into subsequent classes by ConvNets is explained in the subsequent sections. In the next subsection visualization of the features that the ConvNets learn inherently is discussed.

3.1.2.2 Visualizing the features learnt by ConvNet

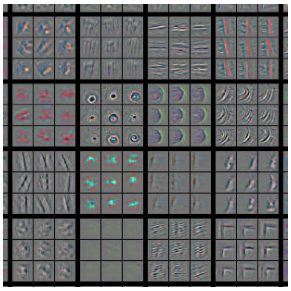
Its very important to know why the deep learning outperforms the shallow approach, the main advantage of deep learning is they learn the features that are required for the task inherently over the training. The visualization technique proposed by [34], helps us to have a look at the input stimuli that fires the feature maps at different layer in the chosen ConvNet model. The visualization technique proposed by [34] is with the *Deconvnet*, the approach maps the feature activities back to pixel space. In the paper they use the popular model by [4] and perform training on ImageNet 2012 training set, fig.10 shows feature visualized on train model at different layers. Top nine activations in a feature map at each layer is shown, alongside the feature maps you can find the corresponding image patches. In Layer 1 we notice that the feature map gets exited for a particular edge orientation. Layer 2 excites for complex edge/colour conjunctions and corners. As you the layer 3 responds for more complex invariance, firing for similar structures in the patches. In layer 4 you can find it activates for more significant class specific variations like dog faces; birds legs. In the last layer feature maps visualized we see the feature map responds for significant pose variations.

Deep visualization toolbox by [35] provides for better understanding and visualizing of learnt features by ConvNet model, and also it allows for better development of further CNN architecture.

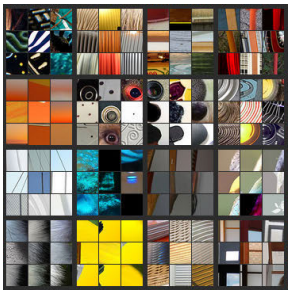


(a) corresponding image patch Layer 1

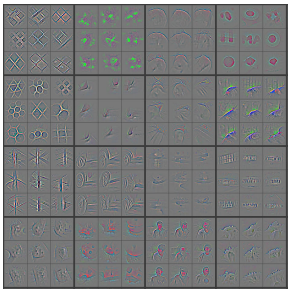
(b) Layer 1 Feature map



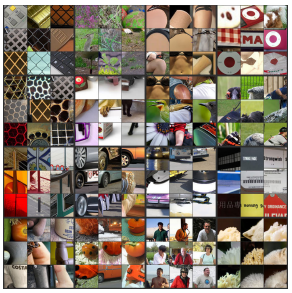
(c) corresponding image patch Layer 2



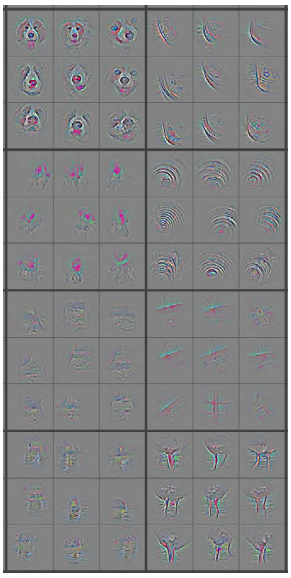
(d) Layer 2 Feature map



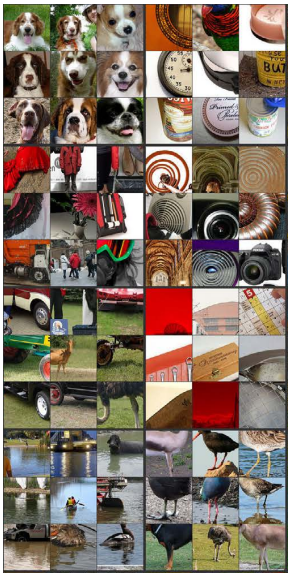
(e) corresponding image patch Layer 3



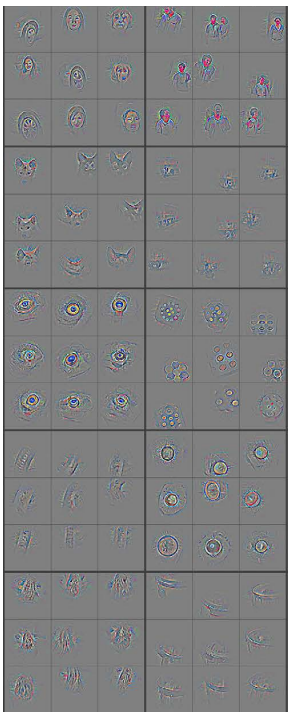
(f) Layer 3 Feature map



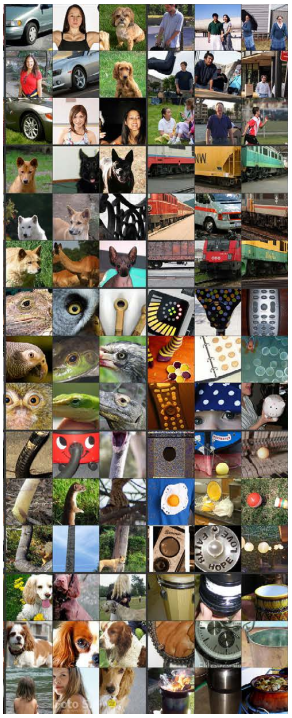
(g) corresponding image patch Layer 4



(h) Layer 4 Feature map



(i) corresponding image patch Layer 5



(j) Layer 5 Feature map

Figure 10: Illustration of feature activations and corresponding image patches
Notes: all the figures are taken from the work done by [34]. The images are best appeared when viewed in electronic form.

3.2 Image classifiers

The task of classifiers in the image classification problem is to map the given image to a single label from pool of fixed set of categories. Some of the linear classifiers are k-NN, SVM and softmax. The k-Nearest Neighbour (kNN) classifier labels test images by comparing them to the images from the training set. The disadvantage of the k-NN is it must remember all the training data to make comparison with the test data, this leads to downside of k-NN because its space inefficient and it could be seen that the dataset ranges to gigabytes in size. Secondly classifying is expensive because it needs to make comparison with all training data. Due to the downside of the k-NN we just focus on how a SVM and softmax classifiers are used in shallow learning and deep learning approaches. Unlike the k-NN most of the linear classifiers need not retain training data they just learn the parameters which does the classification task. In the further subsection we find out how Support Vector Machines (SVM) and softmax are used to formulate score function and loss function.

Support Vector Machines (SVM)

Classification of the given data is a common task in machine learning. In a classification problem, the learner approximates a function which can map a given vector

data into one of the various class labels. In supervised setting, it is done by looking at a set of input-output examples of the function. The finite input-output example data known as *training data* is used for learning the classification function. Support Vector Machines (SVM) is one of the successful supervised learning methods that have strong theoretical foundations in solving classification problem in. The trained SVM decision function would classify the unseen example data accurately. High generalization capability and simple linear classification ability are the main reasons for the success of SVMs. Linear classification of a two-class case (+ve class and -ve class) is considered unless specified. Given a set of a d-dimensional vectors, *linear classifier* tries to separate them with a R^d dimensional hyperplane this is the basic idea behind SVM. There exists many hyperplanes that classify the data. Margin is defined as the distance between the nearest samples on both sides of the hyperplane, SVM are designed to choose the hyperplane that has the largest margin between the two classes. Basic theory of SVMs for different cases is discussed in the following sections [36, 37, 38].

Notations: To describe the task in mathematical terms, we introduce the following notations

- an example data point is denoted by $x \in R^d$,
- class membership for a data point is denoted by $y \in \{+1, -1\}$,
- the set of training examples is denoted by $X = \{x_1, x_2, ...x_n\}$,
- class labels for the training set is denoted by $Y = \{y_1, ...y_2\}$

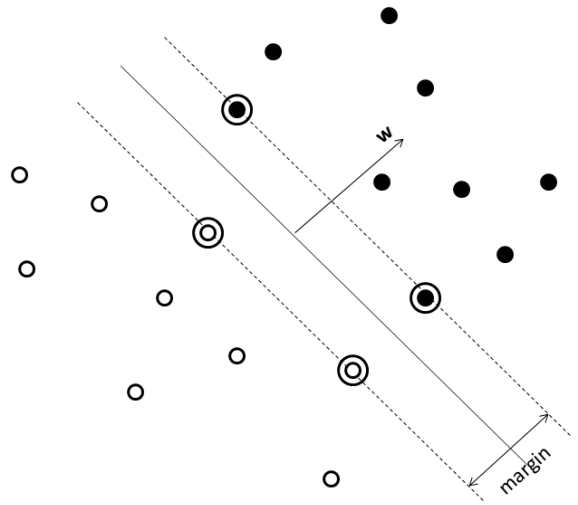


Figure 11: Separable SVM case, linear hyperplane can be drawn [36]

3.2.1 Linear Support Vector Machine

3.2.1.1 Hard-Margin case

SVM is the knowledge inherited from the statistical theory. In machine learning, classification of the given data into a specific class out of many very important. When dealing with classification problem, a learnable function F : $\mathbb{R}^d \rightarrow \{-1, +1\}$ is required that defines the class of a given example x. F [36] in the SVMs is given as a linear function i,e.

$$F = sign(f(x)) \tag{3.5}$$

The decision function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, which is responsible for transforming d dimension data to class label. This transformation is defined by the hyperplane separating the two classes. An hyperplane separating the two classes is having the form:

$$w^1x^1 + w^2x^2 + ... + w^dx^d + b = 0 \tag{3.6}$$

where x^1,x^2,....,x^d are the components of x , w^1,w^2,....,w^d are the components of the weight vector w and b is the bias. Decision function f can be written as:

$$f(x) = w \cdot x + b \tag{3.7}$$

$w \cdot x_i + b = 0$ is satisfied when the points x_i lie on the hyperplane, where w is normal to the hyperplane, $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin, and $\|w\|$ is the Euclidean norm of w. The minimum distance from the separating hyperplane to the closest positive and negative example is given by $d+$ and $d-$ respectively. Therefore, $d^+ + d^-$ defines the margin of a separating hyperplane. In case of linearly separable data, the support vector algorithm simply looks for the separating hyperplane with largest margin. As the data is linearly separable in this case, we can select two hyperplanes in a way that there are no points between them and maximize the distance between them. These hyperplanes can be written as follows:

$$w \cdot x + b = 1$$
$$w \cdot x + b = -1$$

The margin which is the distance between those hyperplanes will be equal to $\frac{2}{\|w\|}$. As we want to maximize the margin, we want to minimize the term $\|w\|$. It would be difficult to solve $min\|w\|$ because of the square root involved in calculation of $\|w\|$. Therefore, $min\|w\|^2$ is used as the optimization problem to make it easier. Also, as we do not want any data points falling into the margin, we add the following constraints:

$$w \cdot x + b \geq 1 \forall \quad x_i \quad with \quad y_i = 1,$$
$$and$$
$$w \cdot x + b \leq 1 \forall \quad x_i \quad with \quad y_i = -1$$

After combining the above inequality constraints into a single constraint, we will have the following primal optimization problem. This is a quadratic programming problem.

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{3.8}$$

with reference to,

$$y_i(w \cdot x + b \geq 1) \tag{3.9}$$

Note that factor $\frac{1}{2}$ is used for mathematical convenience. The Lagrangian formulation [36, 37, 38] of the above problem which replaces the inequality constraints with equality constraints.

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(w \cdot x + b) - 1] \tag{3.10}$$

where $\alpha_i \geq 0$, $\forall i$ are lagrangian multipliers for each of the inequality constraints in equation(3.9). The above Lagrangian is maximized with respect to α_i, and minimized with respect to w and b. Consequently,at this saddle point, the derivatives of L with respect to primal variables must vanish,

$$\begin{gathered}
\frac{\partial}{\partial b} L_P(w, b, \alpha) = 0 \\
\frac{\partial}{\partial w} L_P(w, b, \alpha) = 0 \\
\alpha[y_i(w \cdot x + b) - 1] = 0 \quad \forall \quad i \\
\alpha_i \geq 0
\end{gathered}$$

which leads to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.11}$$

and

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{3.12}$$

This problem is solved by using standard quadratic programming techniques. It can also be solved by solving its dual problem, which is easier and gives the same solutions as the one obtained by solving the primal version. Due to the convexity of the primal optimization problem, the solution is unique but the coefficients α_i need not be unique. The points for which α_i are non-zero are called as support vectors. Fig. 11 summarizes the situation for a 2-dimensional data, plotting support vector

points with extra circles around them. Dual form of the optimization problem is written as

$$L_D = \max_{\alpha \in \mathbb{R}^n} \{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \} \tag{3.13}$$

$$w.r.t \alpha_i \geq 0, i = 1, ..., n \tag{3.14}$$

$$and \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.15}$$

If we substitute the Equation (3.12) in the original decision function Equation(3.7), then we have

$$f(x) = \sum_{i=1}^{n} \alpha_1 y_i x_1^T x + b \tag{3.16}$$

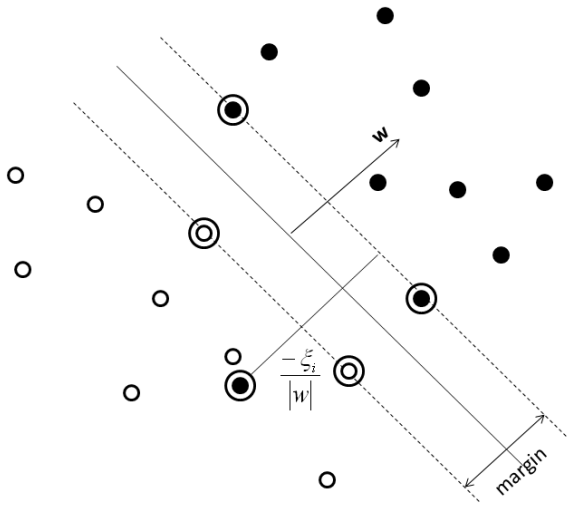


Figure 12: Non-separable SVM case, linear hyperplane cannot be drawn [36]

3.2.1.2 Soft-Margine case

Till now, we showed how SVMs are designed when that the training data is linearly separable. When the data is not linearly separable, there is no feasible solution and hard-margin SVM is unsolvable. An example of this can be found in the fig.12. This section shows how hard SVMs are modified to apply them in inseparable case. In order to assign the penalty on the errors, *non-negative slack variables* $\xi_i \geq 0, i = 1, ...n$ are introduced in the equation (3.9) as follows

$$(y_i(w \cdot x_i) + b) \geq 1 - \xi_i \quad for \quad i = 1, ..., n \tag{3.17}$$

$$\xi_i \geq 0 \quad \forall \quad i \tag{3.18}$$

For any point in the training data x_i (Fig. 12), if $0 < \xi < 1$, the data do not have the maximum margin but are still correctly classified. But if $\xi \geq 1$, the data

are misclassified by the optimal hyperplane. Therefore, we have $\sum_i \xi_i$ as an upper bound on the number of training errors. In order to assign an extra cost for these errors, the objective function to be minimized will be

$$Q(w, b, \xi) = \|w\|^2 + C(\sum_i \xi_i^k) \tag{3.19}$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1 - \xi_i \text{ for } i = 1, ..., n \tag{3.20}$$

In the above equations, C is the margin-parameter to be chosen by the user, which determines the trade-off between the maximization of the margin and minimization of the classification error. This is a convex programming problem for any positive integer k for $k = 2$ and $k = 1$ it is also a quadratic programming problem, and the choice $k = 1$ has the further advantage that neither the ξ_i , nor their Lagrange multipliers, appear in the dual optimization problem. We call the obtained hyperplane the soft-margin hyperplane. When $k = 1$, we call the support vector machine as the *L1* soft-margin support vector machine and when $k = 2$, the *L2* soft-margin support vector machine. First we shall discuss *L1* soft-margin support vector machines. Similar to the case of separable case, at first lagrangian multipliers are introduced in the optimization function as follows

$$L_P(w, b, \xi, \alpha, \beta) \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i(y_i(w \cdot x_1) + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_1 \xi_i \tag{3.21}$$

where α_i,β_i,$\forall_i$ are non-negative Lagrangian multipliers. For optimal solution, the following KarushKuhn-Tucker (KKT) [36, 37] conditions should be satisfied

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial w} = 0 \tag{3.22}$$

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial b} = 0 \tag{3.23}$$

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial \xi} = 0 \tag{3.24}$$

$$\alpha(y_i((w \cdot x_i) + b) - 1 + \xi_i) = 0 \text{ for } i = 1, ..., n \tag{3.25}$$

$$\beta_i \xi_i = 0 \quad \forall_i \tag{3.26}$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0 \text{ for } i = 1, ..., n \tag{3.27}$$

Using equations (3.22) to (3.24) on equation (3.21), we arrive at the following

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{3.28}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.29}$$

$$\alpha_i + \beta_i = c \text{ for } i = 1, ..., n \tag{3.30}$$

substituting equation (3.21), we obtain the following dual problem

$$L_D 9\alpha \equiv \max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j (x_i, x_j) y_i y_j \tag{3.31}$$

$$\text{subject to the constraints; } 0 \leq \alpha_i \leq C \tag{3.32}$$

$$\sum_i \alpha_i y_i = 0 \text{ for } i = 1, ..., n \tag{3.33}$$

Not that the main difference now is that the α_i is now bounded by C. Substituting equation (3.28) in equation (3.27) now the decision function becomes,

$$f(x) = \sum_{i=0}^{n} \alpha_i y_i (x_i \cdot x) + b \tag{3.34}$$

where n is the number of training samples out of which the feature vectors for x_i,for which α_i is non-zero are called as support vectors [36, 37, 38].

3.2.2 Softmax

CNNs for image classification includes two major functions known to be **score function** and **loss function**. Score function and loss function are responsible for mapping the raw data into class scores and quantifying the class score with the ground truth labels respectively. The loss function is minimized with respect to the parameters of the score function and this event of minimization in supervised deep learning is termed as optimization problem [31, 33].

3.2.2.1 Score function

Linear classifier is a score function which maps the raw data into the class scores. Consider the training dataset consisting of images $x_i \in R^D$, with each image bounded to an label y_i. Here $i = 1, 2, 3, ...N$ and $y_i = 1, 2, ...K$. That is, there are N examples and K discrete categories. The linear score function can be defined as $f : R^D \rightarrow R_K$, which maps the raw data to the class scores. A simplest function can be a linear classifier, i.e;

$$f(x_i, W, b) = Wx_i + b \tag{3.35}$$

Parameters W is called as **weight**, and b is often to be called as **bias vector** beacuse it influence the output class scores without interacting with the input data x_i. Several key points in process of image classification using CNNs are to be noted:

- Usually the input data (x_i, y_i) are consider to be fixed and given, hence we have no authority over them. During training of CNN we learn the **weights** interchangeably also termed as parameters **W,b** are learnt. We learn the parameters such a way that the class score and the ground truth label has minimum error.

- Advantage of this approach is, the training data is used to learn the parameters **W,b** and during testing we just need the learnt parameters **W,b** and we discard the necessity of complete training data to be known.
- Classifying an image is just involved with matrix multiplication and addition, which results in faster classification when compared to methods that involve comparing test image with all of the training data [31, 33].

3.2.2.2 Loss function

As from the previous section we know that we don't have control over the input data(x_i, y_i), the only control we have is on the parameters ***W,b***. The loss function is responsible to minimize the error between the predicted score and ground truth label of training image. The loss will be high if we are performing a poor job of classifying the training data, and it will be minimum when performing better. There are several ways to define the loss function but we consider only limited types [31].

Multiclass Support Vector Machine loss

Multiclass support Vector Machine loss also termed as SVM loss wants the correct class score for each image to have a higher score compared to the incorrect class with an degree of some fixed margin Δ. Consider we are given an i^{th} image with class label y_i which specifies the index of the correct class. The score function $f(x_i, W, b)$ gives the class scores (class scores are abbreviated as **s**). For our understanding the class score of j^{th} class is the j^{th} element: $s_j = f(x_i, W, b)_j$. The Multi-class SVM loss for the i^{th} image is formalized as:

$$L_i = \sum_{j \neq y_i} max(0, s_j - sy_i + \Delta) \tag{3.36}$$

You can notice in the formal (3.36) [31, 33], we perform thresholding at zero $max(0, -)$ this is often termed as **hinge loss**. People also use the squared hinge loss SVM or L2-SVM , which has the form $max(0, -)^2$. But most of the time we find standard SVM loss function used [31, 33].

Softmax classifier

Gives intuitive output has a probabilistic interpretation, where as the the SVM outputs scores for each class. The score function $f(x_i, W, b)$ remains unchanged likewise that is used in the SVM classifier, but the scores are interpreted as the unnormalized log probabilities for each class and the hinge loss is replaced with the **cross-entropy loss**, that takes the form:

$$L_i = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}) \tag{3.37}$$

or equivalently

$$L_i = -f_{y_i} + log \sum_j e^{f_{y_j}}$$

where, f_j is the j^{th} element of the vector of class scores f. In the earlier case SVM loss function the full loss for the dataset is the mean of L_i (even including the regularization term) over the training dataset. The fraction inside the log function in (3.37) [33, 31] is the **softmax function**, it squashes the real-valued scores to a vector of values between zero and one the sum of the vector sum ups to one [31, 33].

3.3 Maximum Likelihood Estimation

3.3.1 The Likelihood Function

Lets consider some notations for defining the Likelihood function, the probability distribution that we choose is represented by $f(\cdot)$. The probability distribution is characterized by the parameter θ(θ is vector of parameter, $\theta = \theta_1, \theta_2, ...\theta_k$). The parameter space is given by Ω, it is the set of all possible vector of parameters. Inference is made using observed data to make an statement about the parameter that governs the model, observation of data from the distribution is given by:

$$X \sim f(x|\theta)$$

The observed variables are random $X_1, X_2, ..., X_n$ which are independent and identically distributed (iid) due to that they are drawn independently. The parameter θ is learnt by observing the data x. Inferencing using the Bayes' Rule; as the aim of the inference is to estimate the probability that the parameter governing the distribution i.e θ is conditional on the samples that were observed, which is denoted as $\mathbf{x}$. The probability that is to be estimated is denoted as $\xi(\theta|x)$. We can define this probability using the Bayes' Rule [39, 40]:

$$\xi(\theta|x) = \frac{f(x|\theta)\xi\theta}{g_n(x)} = \frac{f_n(x|\theta)\xi\theta}{\int_\Omega f_n(x|\theta)\xi\theta} \tag{3.38}$$

$$\text{for } \theta \in \Omega$$

In (3.38) the denominator is just the constant in the pdf of θ, The (3.38) can be rewritten as:

$$\underbrace{\xi(\theta|x)}_{posterior} \propto \underbrace{f_n(x|\theta)}_{likelihood} \underbrace{\xi(\theta)}_{prior} \tag{3.39}$$

According to R.A Fisher [39, 40] the likelihood is termed as:

$$L(\theta|x) \propto f_n(x|\theta) \tag{3.40}$$

Knowledge regarding the parameter based on the observed data is summarized in the Likelihood function. From equation (3.40) [40, 39] it is noticed that the likelihood

function is the function of θ. usually the Bayesian inference aims at estimating the posterior probability of the parameter distribution, $\xi(\theta|x)$. We can conclude that, the data is featured as the joint density function which is conditional dependent on the parameters of the hypothesis model, i.e $f_n(x_1, x_2, ..., x_n) = f_n(x|\theta)$. When we consider the samples to be iid, we get $f(x_1|\theta) \cdot f(x_2|\theta) \cdot \cdots \cdot f(x_n|\theta)$. The function $f_n(x|\theta)$ is known as **likelihood** [39, 40].

3.3.2 Maximum Likelihood Estimator(MLE)

Lets consider the same n iid samples which are characterized by the parameter θ_o, MLE finds the $\hat{\theta}$ that is the best estimator of θ_o. The main focus of MLE [39, 40] is to find the best θ that maximizes the likelihood function, it can also be stated that the MLE finds θ that is likely to have generated the vector of observed data, x. Each point in the parameter space Ω is assigned a value to indicated how likely is the point in the space to have generated the observed vector of data. We saw in subsection 3.3.1 the likelihood function is proportional to joint probability distribution of the data. All the information that we have about the parameters given the observed data is summarized by the likelihood function. The method of maximum likelihood obtains the values for the model parameter that define a distribution that is most likely to have resulted in the observed data. The likelihood is not a probability of the data defined upon some conditional parameter. Instead it is a measure of **relative** uncertainty about the probable values of θ, given by Ω. The likelihood as proportional to the joint probability of the data conditional on the parameter can be formally define as:

$$L(\theta|x) \propto f(x|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

As we discussed the MLE estimates θ, which we denote as $\hat{\theta}_{MLE}$ that maximizes the likelihood function. That particular value of θ is most likely to have generated the data. We can write the MLE mathematically as:

$$\hat{\theta}_{MLE} = \max_{\theta \in \Omega} L(\theta|x) = \max_{\theta \in \Omega} \prod_{i=1}^{n} f(x_i|\theta)$$

It is possible to work with the log-likelihood function because maximizing the logarithm of the likelihood is same as maximizing the likelihood (due to monotonicity):

$$\hat{\theta}_{MLE} = \max_{\theta \in \Omega} logL(\theta|x) = \max_{\theta \in \Omega} l(\theta|x) \sum_{i=1}^{n} log(f(x_i|\theta))$$

Finding analytically the MLE involves taking the first derivatives of the log-likelihood, setting it to **zero** and solving for the parameter θ. Later checking that we have indeed obtained a maximum by calculating the second derivative at the critical value and checking that it is negative. Let us define **score** as the first derivative of the

log-likelihood function with respect to each of the parameters(gradient). For a single parameter:

$$S(\theta) = \frac{\partial l(\theta)}{\partial \theta}$$

Then the first order score is to zero and solved for θ. When there are multiple parameters (a vector θ pf length k), the score is defined as [39, 40]:

$$S(\theta) = \nabla l(\theta) = \begin{pmatrix} \frac{\partial l(\theta)}{\partial \theta_1} \\ \frac{\partial l(\theta)}{\partial \theta_2} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial l(\theta)}{\partial \theta_k} \end{pmatrix}$$

MLE Estimation of Mean and Variance for Sampling from Normal Distribution

The vector of samples $x_1, x_2, x_3, ..., x_n$ form the normal distribution the parameter that define this distribution are unknown, $\theta = (\mu, \sigma^2)$. MLE focus to find estimator for θ.
Likelihood for this distribution is given as:

$$L(\theta|x_1, ..., x_n) = f_n(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} exp[-\frac{1}{2\sigma^2}\sum_{i=1} n(x_i - \mu)^2]$$

Applying log of the likelihood function, we get:

$$\ell(\theta|x_1, x_2, ...x_n) = -\frac{n}{2}log(2\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1} n(x_i - \mu)^2 \tag{3.41}$$

The likelihood function needs to be optimized with respect to paramteres μ and σ^2, where $-\infty < \mu < \infty$ and $\sigma^2 > 0$.
At first we assume that we know σ^2 as known and lets find $\hat{\mu}(\sigma^2)$. Now, take the partial derivative of the log-likelihood with respect to the μ parameter and set it equal to **zero**:

$$\frac{\partial \ell(\theta)}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1} n(x_1 - \mu) = 0$$
$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x_n}$$

We check the achieved **estimator** is the maximum by taking the second derivative:

$$\frac{\partial \ell(\theta)}{\partial \mu} = \frac{-n}{\sigma^2} < 0 \quad \forall \quad n, \sigma^2 > 0$$

Now lets estimate for the variance. We know from above results $\hat{\mu} = \bar{x}_n$, now lets substitute this condition for μ in the (3.41) and taking the derivative with respect to σ^2:

$$\frac{\partial^2 \ell(\theta)}{\partial \sigma^2} = -\frac{n}{2}\frac{1}{\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1} n(x_i - \bar{x}_n)^2 = 0$$

Solving for σ^2, we arrive at:

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1} n(x_i - \bar{x_n})^2$$

The **Maximum Likelihood Estimators** [39, 40] for μ and σ^2 are:

$$\boxed{\hat{\mu} = \bar{X} \quad and \quad \hat{\sigma^2} = \frac{1}{n}\sum_{i=1} n(X_i - \bar{X_n})^2} \tag{3.42}$$

Chapter 4

Overview of methods

4.1 Fast-RCNN and Bayesian classifier fusion

This section provides design procedure followed for pedestrian detection using the concepts of a deep learning, naive Bayes classifier, and fusion of both the classifiers scores at the post processing stage. Fig.13 provides an visual overview of the approach used to achieve the task. The classifier $\psi_{FastRCNN}$ is a *convolutional neural network* for object detection. Keeping in mind the timing constrain for the thesis, development of the state-of-art ConvNet for the person detection is out of scope of this work. The utilized ConvNet architecture is the work done by [9] for object detection on Pascal VOC dataset [7]. The architecture of the used *Fast RCNN* [9] model and other relevant information regarding model will be discussed in the upcoming chapters.

Classifier ψ_{AR} which is a simply a *naive Bayes classifier* that provides the score for each region hypothesis based on the simple context feature; aspect ratio(AR) of person class.

Scores $\xi_{FastRCNN}$, ξ_{cb_c} given by classifiers $\psi_{FastRCNN}$, ψ_{AR} is a non-linear function, that attaches the confidence value to each of the generated region hypothesis. At the post processing stage the output scores $\xi_{FastRCNN}$, ξ_{AR} from two different classifiers are fused inside a *supervisory classifier* ψ_H. Fusion of scores by the *supervisory classifier* ψ_H results in *override* of the earlier classifiers scores resulting in new score ξ_H. This additional information incorporation at the post processing stage of CNNs results in performance increase. Further information regarding the operation of scores fusion method and override utility will be discussed in later chapters.

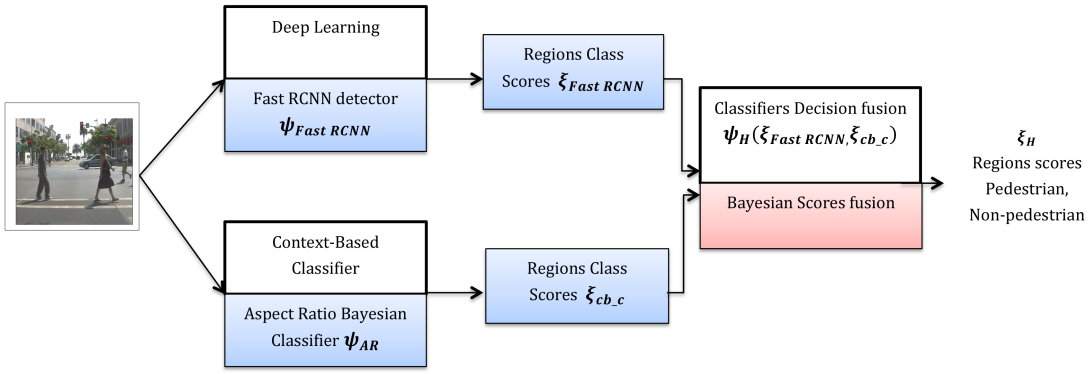


Figure 13: An overview of followed approach for person detection using deep learning, Bayes context based classifier and fusion classifier.
Notes: Classifier $\psi_{FastRCNN}$ is a deep learning method for object detection and classifier ψ_{AR} is context based Bayesian Classifier, the score for each region hypothesis; $\xi_{FastRCNN}$ and ξ_{cb_c} are given from two classifiers respectively. Final score ξ_H is obtained by fusing the score's from the classifiers $\psi_{FastRCNN}$ and ψAR.

4.1.1 Stereo information for contextual modelling

This part of the work is an outlook to show, how the stereo information will be effective for achieving better *precision* of the object detector. As you see from the fig.14 the *classifier fusion* ψ_F operation is similar to that of the model explained in above section. The classifier ψ_f gives the score ξ_F for each of the generated region hypothesis. The region hypothesis are thresholded by a specific confidence TH_S score to obtain *detected regions* with greater fusion scores than the threshold TH_S. The detected regions consists of many false positive that can be eliminated by using the stereo information. detected regions are projected on to the ground plane, that is estimated using the computed disparity map, camera intrinsic and extrinsic parameters. The height between the regions and ground plane is computed, those height greater than certain threshold TH_H are eliminated. This reduces many false positives, and further the filtered detection regions are subjected to another stage of thresholding. Where the regions real height and width in meters are computed using average disparity value of the image. Those regions with height and width greater than the thresholds TH_h and TH_w are discarded. Finally we have the detections that hold no false positives or less number of false positives which are not possible to eliminate using stereo information.

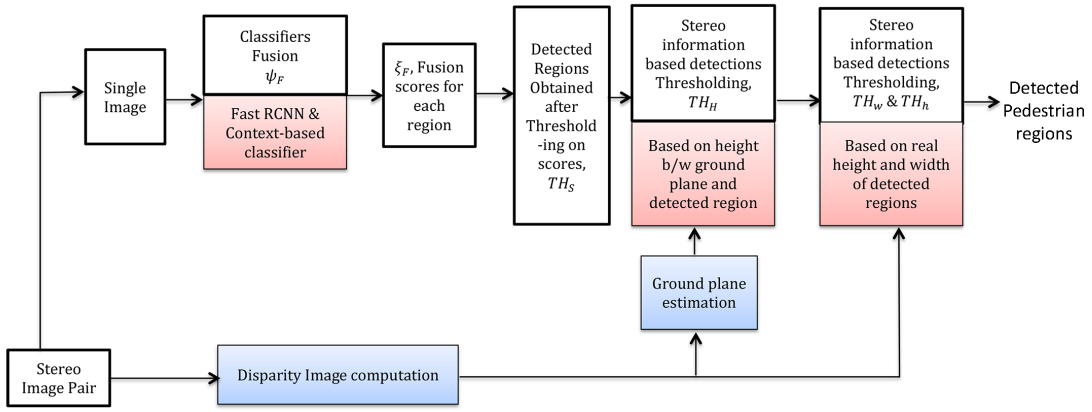


Figure 14: An overview of thresholding detections using stereo based contextual information

Notes: TH_S: Threshold value for obtaining regions having fusion score greater or equal to the specified value; TH_H: Threshold value specifying acceptable height in meters between detected regions and estimated ground plane; TH_w and TH_h: are the acceptable height and width(in meters) of the detected regions.

4.2 SVM-HOG and Bayesian classifier fusion

In the earlier subsection the scores from two different classifiers were fused to obtain the new score. This section make use of decisions made by the marginal classifiers for obtaining new decision from the supervisory classifier. Fig.15 provides an visual overview of the shallow approach used for detecting pedestrian in the given environment. The classifier $\psi_{SVM-HOG}$ is the followed pipeline for pedestrian detection that uses shallow approach procedure. This classifier is responsible for mapping each region's HOG descriptor to a discrete decision $\delta_{SVM-HOG}$ using linear SVM.

Classifier ψ_{AR} which is a simple *naive Bayes classifier* provides the decision based on the simple context features of pedestrian class. Context feature of the object class used in modelling the Bayes classifier is a *aspect ratio*(AR) of pedestrian. The modelling details will be explained in detailed in the upcoming chapters.

Decisions $\delta_{SVM-HOG}$, δ_{AR} made by the classifiers $\psi_{SVM-HOG}$, ψ_{AR} respectively is a non-linear function of mapping from the feature space Π to discrete decision space Δ. Approached method by [41] is followed to fuse the decisions made from $\psi_{SVM-HOG}$, ψ_{AR} using the *supervisory classifier* ψ_H. Fusion of decisions by the *supervisory classifier* ψ_H will result in *override* of the decision made by the marginal classifier $\psi_{SVM-HOG}$, ψ_{AR}. The details of implementation will be discussed in the later chapters.

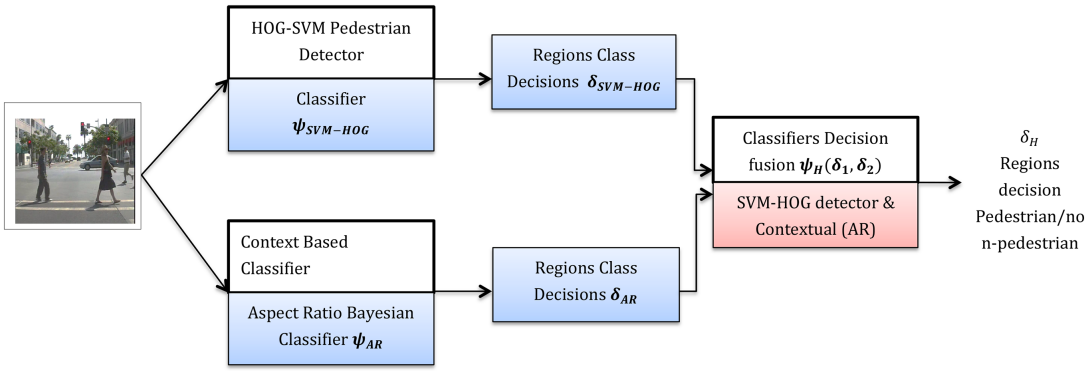


Figure 15: An overview of followed approach for person detection using SVM-HOG detector, Bayes context based classifier and decision fusion classifier.

Notes: Classifier $\psi_{SVM-HOG}$ and classifier ψ_{AR} are shallow approach for pedestrian detection, the decisions for each region hypothesis; $\delta_{SVM-HOG} \simeq \delta_1$ and $\delta_{AR} \simeq \delta_2$ are given from two classifiers respectively. Final decision δ_H is obtained by fusing the decisions from the classifiers $\psi_{SVM-HOG}$ and ψ_{AR}.

Chapter 5

Evaluation methodology

The same scheme laid out in the PASCAL object detection challenges [7] is used, where it is based on per-image evaluation. The detector takes in an image and returns a *bounding box* BB with its score or decision based on different followed approach. We threshold these BBs with the threshold confidence to get *detected* BB_{dt} with confidence greater than the threshold. Each BB_{dt} are matched once with the *ground truth boxes* BB_{gt} as per the given equation in (5.1) [7]. The match is accounted if the overlapping area is greater than the specified value A_{match}.

$$a_0 = \frac{area(BB_{dt} \bigcap BB_{gt})}{area(BB_{dt} \bigcup BB_{gt})} > A_{match} \tag{5.1}$$

Unmatched BB_{dt} are accounted as *false positives*(TN) and unmatched BB_{gt} are taken into count as *false negative*(FN), where as the matched BB_{dt} are counted to be *true positives* (TP). *True negatives* (TN) correspond to negatives that are not detected by the detectors as potential object. These decisions made by the detector can be represented in a structure known as a *confusion matrix* or contingency table. A confusion matrix is shown in Table 1.

Table 1: Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	*TP*	*FN*
Actual Negative	*FP*	*TN*

In machine learning, its not sufficient to state the algorithm or detector performance by presenting the accuracy. For having better empirical validation of the detector their is necessity to consider the Receiver Operator Characteristic (ROC) curves, Precision-Recall curves and very important in automotive applications is to use the plot with miss rate (MR) against false positives per-image (FPPI). The generic difference between these curves is the visual representation of the curves.

Given the confusion matrix it is possible to have all the plots that were stated earlier. The plot miss rate against false positives is obtained by varying the threshold on the detection confidence. With the confusion matrix as the reference it is possible

to construct a point in the ROC space and PR space. Equation.(5.2) to (5.7) [42] are made use to define point in each space. In ROC space, one plots the False Positive Rate (FPR) on the $x-$axis and the True Positive Rate (TPR) on the $y-$axis. The FPR measures the fraction of negative examples that missed classified as positive. The TPR measures the fraction of positive examples that are correctly labelled. In PR space, one plots Recall on the $x-$axis and Precision on the $y-$axis. Recall is the same as TPR, whereas Precision measures that fraction of examples classified as positive that are truly positive. In MR against FPPI plot, one plots the $x-$axis with the miss rate and the $y-$axis with the FPPI. The MR is the measure of true positives that were undetected and FFPI is the measure of FP per image at particular confidence threshold. As stated earlier the MR against FPPI is preferred in automotive applications, as there is typical upper limit on the acceptable false positive per-image rate independent of the pedestrian density.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$True\_Positive\_Rate = \frac{TP}{TP + FN} \quad or \quad TPR = Recall \tag{5.4}$$

$$Specificity = \frac{TN}{TP + FP} \tag{5.5}$$

$$False\_Positive\_Rate = \frac{FP}{FP + TN} \quad or \quad FPR = 1 - Specificity \tag{5.6}$$

$$Missrate = \frac{FN}{FN + TP} \tag{5.7}$$

Chapter 6

Implementation and performance study

6.1 Region hypothesis generation

Region of Interest (ROI) are crucial for pedestrian detection for localizing pedestrian along with the classification. With the given image many regions are generated for the classifier for classifying them as the object candidate.

6.1.1 Brute force method

This technique generate random set of regions with the help of distribution of pedestrian in Caltech pedestrian Dataset [1]. The heat map in the fig.16 points to distribution of the pedestrian in the Caltech dataset. In the developed brute force method we made sure to have large count of negative regions in the proposed ROI for detections. Several windows of different size (*width,height*) were scanned along the dimension of the input image relative to ground truth. The large count of negative regions is to study the efficiency of the used detector, mainly to study the reason for misclassifying the proposed regions. The red boxes on Fig.17a are the pedestrian ground truth where as the image is the sample from Caltech pedestrian dataset. The blue boxes on fig.17b visualize the regions proposed using brute force method (the proposed regions are restricted while plotting on the image to have better appearance). This region proposal system is incorporated in the Fast RCNN pipeline. In the upcoming section the complete architecture about Fast RCNN object detector from [9] is discussed. The large number of negative regions in the proposals will account to downside of the precision of the detector. The obtained results justifies this reason.

6.1.2 Multi-scale scanning window method

Pedestrian in the image occur in different scales. To have high recall, i.e the generated regions should overlap on maximum pedestrians (ground truths) in the given image. Therefore it is necessary to scan the image at different scales. In the proposed method by [2] a fixed size (*width,height*) detection windows/ROI are scanned over the spatial dimension of the image at different scale, so that it ensure that the percentage of *recall* is considerably very high. This can be visualized as image scale

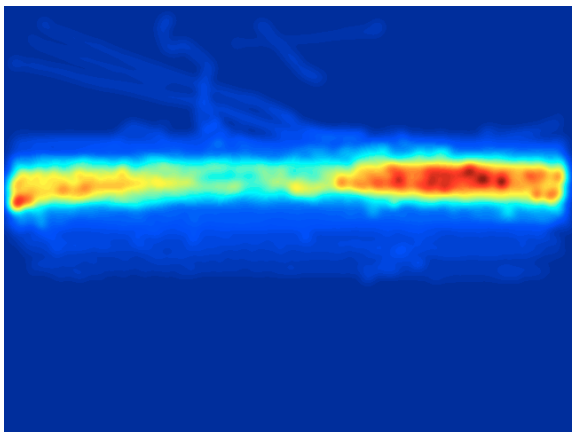
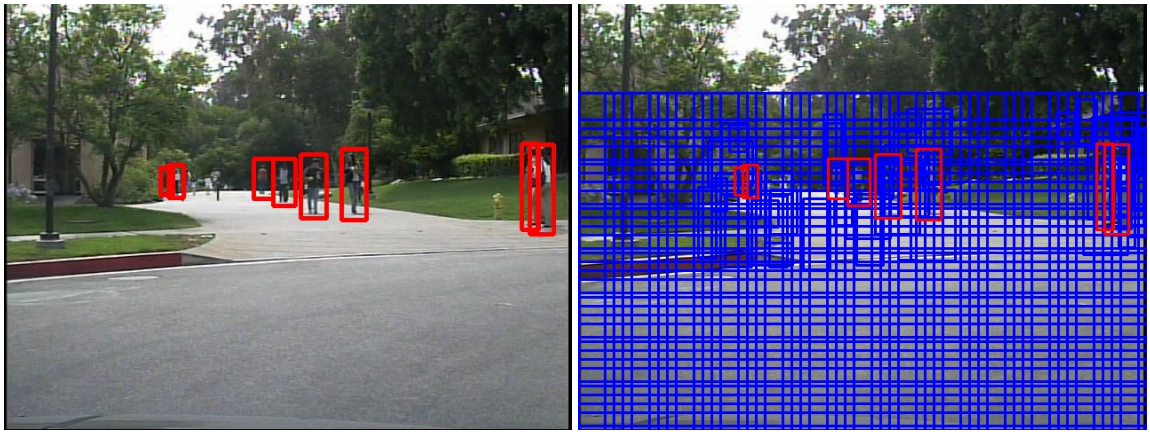


Figure 16: Heat map of pedestrian position in the Caltech dataset
Note: Figure courtesy from [1]



(a) A sample image from the Caltech pedestrian dataset[1]

(b) Regions proposed using the brute force method relative to ground truth data

Figure 17: Visualizing the proposed ROIs using brute force method

space pyramid, see fig.18. For implementation of this method we use the OpenCV libraries[24]. The function *void detecMultiscale()* allows us to specify the input parameter to specific required number of scales and window size of the ROI. The size of the ROI/detection window was set to bet $(48 * 96)$. The reason behind considering the mentioned window size will be discussed in the upcoming sections of this chapter. This approach is used as a region proposal system in shallow learning method (SVM-HOG detector pipeline).

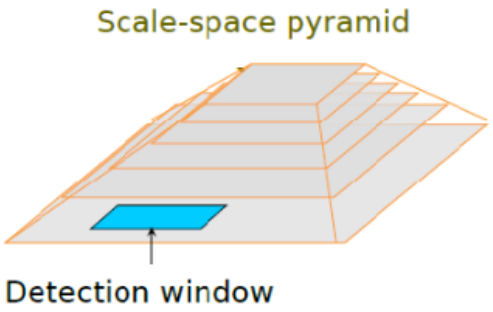


Figure 18: Detection window generation over scale-space pyramid
Note: Figure courtesy from [2]

6.2 Feature extraction and classification

6.2.1 Fast-RCNN architecture

Fast Region-based Convolutional Neural Network (Fast R-CNN) from [9] is a object detector which uses the principle of *Convolution Neural Network*(CNN). The fig.19 is the architecture of Fast R-CNN. The detector takes image and corresponding region proposals/ region of interest (ROIs) as the input. As you see from the figure the first process involves Conv-feature map (its the feature at the last convolution layer before RoI pooling layer in this architecture) computation. The responsibility for computation of Conv-feature map is Deep ConvNet that process the image with several basic layers as seen in section 3.1.2. The proposed ROIs are then projected on to the computed Conv-feature map. The region of interest pooling layer (RoI-pooling-layer) extracts fixed length feature vector for each of the proposed ROIs. The extracted feature vector by the RoI-pooling-layer for each of the proposed ROIs are forwarded through sequence of fully connected layers (*fc layer*). Then the fc layer branches into two parallel layers: One of the layer presents the softmax probability for each proposed ROIs over $K+1$ class ($+1$ class is indicates the background class) and the other output layer gives redefine position of the corresponding proposed ROIs as a tuple $(x1, y1, h, w)$; Where point (x1,y1) defines top-left corner and (h,w) are the height and width of the redefined ROI.

The architecture is dependent on the external ROI proposals. In the default Fast RCNN architecture pipeline the region proposal method used was the *selective search*

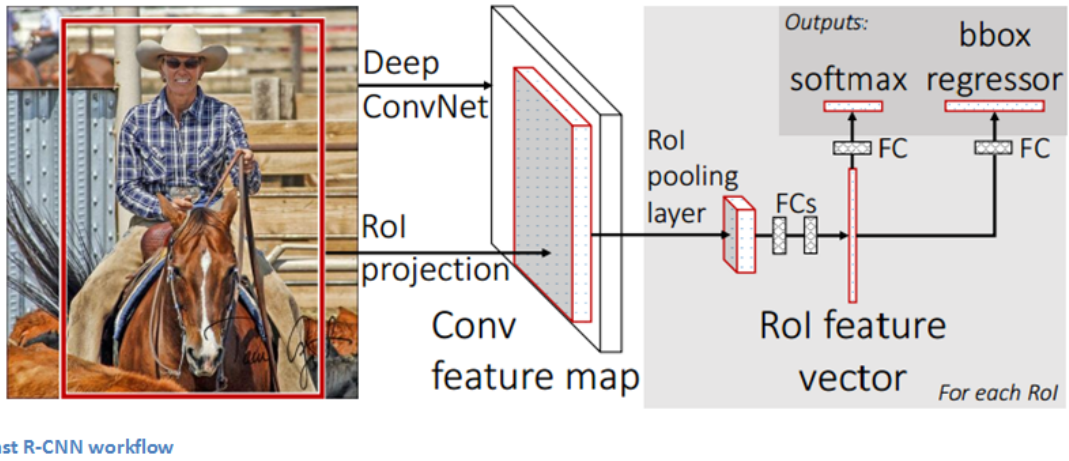


Figure 19: Architecture of Fast Region-based Convolutional Neural Network (Fast RCNN) [9]

method from [8]. The selective search method takes 3.79 seconds on an average for $\sim 2K$ ROI proposals, hence due to its limitations we make use of brute force method for generating region proposals. Approximately $\sim 2.5K$ ROI are proposed using brute force method. Lets look briefly about training procedure and other aspects involved in Fast R-CNN detector. As stated earlier the Fast R-CNN uses Deep ConvNet to process the image for learning Conv-feature map and fc layers outputs for each proposed regions the class scores $(K+1)$ and redefined ROI position (tuple of four real values (r,c,h,w)) for each class. The Conv feature map and bounding box regressor are trained over data which enables inherently to perform object detection task. The pre-trained classification network on ImageNet [32] were used to initialize Fast R-CNN, these pre-trained networks undergo few transformations. Pre-trained Conv-nets are those deep networks that are designed for general image classification task [4, 10, 18, 43].

1. An extra ROI-pooling-layer is inserted after the last convolution layer, which is made compatible with the first fc layer.

2. The pre-trained network on ImageNet are trained for 1000-way ImageNet categories, hence the networks' last fc layer and softmax are replaced with two sibling layers; as seen from the fig. 19, one brach with fc layer and softmax over $(K+1)$ categories and other for class-specific/category-specific *bounding-box regressors.*

3. The input to the pre-trained network was just an image, it has been modified to accept input data to be image and the corresponding region proposals.

The pre-trained classification networks from [10] was used to initialize the Fast R-CNN network that was responsible for classification over 1000-category classes on ImageNet challenge 2014. After initialization and transformation of the utilized pre-trained classification network fig.20, the Fast R-CNN is subjected to fine-tunning

Table 2: The Object Classes on which Fast RCNN was fine-tunned on.

PASCAL VOC Detection Classes
'__background__','aeroplane', 'bicycle', 'bird', 'boat','bottle', 'bus', 'car', 'cat', 'chair','cow', 'diningtable', 'dog', 'horse',,'motorbike', 'person', 'pottedplant','sheep', 'sofa', 'train','tvmonitor

for detection over $K+1 = 21$ categories on PASCAL VOC dataset [7] (K=20 object categories and 1 extra category for *background class*). The fine-tunning is a streamlined training procedure that jointly optimizes a softmax classifier and bounding-box regressors. During this training process a *Multi-task loss function* was used to compute loss jointly for ROI classification and bounding-box regression. The architecture is designed with efficient streamlined training procedure which allows to back-propagate the computed loss through the ROI-pooling-layer to minimize the loss function, the main advantage of streamlined training is that it eliminates the requirement of multistage training strategies followed in RCNN [6]. Stochastic gradient descent (SGD) is used to minimize the cost function during training. The work does not focus on retraining the network or fine tunning the network further on the specific data set. Training the detector has hardware and time constrains, hence it is focused in the future work. The complete information regarding the followed training procedure is provided in [9].

In the work we are interested only in *pedestrian* and *background* categories out of 21 categories that the Fast RCNN is trained over. The table 2 shows the PASCAL VOC 21 detection classes that the Fast RCNN was fine tuned over. The probability for pedestrian is given by the softmax probability for person class from the Fast RCNN detector and for background category the class probability is *1 - probability of pedestrian.*

The Results of the used Fast RCNN on Caltech pedestrian dataset is discussed in the next chapter. Further we fuse the Fast RCNN with an context based classifier for improving the detector performance. For fusing the Fast RCNN with an context based classifier we learn the scores of Fast RCNN for pedestrian and background class. The part of learning the scores will be discussed in next section.

6.2.2 HOG-SVM Pedestrian detector pipeline

Complete shallow learning implementation is achieved using OpenCV computer vision library [24]. The fig.21 gives an overview of the followed method for people detection using HOG-SVM pipeline. The followed approach is in accordance to the work from [2]. HOG descriptors capture a robust features that encode the pedestrian image so that a classifier can be trained to classify an object efficiently. As known many variants of HOG descriptors have been proposed from the date a original HOG-SVM pedestrian pipeline was published. The same original approach followed by [2] to perform pedestrian detection is followed, but with linear SVM trained on HOG descriptor that was captured on Daimler monocular dataset [44].

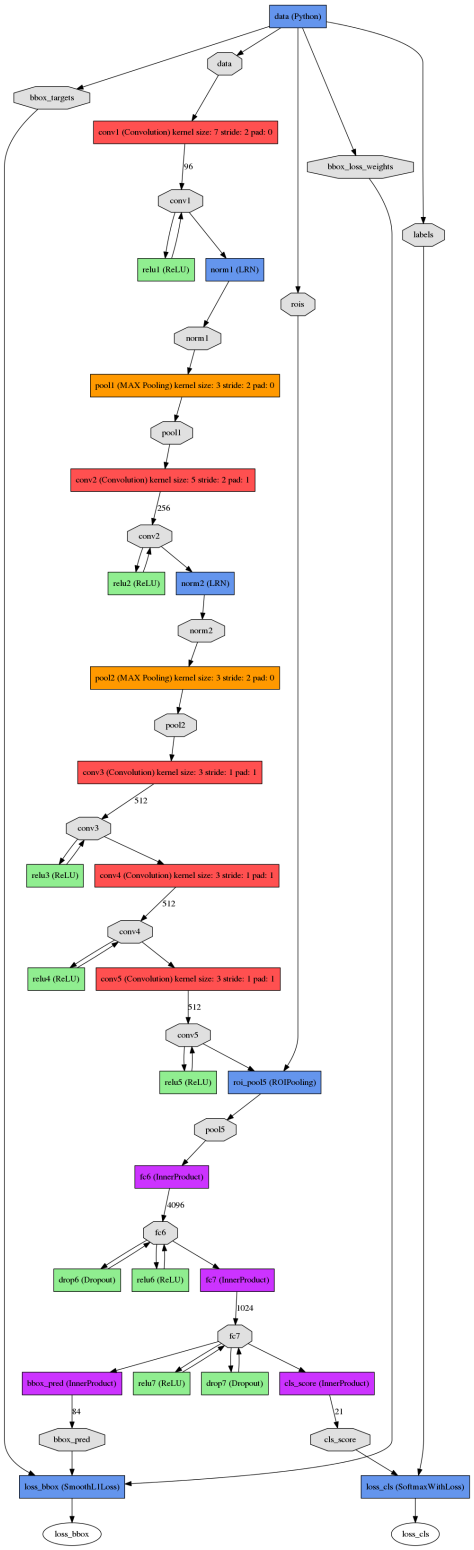


Figure 20: VGGNet from [10] transformed with accordance to Fast RCNN object detection architecture.

Note: This diagram was possible with the help of Caffe Framework Utility scripts.

Lets look in to the properties of the HOG descriptor used. The below line code explains the parameter setting to compute HOG descriptor;

Table 3: OpenCV function for defining the HOG-descriptor setting

cv::HOGDescriptor::HOGDescriptor(Size win_size=Size(48,96), Size block_size=Size(16, 16), Size block_stride=Size(8, 8), Size cell_size=Size(8, 8), int nbins=9, double win_sigma=DEFAULT_WIN_SIGMA, double threshold_L2hys=0.2, bool gamma_correction=true, int nlevels=DEFAULT_NLEVELS

- *win_ size*: defines the size of the window for which the HOG descriptor should process on in other way it is the detection window size. We set the size to be (48,96), because of the cropped image size of the object samples in Daimler dataset [44]
- *block _ size*: defines the size of the block, which is collection of cells. This is used to perform local normalization to compensate for the gradient energy variation over wide range.
- *block _ stride*: it tells how the block overlaps with other blocks so that it benefits the cell response with contribution of several components tot he final descriptor vector.
- *cell _ size*: the default cell size supported is (8,8)
- *nbins:* it defines the number orientation bins used to capture the gradient direction, Increasing the bins over 9 does not have any effect on results hence we use 9 bins.
- *win_ sigma* , *threshold _ L2hys*, and *gamma_ correction* are used to perform image preprocessing before to achieve counter action to image noise effects.

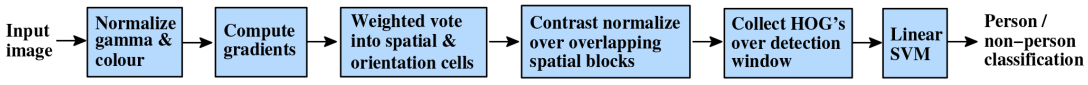


Figure 21: Shallow learning detection overview, HOG-SVM detector pipeline.
Note: Figure courtesy from [2]

The brief explanation of the followed detection pipeline is stated in section 3.1.1. The scanning window detection approach is explained in the section 3.1.1. In the earlier paragraph we saw the properties of the HOG descriptor that is used to encode the given image sample, each of this descriptor accounts to feature vector for linear SVM. The linear SVM is trained to learn positive and negative weight. In our work we use the pre-trained SVM weights, which is trained on the Daimler mono pedestrian dataset [44] rather than the default dataset INRIA pedestrian dataset used in

[2].We use of-the-shelf trained linear SVM weights on Daimler Mono dataset that is available in OpenCV computer vision framework. Followed training procedure of SVM-HOG detector is given in [2]. The function from OpenCV (table.4) allows us to set the SVM detector with the SVM weights that is pre-trained on Daimler mono dataset [26].

Table 4: OpenCV function for setting SVM detector with pre-trained SVM weights on [44] dataset.

cv::HOGDescriptor:: getDaimlerPeopleDetector()

With the SVM detector loaded with the pre-trained model it is active to perform linear classification of given sample that is encoded by the HOG-descriptor to either of two classed (*pedestrian, non-pedestrian*). As witnessed from fig.21, during testing/inference an input image is processed for normalization and gamma correcte. Then detection window is proposed by using the scanning image at multiple scales as explained in section 6.1.2 . The HOG descriptor is computed and the feature vector is further given to linear SVM for classifying the detection window which is a patch of the image for potential object existence. Each detection window/region in the image is classified either as pedestrian and non-pedestrian based upon its computed HOG descriptor feature vector. HOG descriptor computed on each detection window is weighted with the learnt SVM weights (see fig.2), based on its weighted results the detected region is provided a decision $\delta_{SVM-HOG}$. Those regions if classified as pedestrians are given decision $\delta_{SVM-HOG:ped}$ and if classified as non-pedestrian then it is assigned decision $\delta_{SVM_HOG:non-ped}$. As seen in the fig. 15, the top classifier $\psi_{SVM-HOG}$ is the followed HOG-SVM pedestrian detector pipeline explained in this section. Further decisions on Caltech pedestrian dataset are used to fuse with context-based classifier ψ_{AR}. The fusion classifierψ_H is explained further in this chapter.

6.2.3 Context-Based classifier

Contextual information allows use to improve the performance of object detector. This section explains about a classifier that deals with context information of object for classifying the proposed regions to respective classes.

Contextual information

There are many object contextual cues [23] that can be drawn from the image. Aspect ratio of an pedestrian is generic context information that can be utilized by the classifier for supporting precise detection of pedestrian. Hence we show in this section how we modelled a classifier with aspect ratio of the pedestrian. The work just shows how a additional information will improve the detector efficiency, aspect ratio of pedestrian is alone used but based on the available information in dataset we can consider many object cues. Form both the figures 13 and 15 we notice that the classifier ψ_{AR} is a context based classifier dealing with context information (aspect

ratio) of pedestrian to classify given region as pedestrian or non-pedestrian by defining scores or decision for each regions respectively. Caltech pedestrian dataset [1] is used to model the contextual classifier. *Set05* of Caltech pedestrian dataset alone is used to get the aspect ratio distribution information of the pedestrian object in the images. The fig.22 tells the distribution of aspect-ratio of the pedestrian object in Caltech pedestrian set05 dataset. From the fig.22 it is noticed that the distribution can be modelled as a *Gaussian function*.

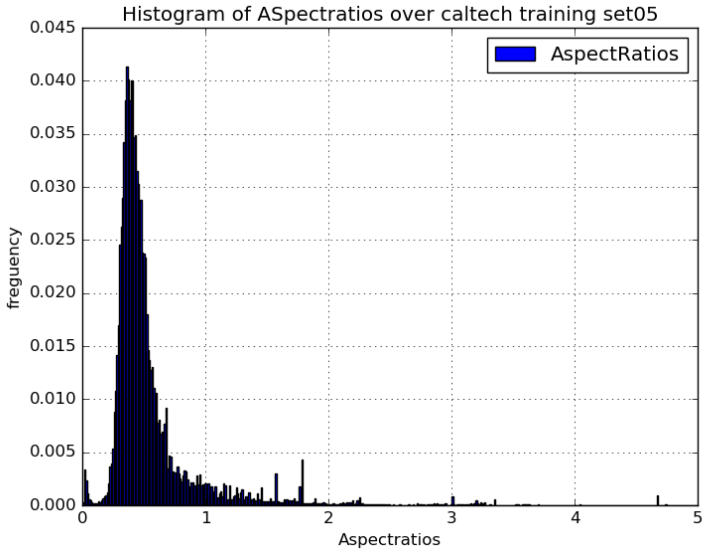


Figure 22: Histogram/distribution of aspect ratio's of pedestrian object in set 05 of Caltech pedestrian dataset

Modelling Bayesian classifier with contextual information

In this work we model the contextual classifier ψ_{AR} as seen in the figures 13 and 15 as *Naive Bayesian Classifier*. The naive Bayesian classifier is simple probabilistic classifier that make use of *Bayes' Theorem*. The Bayes' Theorem is given by equation (6.1) [37];

$$P(\omega_k|x) = \frac{P(x|\omega_k) * P(\omega_k)}{P(x)} \tag{6.1}$$

Alternatively,

$$Posterior\ Probability = \frac{(Likelihood) * (Prior)}{(Marginalization)} \tag{6.2}$$

- ω_k: k defines the number of classes, in our case its binary (pedestrian, non-pedestrian).
- x: independent feature, aspect ratio of the pedestrian object.

Parameter estimation In this case the class prior is assumed to be equiprobable. To estimate the parameters for a features' distribution (aspect ratio distribution) fig.22, the assumed *event model* (i.e the assumptions on distributions of feature) is a simple *Gaussian naive Bayes*. The discrete Gaussian distribution is given by equation (6.3) [38],

$$P(x|\omega_k) = \frac{1}{\sqrt{2\pi\hat{\sigma_k^2}}} e^{-\frac{(x-\hat{\mu_k})^2}{2\hat{\sigma_k^2}}} \tag{6.3}$$

The training data distribution fig.22 is modelled to be Gaussian distribution. For each class, the estimated *mean* $\hat{\mu}_k$ and *variance* $\hat{\sigma}_k^2$ is computed using the *maximum likelihood estimator*(MLE) as given in the equation (3.42). The fig.23 shows the Gaussian distribution of the training data distribution, where the parameters of the Gaussian curve is estimated using MLE.

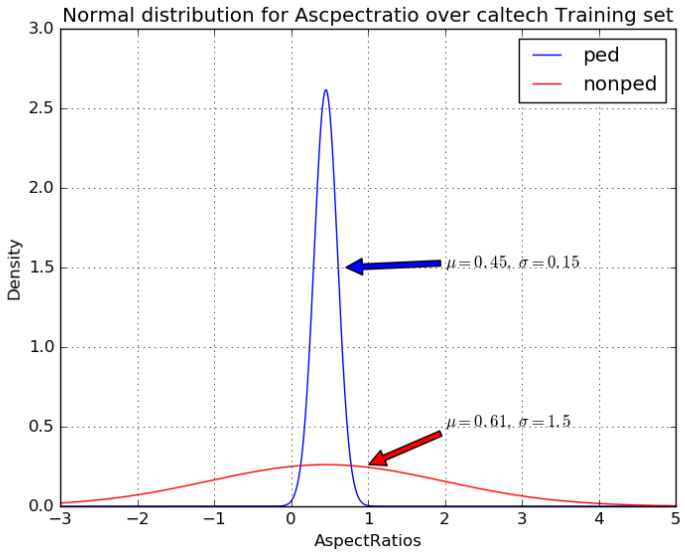


Figure 23: Gaussian distribution for aspect ratio values over Caltech pedestrian dataset
Note: The distribution is modelled using only the set05 data form Caltech pedestrian dataset.

The blue curve is the modelled distribution of aspect ratio over the set05 of Caltech pedestrian dataset. While estimating the event model parameter we just considered aspect ratio of pedestrian that are less the one. For non-pedestrian you see from the fig.23, the standard deviation is 10x greater than the standard deviation of pedestrian class. This allows to have a flat distribution for the non-pedestrian class. It can be stated that the contextual classifier ψ_{AR} is a univariate naive Bayesian classifier, where this classifier provides the class specific posterior probability (thresholding the resulted probability with confidence threshold gives the class decision δ_{AR} of the region). Knowing the parameters ($\hat{\mu}_k$ and $\hat{\sigma}_k$) of the distribution on training data, class specific likelihood $P(x|\omega_k)$ are computed by using equation (6.3).

During testing the class specific posterior probability is computed as follow.

- Class specific likelihood is calculated using equation (6.3). Where the aspect ratio value of the region is substituted along with learnt class specific parameters ($\hat{\mu}_k$ and $\hat{\sigma}_k$) in the equation (6.3).
- The class priors are assumed to be equiprobable.
- Posterior probability of the new aspect ratio value is given by substituting the computed class likelihood and marginalization factor in equation (6.1).

6.3 Calssifier's fusion

6.3.1 Fast-RCNN ($\psi_{FastRCNN}$) + Context based Classifier (ψ_{AR}) fusion

From the fig.13, the two classifiers $\psi_{FastRCNN}$ and ψ_{AR} are the dynamic classifier. Where classifier $\psi_{FastRCNN}$ that is explained in section 6.2.1 is a deep learning model and classifier ψ_{AR} is a univariate context-based classifier which is explained in section 6.2.2. Each of these classifiers accepts the regions proposed from *brute force* method explained in section 6.1.1 and attaches corresponding *pedestrian* and *non-pedestrian* scores for each region. Fast-RCNN results $\xi_{FastRCNN} = (\xi^{ped}_{FastRCNN}, \xi^{non-ped}_{FastRCNN})$ for each region, where $\xi^{ped}_{FastRCNN}$ is the pedestrian score for the region and $\xi^{non-ped}_{FastRCNN}$ is the score for non-pedestrian from the Fast RCNN deep architecture. Likewise, the univariate context-based classifier ψ_{AR} assigns score $\xi_{cb\_c} \approx \xi_{AR} = (\xi^{ped}_{AR}, \xi^{non-ped}_{AR})$ for each proposed regions (where ξ_{cb_c} is the score for the region based on classifier model with context information, her we use just aspect ratio (AR), so it scores based on AR value). Therefore each region has *four* scores $(\xi^{ped}_{FastRCNN}, \xi^{non-ped}_{FastRCNN}, \xi^{ped}_{AR}, \xi^{non-ped}_{AR})$ from dynamic classifiers ($\psi_{FastRCNN}$ and ψ_{AR}), the scores from these classifier can also be viewed as;

- $\xi^{ped}_{FastRCNN}$: probability/score of a region predicted to be *pedestrian* by Fast RCNN.
- $\xi^{non-ped}_{FastRCNN}$: probability/score of a region predicted to be *non-pedestrian* by Fast RCNN.
- ξ^{ped}_{AR}: probability/score of a region predicted to be *pedestrian* by context-based classifier (aspect ratio of the region).
- $\xi^{non-ped}_{AR}$: probability/score of a region predicted to be *non-pedestrian* by context-based classifier (aspect ratio of the region).

These obtained scores for each proposed regions of a image is fused inside an supervisory classifier ψ_H as shown in the fig.13. Supervisory classifier fuse the score provided from the dynamic classifiers for each proposed region and attaches a new fused score $\xi_H = (\xi^{ped}_H, \xi^{non-ped}_H)$ for each region. The fusion classifier is a Bayesian

classifier which governs on the principle of *Baye's Theorem*, the classification is based on the *posterior probability*. The fusion classifier is defined as given by equation (6.4),

$$P(\omega_k|\xi_{FastRCNN}, \xi_{AR}) = \frac{P(\xi_{FastRCNN}, \xi_{AR}|\omega_k) * P(\omega_k)}{P(\xi_{FastRCNN}, \xi_{AR})} \tag{6.4}$$

In the equation (6.4) class-conditional probability density function $P(\xi_{FastRCNN}, \xi_{AR}|\omega_k)$ is the *likelihood function*. The likelihood function is given as in the equation (6.5),

$$P(\xi_{FastRCNN}, \xi_{AR}|\omega_k) = P(\xi^{ped}_{FastRCNN:j}|\omega_k) * P(\xi^{non-ped}_{FastRCNN:j}|\omega_k) * P(\xi^{ped}_{AR:j}|\omega_k) * P(\xi^{non-ped}_{AR:j}|\omega_k) \tag{6.5}$$

where, j= ped, non-ped. As seen in the equation (6.5), likelihood function is the product of four univariate likelihood density function. Set-05 of Caltech pedestrian dataset is used to get these four density function (actually we get eight likelihood functions for both classes (k=ped,non-ped)), the event model of the density function is assumed to be Gaussian function described by equation (6.3) and the parameters of each univariate density function is estimated using MLE from the equation (3.42).

Figure 24 shows the estimated parameters of four likelihood density functions based upon scores given by Fast RCNN for each proposed ROI.

- In fig.24a, the blue curve defines the estimated parameter for event model, which is assumed to be Gaussian distribution of pedestrian scores ($\xi^{ped}_{FastRCNN:ped}$) from $\psi_{FastRCNN}$ for *true pedestrian/person regions* that are proposed on overall set-05 of Caltech pedestrian dataset. Whereas, the red curve is the estimated Gaussian parameter for non-pedestrian scores ($\xi^{ped}_{FastRCNN:non-ped}$) from $\psi_{FastRCNN}$ for true pedestrian region. The parameter are estimated using MLE method as explained in section 3.3. In the proposed ROI using brute force method the true pedestrian regions are captured using *Intersection over Union*(IoU) as given by the equation(5.1), the overlapping threshold (A_{match} in equation 5.1) was set to be 0.5, if the region overlaps with the pedestrian ground truth of the image greater than 50% then it is captured to be true pedestrian region. Also the true pedestrian regions consisted of pedestrian ground truths of the all the images in Set-05 of Caltech dataset. In fig.25a, the blue curve defines the estimated parameter of assumed Gaussian distribution of pedestrian scores $\xi^{ped}_{AR:ped}$ for true pedestrian region from context based classifier ψ_{AR} and red curve defines the estimated parameter of assumed Gaussian distribution of non-pedestrian scores $\xi^{ped}_{AR:non-ped}$ for true pedestrian region from context based classifier ψ_{AR}.

- In fig.24b, the blue curve defines the estimated parameter for event model, which is assumed to be Gaussian distribution of pedestrian scores ($\xi^{non-ped}_{FastRCNN:ped}$) from $\psi_{FastRCNN}$ for *true non-pedestrian/background regions* that are proposed on overall set-05 of Caltech pedestrian dataset. Whereas, the red curve is the estimated Gaussian parameter for non-pedestrian scores ($\xi^{non-ped}_{FastRCNN:non-ped}$)

from $\psi_{FastRCNN}$ for true non-pedestrian region. The parameter are estimated using MLE method as explained in section 3.3. In the proposed ROI using brute force method the true non-pedestrian regions are captured using *Intersection over Union*(IoU) as given by the equation (5.1), the overlapping threshold (A_{match} in equation 5.1) was set to be < 0.5, if the region overlaps with the pedestrian ground truth of the image lesser than 50% then it is captured to be true non-pedestrian region. In fig.25b, the blue curve defines the estimated parameter of assumed Gaussian distribution of pedestrian scores ($\xi^{non-ped}_{AR:ped}$) for true non-pedestrian region from context based classifier ψ_{AR} and red curve defines the estimated parameter of assumed Gaussian distribution of non-pedestrian scores ($\xi^{non-ped}_{AR:non-ped}$) for true non-pedestrian region from context based classifier ψ_{AR}.

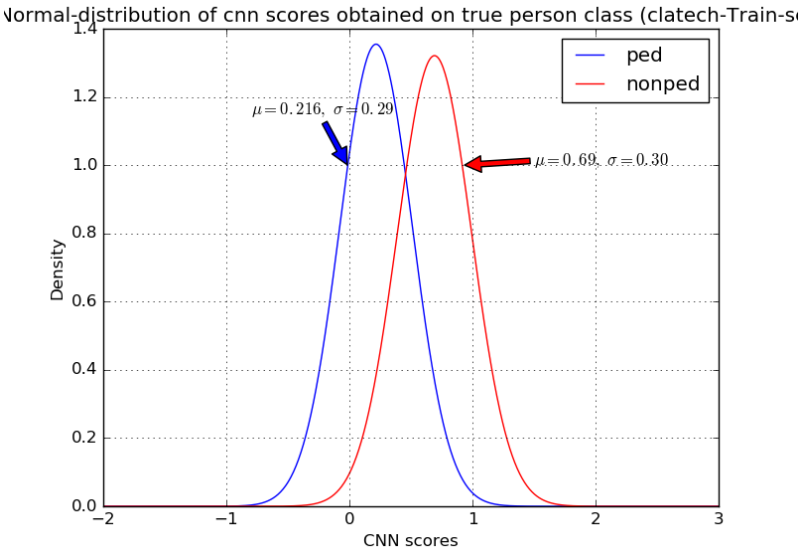
During testing the new posterior probability/fusion score ξ_H for the proposed region is computed as, computation of pedestrian fusion score ξ^{ped}_H;

- Each region has four scores ($\xi^{ped}_{FastRCNN}, \xi^{non-ped}_{FastRCNN}, \xi^{ped}_{AR}, \xi^{non-ped}_{AR}$) from the dynamic classifiers as seen in the fig.13
- The class priors $P(\omega_k)$ are assumed to be equiprobable.
- The fusion score for pedestrian is given by the equation (6.4). The likelihood term in the equation (6.4) is product of four univariate density functions as seen in equation (6.5). The likelhood term in equation (6.4) for finding posterior probability of funsion classifier for pedestrian class can be given as in equation (6.6).

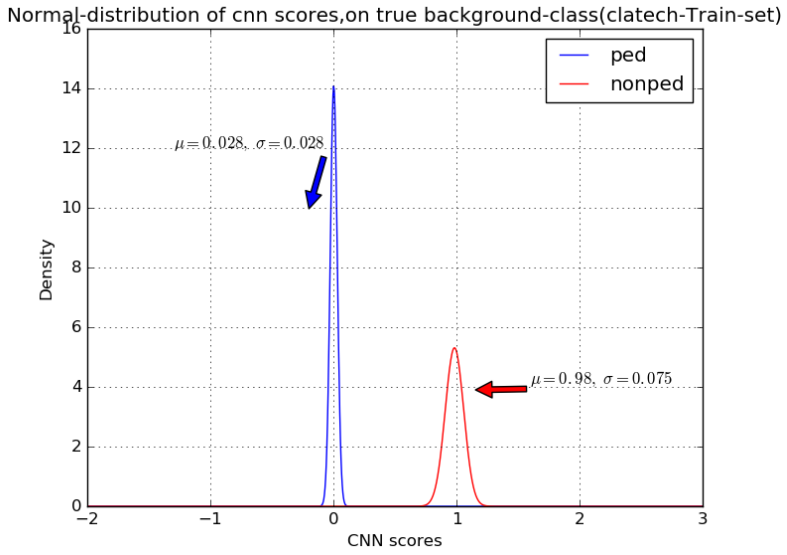
$$P(\xi_{FastRCNN}, \xi_{AR}|\omega_{ped}) = P(\xi^{ped}_{FastRCNN:ped}|\omega_{ped}) * P(\xi^{non-ped}_{FastRCNN:ped}|\omega_{ped}) * P(\xi^{ped}_{AR:ped}|\omega_{ped}) * P(\xi^{non-ped}_{AR:ped}|\omega_{ped}) \tag{6.6}$$

- $P(\xi^{ped}_{FastRCNN:ped}|\omega_{ped})$, term is computed using equation (6.3) by substituting $x = \xi^{ped}_{FastRCNN}$ (which is obtained from $\psi_{FastRCNN}$), and $\hat{\mu}_k$, $\hat{\sigma}^2_K$ which are estimated parameters for distribution of pedestrian scores for true pedestrian regions (blue curve in fig.24a). The same computation is followed to compute $P(\xi^{non-ped}_{FastRCNN:ped}|\omega_{ped})$, $P(\xi^{ped}_{AR:ped}|\omega_{ped})$, $P(\xi^{non-ped}_{AR:ped}|\omega_{ped})$ by subsituting specific scores and estimated parameters in (6.3). This enables to estimate the likelihood using equation (6.6).
- The posterior probability $P(\omega_{ped}/\xi_{FastRCNN}, \xi_{AR}) \approx \xi^{ped}_H$ of the region from the fusion classifier (ψ_H in fig.13) for pedestrian class is obtained from equation (6.4) by substituting likelihood $P(\xi_{FastRCNN}, \xi_{AR}/\omega_{ped})$, assumed class priors $P(\omega_k)$, and computed marginalization factor $P(\xi_{FastRCNN}, \xi_{AR})$.
- With the known posterior probability for pedestrian class the posterior probability of non-pedestrian is given as below,

$$\xi^{non-ped}_H = P(\omega_{non-ped}/\xi_{FastRCNN}, \xi_{AR}) = 1 - P(\omega_{ped}/\xi_{FastRCNN}, \xi_{AR})$$

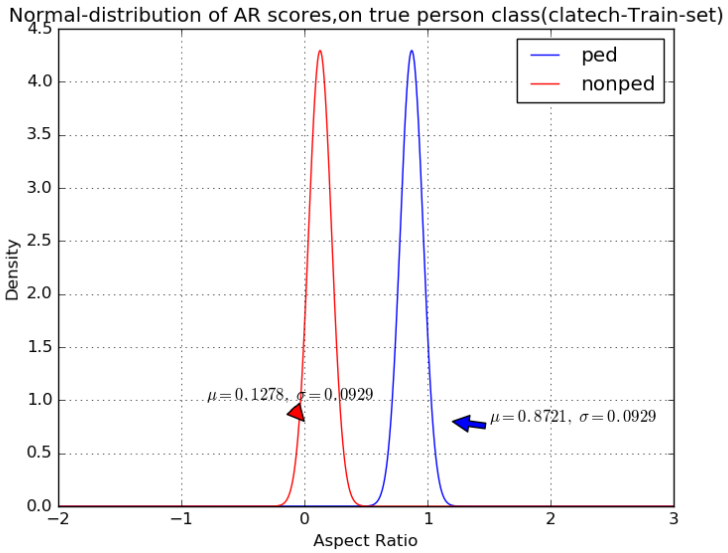


(a) Gaussian estimated parameter on scores ($\xi^{ped}_{FastRCNN:ped}$, $\xi^{ped}_{FastRCNN:ped}$) of true pedestrian/person region given by Fast RCNN $\psi_{FastRCNN}$

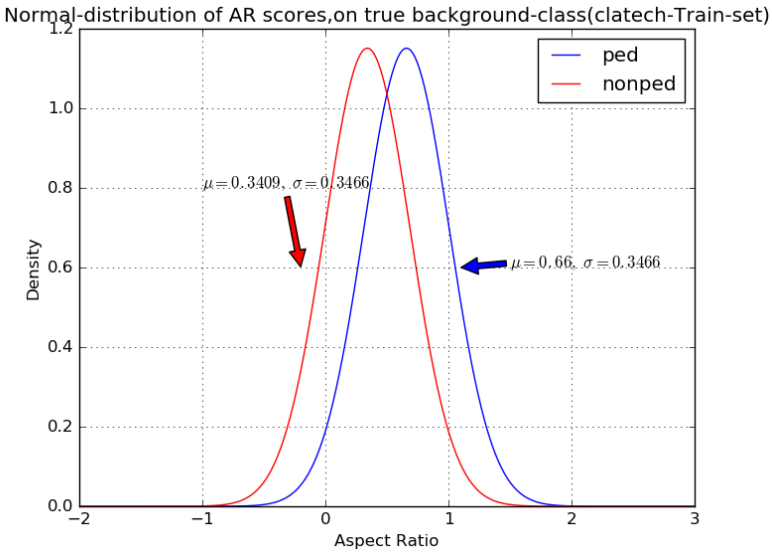


(b) Gaussian estimated parameter on scores ($\xi^{non-ped}_{FastRCNN:ped}$, $\xi^{non-ped}_{FastRCNN:non-ped}$) of true non-pedestrian/background region given by Fast RCNN $\psi_{FastRCNN}$

Figure 24: Gaussian distribution of scores from Fast RCNN $\psi_{FastRCNN}$



(a) Gaussian estimated parameter on scores ($\xi^{non-ped}_{AR:ped}$, $\xi^{non-ped}_{AR:non-ped}$) of true pedestrian/person region given by Fast RCNN ψ_{AR}



(b) Gaussian estimated parameter on scores ($\xi^{non-ped}_{AR:ped}$, $\xi^{non-ped}_{AR:non-ped}$) of true non-pedestrian/background region given by Fast RCNN ψ_{AR}

Figure 25: Gaussian distribution of scores from context-based classifier ψ_{AR}

6.3.2 SVM-HOG ($\psi_{SVM-HOG}$) + Aspect ratio classifier (ψ_{AR}) fusion

From the figure 15, the classifiers $\psi_{SVM-HOG}$ and ψ_{AR} are the dynamic classifiers. These two classifiers are shallow method, where in each classifier is responsible for mapping encoded object features which are in Π space to appropriate class label ω which is a discrete space Δ. As noticed from the fig.15, the classifier $\psi_{SVM-HOG}$ is a linear SVM classifier that acts on the encoded HOG descriptor of the object for classifying the detection window from *feature space* Π to discrete decision space Δ. Thereby for all the regions proposed for HOG-SVM detector pipeline are given decisions $\delta_{SVM-HOG}$ that defines for which class label ω_k does the region is associated to. Here, k is binary (pedestrian, non-pedestrian). The context-based classifier ψ_{AR} is the simple *naive Bayesian classifier* that depend on the univariate density function of the context feature; the context information used are the aspect ratio of the pedestrian object from Caltech pedestrian dataset (only set 05). Hence, for each region proposed the context-based classifier provides posterior decision ω_k, which is class label in discrete space (Bayesian classifier is responsible for posterior probability, hence thresholding the probability with specific confidence value will provide decision of the region with respect to decision space Δ). Therefore, these two dynamic classifiers can be given as $\psi : \Pi \rightarrow \Delta$. For implementation details of this two classifiers author points to the sections 6.2.3 and 6.2.2 of this chapter. The obtained decisions for each proposed region from classifiers $\psi_{SVM-HOG}$ and ψ_{AR} are used to fuse classifier fig.15, whereas in section 6.3 scores from the dynamic classifiers were used in fusing classifier. The fusion classifier ψ_H in fig.15 is modelled on the marginal decisions $(\delta_{SVM-HOG}, \delta_{AR})$ obtained for each region proposals, for modelling the fusion classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$, *set 05* of the Caltech pedestrian dataset is used. The fusion classifier/supervisory classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$ is a *naive Bayesian classifier*. This supervisory classifier make decisions on the marginal decisions of the dynamic classifiers rather than on the joint probability densities of the feature space. The method proposed in [45] is followed for fusing the decision from the dynamic classifiers. The usual Bayesian bivariate classifier ψ_b is formed from the bivariate class-conditional probability density function $P(X_{HOG}, X_{AR}|\omega_k)$ and the prior probability $P(\omega_k)$ using equation (6.1), therefore it requires to estimate the bivariate density functions $P(X_{HOG}, X_{AR}|\omega_k)$ which are to be known beforehand. The need to estimate the bivariate density functions are eliminated by just considering the decisions $\delta_{SVM-HOG}$, δ_{AR} from the dynamic classifiers $\psi_{SVM-HOG}$, ψ_{AR}. The supervisory Bayesian classifier is given as in the equation (6.7),

$$\begin{aligned} \psi_H(\delta_{SVM-HOG}, \delta_{AR}) &\approx P(\omega_k|\delta^k_{SVM-HOG}, \delta^k_{AR}) \\ &= \frac{P(\delta^k_{SVM-HOG}, \delta^k_{AR}|\omega_k) * P(\omega_k)}{P(\delta_{SVM-HOG}, \delta_{AR})} \end{aligned} \tag{6.7}$$

where, k=binary (pedestrian, non-pedestrian). The likelihood or bivariate class-conditional probability $P(\delta^k_{SVM-HOG}, \delta^k_{AR}|\omega_k)$ is estimated on basis of likelihood value learnt on true class regions being given decision as pedestrian and non-pedestrian on training dataset. In the same way likelihood value for true class boxes from

Table 5: Eight likelihood values computed on decisions made from the dynamic classifiers $\psi_{SVM-HOG}$, ψ_{AR}.

SVM-HOG detector $(\psi_{SVM-HOG}(X_{HOG}) \rightarrow \delta_{SVM-HOG})$	**Context-based classifier** $(\psi_{AR}(X_{AR}) \rightarrow \delta_{AR})$
$\delta^{ped}_{SVM-HOG:ped} = \frac{True-Positive}{TotalPositives} = 0.0698$	$\delta^{ped}_{AR:ped} = \frac{True-Positive}{TotalPositives} = 0.8339$
$\delta^{ped}_{SVM-HOG:non-ped} = \frac{False-Negatives}{TotalPositives} = 0.93017$	$\delta^{ped}_{AR:non-ped} = \frac{False-Negatives}{TotalPositives} = 0.1760$
$\delta^{non-ped}_{SVM-HOG:ped} = \frac{False-Positives}{TotalNegatives} = 0.0004659$	$\delta^{non-ped}_{AR:ped} = \frac{False-Positives}{TotalNegatives} = 0.3758$
$\delta^{non-ped}_{SVM-HOG:non-ped} = \frac{True-Negatives}{TotalNegatives} = 0.9995$	$\delta^{non-ped}_{AR:non-ped} = \frac{True-Negatives}{TotalNegatives} = 0.6341$

Note: The likelihood values are computed on set 05 of Caltech pedestrian dataset which is consider to be training data in our case.

context-based classifier ψ_{AR} is learnt on the training Set 05 of Caltech pedestrian dataset. This accounts for eight likelihood values. the true pedestrian and non-pedestrian regions are captured as discussed in the section 6.3.

- $\delta^{ped}_{SVM-HOG:ped}$: The upper subscripts define the true class of proposed region in the image, lower subscript *SVM-HOG:ped* defines decision of regions being pedestrian from SVM-HOG detector pipeline.
- $\delta^{ped}_{SVM-HOG:non-ped}$: The upper subscripts define the true class of proposed region in the image, lower subscript *SVM-HOG:non-ped* defines decision of regions being non-pedestrian from SVM-HOG detector pipeline $\psi_{SVM-HOG}$ in figure 15.
- $\delta^{non-ped}_{SVM-HOG:ped}$: The upper subscripts define the true class of proposed region in the image, lower subscript *SVM-HOG:ped* defines decision of regions being pedestrian from SVM-HOG detector pipeline.
- $\delta^{non-ped}_{SVM-HOG:non-ped}$: The upper subscripts define the true class of proposed region in the image, lower subscript *SVM-HOG:non-ped* defines decision of regions being non-pedestrian from SVM-HOG detector pipeline.
- Same as the above we get the likelihood value from the context-based classifier; $\delta^{ped}_{AR:ped}$, $\delta^{non-ped}_{AR:ped}$, $\delta^{ped}_{AR:non-ped}$, $\delta^{non-ped}_{AR:non-ped}$. During training eight decision likelihood from two dynamic classifier are learnt. The table 5 gives information of computed learnt likelihoods values over training set.

Table 6: Class priors obtained on the set 05 of Caltech pedestrian data set

Class Priors	
$P(\omega_{ped})$	0.0588
$P(\omega_{non-ped})$	0.9411

Computation of fusion decision δ_H from the fusion classifier ψ_H during testing; each proposed region gets the decision from the dynamic classifiers $\psi_{SVM-HOG}$ and

ψ_{AR}. For sake of understanding let's assume that for a single region of interest ϕ_i, the decisions from the dynamic classifiers $\psi_{SVM-HOG}$ and ψ_{AR} are $\delta_{SVM-HOG:non-ped}$ and $\delta_{AR:ped}$ respectively. The decision from the SVM-HOG detector is defining that the region is an non-pedestrian and that of the context-based classifier decision based upon its aspect ratio value has given the decision as pedestrian. To get the new fusion decision from the fusion classifier that govern on these obtained decisions is given by the equation (6.7). In this case, equation (6.7) can also be rewritten as follows,

$$P(\omega_k|\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}) = \frac{P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_{ped}) * P(\omega_{ped})}{P(\delta_{SVM-HOG}, \delta_{AR})} \tag{6.8}$$

The *likelihood function* $P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_k)$ in equation (6.8) is not computed on basis of the joint probability density function, but with the help of the likelihood values learnt during modelling the fusion classifier. Hence in this case the likelihood function is given by as follows;

$$P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_{ped}) = \delta^{ped}_{SVM-HOG:non-ped} * \delta^{ped}_{AR:ped} \tag{6.9}$$

form the table 5 substitute corresponding likelihood values in above equation the prior $P(\omega_{ped})$ from the equation (6.8) is given from the table 6. With the computed likelihood in equation (6.9), pedestrian class prior and marginalization factor in (6.8) we arrive with the decision score of the region for pedestrian class. It is given as,

$$P(\omega_{ped}|\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}) = \frac{P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_{ped}) * P(\omega_{ped})}{P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_{ped}) * P(\omega_{ped}) + P(\delta_{SVM-HOG:non-ped}, \delta_{AR:ped}|\omega_{non-ped}) * P(\omega_{non-ped})}$$

$$P(\delta_{SVM-HOG:ped}, \delta_{AR:non-ped}|\omega_{ped}) \approx \delta_{H:ped}$$

For the same considered region decision for non-pedestrian is given as;

$$\delta_{H:non-ped} = 1 - \delta_{H:ped}$$

Chapter 7

Results and Discussion

7.1 Deep learning

As described in the section 4.1, the deep learning detector Fast-RCNN $\psi_{Fast-RCNN}$ is fused at the post processing stage with the context-based classifier ψ_{AR}. Initially it is important to notice the effect of fusion method explained in the section 6.3.1, the obtained results shown in the fig.26 gives evidence of the fusion methodology. The fig.26a and fig.26d shows the detection score provided by the Fast-RCNN deep detector, the fig.26b and fig.26e are the score for each bounding box obtained from context-based classifier, and the fig.26c and fig.26f are the score given from the fusion classifier ψ_H. Pedestrian appearing at the far region are missed by many detectors because of several factors that govern the detectors. As seen from the fig.26, consider the score given by the Fast-RCNN classifier (fig.26d), it is noticed that the score given to the region belonging to pedestrian class is significantly very low (0.055). This shows the necessity of additional object cues to improve the detection ability. The required additional information is provided by the context-based classifier ψ_{AR}, the context information used is the aspect ratio (AR) of the pedestrian. From the fig.26e you notice that the context-based classifier score for the proposed region is having high score (0.849). Section 6.3.1 explains the fusion method that governs on the scores obtained from the Fast-RCNN classifier and the context-based classifier, as a result of fusing the scores it is seen from the fig.26f the increase in the confidence (0.698) of the region belonging to pedestrian category.

The region hypothesis are generated based on the brute force method that is explained in the section 6.1.1. For each image approximately $\approx 2000k$ region hypothesis were proposed, large number of proposed hypothesis have the aspect-ratio size that of an pedestrian object this can be visualised fig.17b. The context-based classifier which governs on AR value of the proposed region hypothesis, has many false positives resulting in very low precision. It is clearly seen from the fig.27, the fourth and fifth row corresponds to the detection given by the context-based (aspect ratio classifier) at two confidence threshold 0.3 and 0.5 respectively. When fusion of Fast-RCNN and context-based classifier score is performed, it is seen along with increase in the detection of pedestrian the additional false positives. Consider

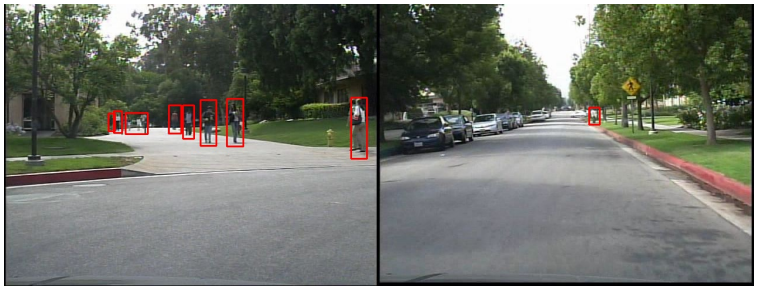
(a) (b) (c)

(d) (e) (f)

Figure 26: Visualizing the effect of fusion.
Note: For the sake of understanding the effect of fusion, we just consider ground truth as the proposed objects for the detector 4.1.

the first image column in the fig.27, at the confidence threshold 0.3 the deep network Fast-RCNN detects very few pedestrians. As a result of fusing with context based classifier we notice that the detected pedestrian score have significantly increased and even it is capable of detecting far region pedestrian which has coarse spatial information (see last two rows of the first image column in fig.27). Along with increased number of true class detection we notice the increased number of false positive that is induced by the consider context-classifier which governs on the aspect-ratio of the proposed regions.

Test Image with GT plotted



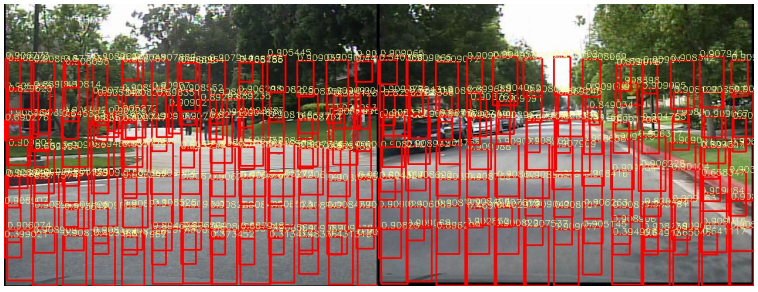
Fast-RCNN (NMS:0.3, Confidence_threshold:0.3)

no detection at confidence threshold 0.3

Fast-RCNN (NMS:0.3, Confidence_threshold:0.5)

no detection at confidence threshold 0.5

Aspect-Ratio classifier (NMS:0.3, Confidence_threshold:0.3)



Aspect-Ratio classifier (NMS:0.3, Confidence_threshold:0.5)

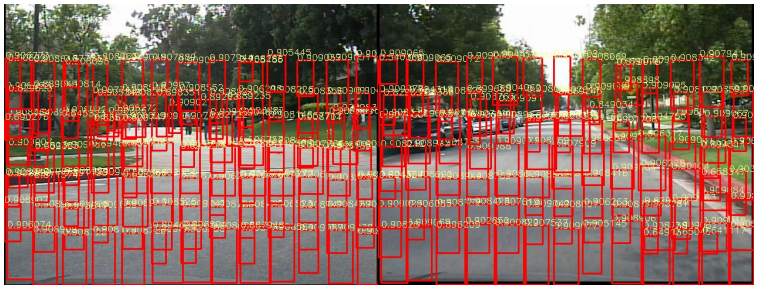
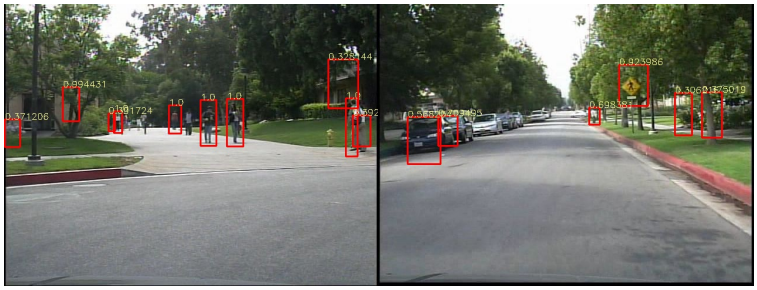


Figure 27: continued....

Fusion Classifier (Fast-RCNN+AR) (NMS:0.3, Confidence_threshold:0.3)



Fusion Classifier (Fast-RCNN+AR) (NMS:0.3, Confidence_threshold:0.5)

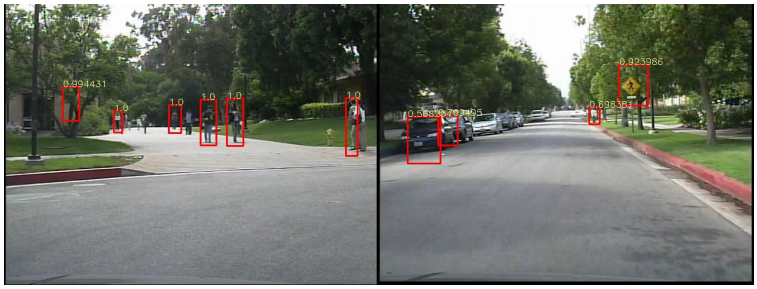


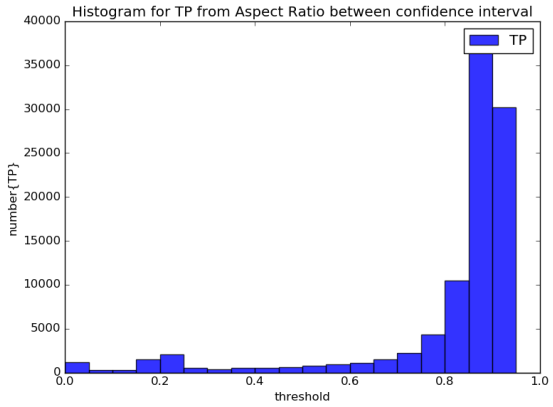
Figure 27: Overview of detection from Fast-RCNN, Context-based(aspect-ratio) classifier and Fusion classifier at different

The classifiers Fast-RCNN $\psi_{Fast-RCNN}$, context-based classifier ψ_{AR} and the fusion classifier ψ_H are evaluated on *set 09* of Caltech pedestrian data [1]. The total number of region hypothesis proposed for each classifier are equivalent. The Receiver Operator Characteristic (ROC) curve fig.32a provides the information of *true positive rate* (TPR) versus *false positive rate or 1-specificity* (FPR) at different confidence threshold, also it provides information regarding how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. It is noticed from the equation (5.4) that the TPR is the ratio of true positives over the total provided positives and FPR equation (5.6) is the ratio of false positives over total number of negatives. For all the considered classifiers at very low confidence threshold the point in the ROC space is at the top extreme right corner stating with high TPR and FPR. With a slight increase in the confidence threshold (increase from 0.0 to 0.1) the TPR of a Fast-RCNN drastically decreases to approximately 0.4/40% fig.32a, the reason behind this is that Fast-RCNN score's the proposed regions with very least scores as noticed in the fig.26d, also the FPR is very low at this TPR position because as increase in the confidence threshold the number of false positives are very less. Many of the true pedestrian region in the set 09 of Caltech dataset are occurring at medium and far region that are very challenging for the detectors. From fig.28c it is noticed that the scores given by the Fast-RCNN classifier $\psi_{Fast-RCNN}$ for true class regions are more concentrated with low scores, at very low confidence threshold many of the true class regions are grouped into true positives hence having very low number of true negatives this give rise to high TPR score initially. As the confidence increases slightly many of the regions with less score than the confidence threshold fall to false negatives, hence leading with low TPR. This is the core reason for having a steep decrease of the Fast-RCNN ROC curve

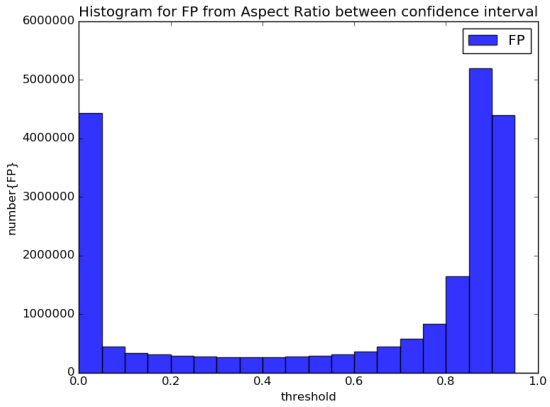
in the fig.32a. The context-based classifier ψ_{AR} which governs on the aspect ration value of the proposed regions for classifying the regions to a specific class. As the region hypothesis is proposed using the brute-force method many of the proposed regions are having the aspect ratio size that of the true pedestrian region, hence this result in many false positives even as the threshold is increased it can be seen in the fig.28b. From the fig.32a the aspect ratio classifiers' ROC curve shows high FPR even at the high confidence threshold. When the both classifiers $\psi_{Fast-RCNN}$ and ψ_{AR} are fused the true pedestrian regions are given high score and even much of the false positives are induced by the ψ_{AR} classifier. This can be seen in the fig.28e and fig.28e respectively. Clearly from the ROC plot fig.32a it is seen that the fusion classifier (blue curve) has dominance over the $\psi_{Fast-RCNN}$ and ψ_{AR} classifiers.

Precision-Recall (PR) curves are alternative to the ROC curves that give better metric for task with large skew in datasets like Caltech pedestrian dataset. An important difference between the ROC space and PR space is the visual representation of the curves. The goal of the PR space is to be in the upper-right-hand corner, and the PR curves in the fig.32b shows that there is still vast room for improvement. Precision is given as in the equation (5.3) which measures that fraction of examples classified as positives that are truly positive, whereas the recall is same as TPR. In other sense recall can be interpreted as the amount of region hypothesis that cover maximum true class regions at given threshold confidence of the classifier. False Positives FP weights the precision of the classifier, it is noticed from the fig.32b the PR curve of the context-based classifier ψ_{AR} almost runs parallel to the Recall axis having very low precision. From the fig.28b, it is seen even at thresholds grater than 0.6 the aspect ration classifier ψ_{AR} contributes large number of FPs, hence this contribution of FPs results in very low Precision even as the confidence threshold increases. Whereas, the Fast-RCNN $\psi_{Fast-RCNN}$ PR curve shows that their is vast space for improving it. Hence, fusing the Fast-RCNN with context-based classifier is tried to compensate for that achievable PR space. As it is noticed from the Fig.32b, classifiers fusion PR curve (blue curve) is shows dominance as the confidence threshold is increased, during very low confidence threshold the curve is dominated by the Fast-RCNN PR curve. It is because at very low thresholds (0.0 to 0.4) the number of false positives are high (seen in the fig.28f) resulting in low precision compared to Fast-RCNN, over increasing the threshold further the PR curve of fusion classifier starts dominating. From the ROC space fig.32a and PR space fig.32b it is observed that the fusion classifer ψ_H dominates, but from both the spaces it is noticed the fusion classifier has vast room for improvement. This improvement can be achieved buy fusing the classifier with additional object cues/information.

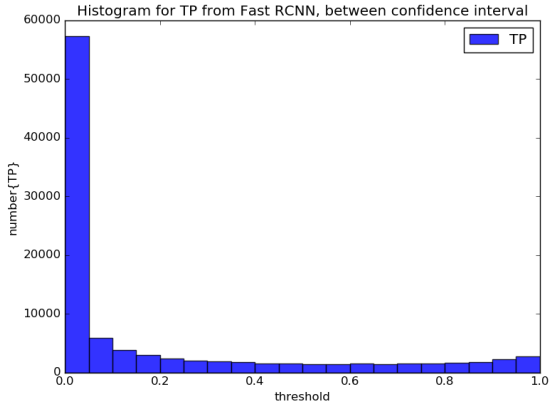
Miss-rate versus false positive per image (MR vs FPPI) log plot are important especially in the automotive environment. In the automotive environment there is a space for acceptable amount of FP's per image. Therefore many of the evaluation of object detectors designed for automotive area make use of MR vs FPPI as a main metric tool. Fig.32c shows the obtained log curves for $\psi_{Fast-RCNN}$, ψ_{AR} and fusion



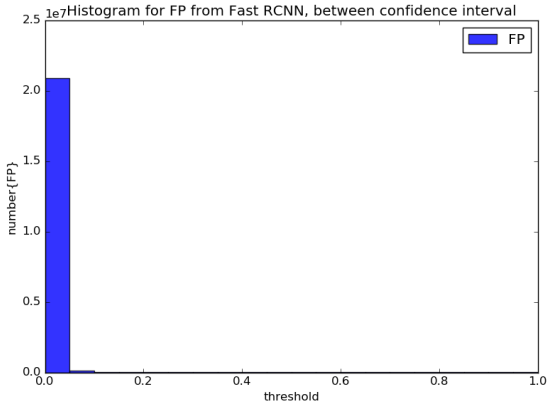
(a) Change in TP with increase in the confidence-threshold of context-based classifier



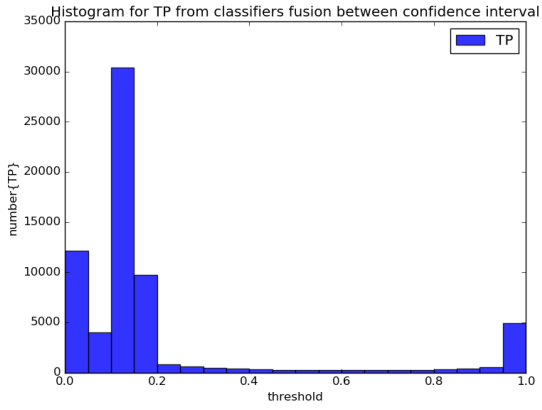
(b) Change in FP with increase in the confidence-threshold of context-based classifier



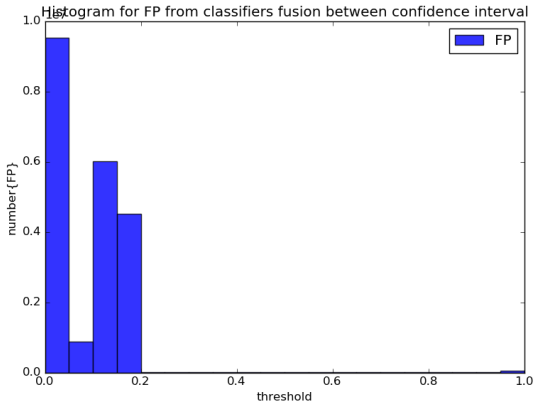
(c) Change in TP with increase in the confidence-threshold of Fast RCNN



(d) Change in FP with increase in the confidence-threshold of Fast RCNN



(e) Change in TP with increase in the confidence-threshold of Fusion classifier (Fast-RCNN + Context-based)



(f) Change in FP with increase in the confidence-threshold of Fusion classifier (Fast-RCNN + Context-based)

Figure 28: Histogram of true positives and false positives over test dataset

classifier ψ_H, observed from the fig.32c all the curves converge to a single point on x-axis because all the classifier receive same number of region proposal per image for classification. From the fig.32c it is noted that the fusion classifier curve (red) at the higher classification thresholds dominates the Fast-RCNN curve (blue). As the confidence threshold decrease the miss-rate remains almost equal, but the Fusion classifier has many FP's which are induced by the context-based (aspect ratio) classifier. All the metric curves shown in the fig.32 explains the effect of fusion classifier in the positive tone. A implemented context-based classifier ψ_{AR} is modelled using just the aspect ratio value. To have an better context-based classifier features like spatio-temporal information, semantic information along with scale features can be used for modelling.

As it is explained form the section 6.3 the event model considered for learning scores from Fast-RCNN $\psi_{Fast-RCNN}$ and context-based ψ_{AR} to model the fusion classifier ψ_H is the univariate Gaussian model. The distribution of scores for pedestrian (TP's) and non-pedestrian regions (FP's) from fig.33 is modelled using assumption as normal distribution, using an *Gaussian mixture model* on the obtained distribution will further eliminate the FP's and even a much sophisticated fusion classifier can be designed. Using Gaussian mixture model will increase the design complexity. In the next section, the outlook work shows how the detected FPs can be tackled using the stereo information. The Caltech dataset does not contain the stereo information, hence DLR data is used. There is no evaluation results for this outlook work as DLR data does not have enough annotation. It is just to give an outlook about how stereo information can be helpful to eliminate the detected false positives.

False positive elimination using stereo data

With the help of stereo image pair an dense stereo disparity image can be constructed. The disparity map is obtained using the *Triangulation* [27] principle. To show effect of adapting stereo-information an pre-computed dense disparity image of a corresponding image pair are utilized. The fig.29a is the detections obtained at the point in the detection pipeline showed in the fig.14, the detections are performed by fusion classifier ψ_F, the fusion classifier ψ_F is equivalent to the one explained earlier ψ_H in section 6.3.1, which make use of deep network Fast-RCNN scores and context-based scores. Due to fusion, along with high TP's we have high FP's fig.29a. These FP's can be eliminated using stereo information, fig.29b is the corresponding pre-computed disparity image. With the help of this disparity image the average disparity Z_{avg} for each detected region is computed, and it ais used to compute the distance (meters) to the region from camera center. Distance is computed using the equation (7.1) [27], where f is the *focal length* and b is the *Stereobasis*. The tabel 7 gives the stereo camera parameters that is used in to acquire DLR dataset. With the know distance and disparity values for each of the detected regions, the ground plane is estimated as shown in the fig.30a (green plane). The detected region bottom right (br) points are perpendicularly projected onto the ground plane and the height (meters) between the ground plane and the detected regions are computed

(The yellow line in fig.30a shows the projection of br on to the ground plane). Detected regions with greater than height-threshold (TH_H) are eliminated, in the work the threshold (TH_H) is decided based on the rough estimation and it is fixed to be $0.5m$. After height thresholding it is noticed that many false positives are eliminated compare fig.29a and fig.30b. Knowing the distance and disparity information of each detected region, the detected regions real height and width in meters are computed using the equations (7.3) and (7.2) [27], where W_{roi} and H_{roi} are real object width and height in meters, w and h are regions width and height in pixels. Whereas, d and f are distance to region from center of the camera and focal length respectively. Upon obtaining the real objects width and height in meters they are subjected to thresholding based on preselected height and width thresholds TH_h and TH_w.

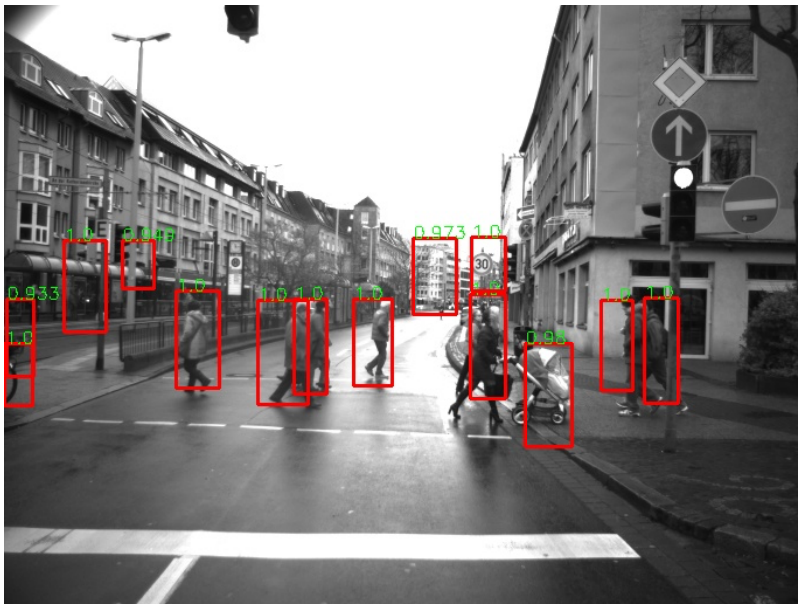
$$d = \frac{f * b}{Z_{avg}} \tag{7.1}$$

$$W_{roi} = \frac{w * d}{f} \tag{7.2}$$

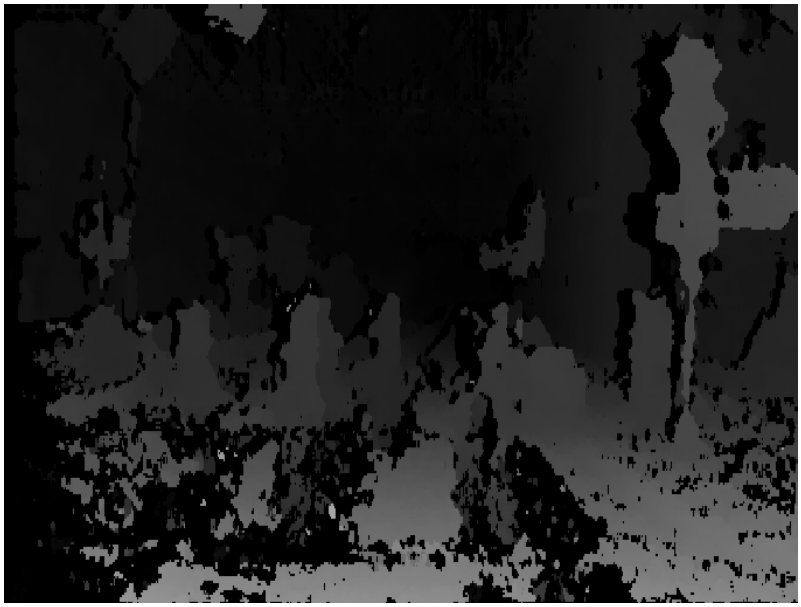
$$H_{roi} = \frac{h * d}{f} \tag{7.3}$$

Table 7: Camera intrinsic parameter of DLR dataset

Focal_length (f)	640.9625
U0	342.81
V0	255.58
Stereo_Basis (b)	0.9182470065



(a) Fusion classifier detection over DLR dataset image at confidence threshold 0.7 with NMS



(b) corresponding pre-computed dense disparity image

Figure 29: Fusion based detection on sample DLR image
Note: Image sample are from the DLR dataset.

(a) Ground plane estimation using stereo information

(b) Projection of bottom co-ordinates of detected boxes on to the estimated ground plane

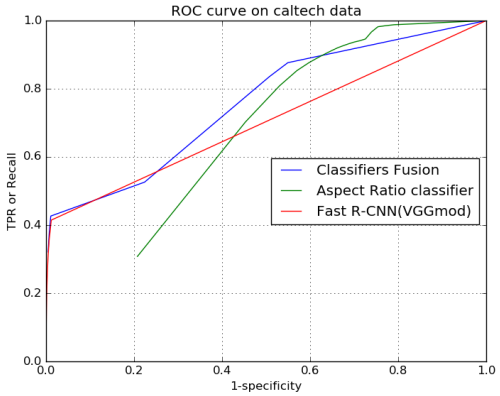
(a) Further eliminating FP based on detected boxes height and width

Figure 31: Elemination of false positives obtained from fusion classifier using stereo information

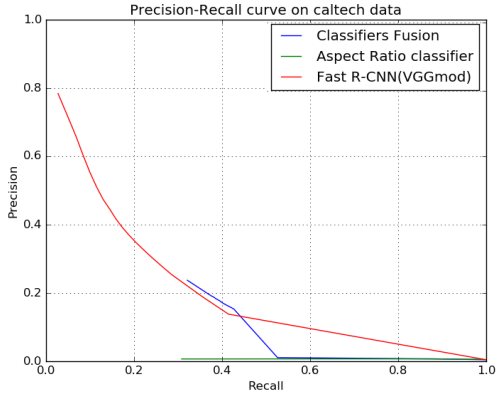
Note: Image sample are from the DLR dataset.

7.2 Shallow approach

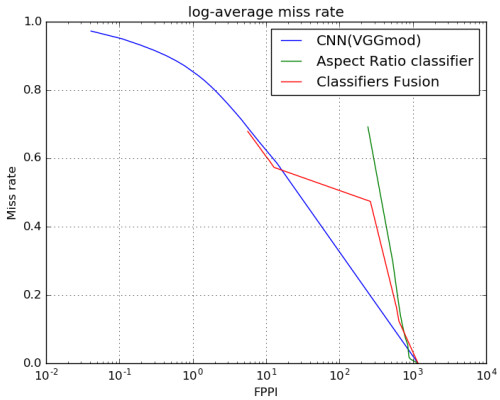
The shallow approach design as explained in the section 4.2 is interpreted in this section. As observed from the decision likelihood table 5, the decision given to positive (pedestrian) ROI from SVM-HOG $\psi_{SVM-HOG}$ to be pedestrian is very much less likely ($\delta^{ped}_{SVM-HOG:ped} = 0.0698$), and even the decision given to negative (non-ped) ROI by the $\psi_{SVM-HOG}$ is also very less likely ($\delta^{non-ped}_{SVM-HOG:ped} = 0.0004659$). From these likelihood value it can be interpreted that the $\psi_{SVM-HOG}$ classifier considers for very low number of TPR and very less FPR on Caltech pedestrian data set. The core reason behind very low TPR and FPR by the $\psi_{SVM-HOG}$, is that the SVM is trained on Daimler detection dataset [26], and this data set is complete different to that of Caltech dataset. On the other hand it is observed that the classifier ψ_{AR} has the high likelihood for ROI being pedestrian ($\delta^{ped}_{AR:ped} = 0.8339$, $\delta^{non-ped}_{AR:ped} = 0.3758$), this allow ψ_{AR} during decision fusion using ψ_H to have dominance over decisions given by the $\psi_{SVM-HOG}$ for ROI. Hence this results in high FPR and Recall, it is visualized from the fig.34. As the design is governed on decisions, only a single precision, recall, TPR and FPR value are obtained. Therefore it is not possible to have a PR and ROC space curves by varying threshold. In the next section we compare these values with the designed deep approach.



(a) Receiver Operator Characteristic (ROC) curve

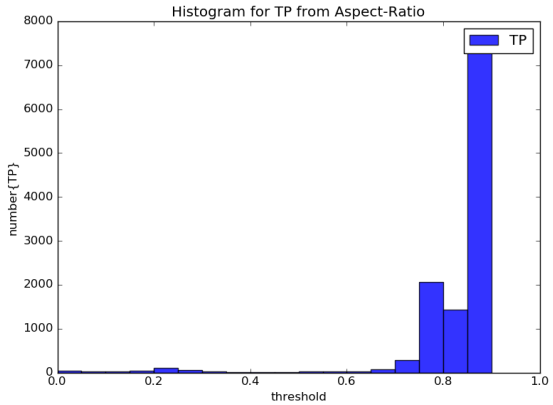


(b) Precision-Recall (PR) curve

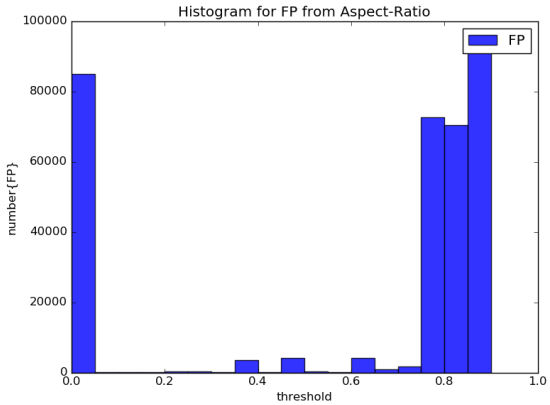


(c) MissRate(MR) vs False positive per image(FPPI) curve

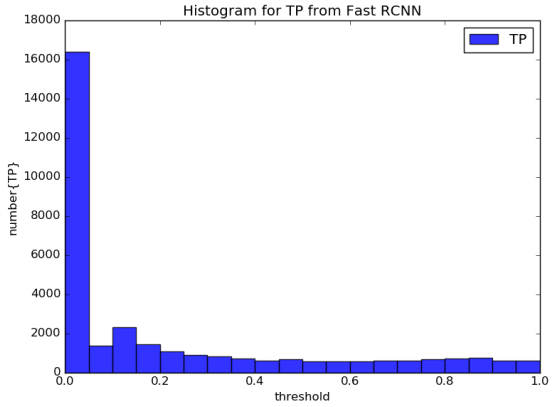
Figure 32: Metric curve's obtained by evaluating designed deep learning method
Note: Evaluation on Caltech pedestrian dataset considering all visible image regions.



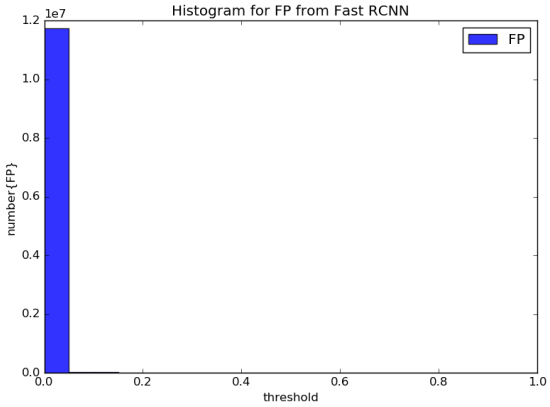
(a) Change in TP with increase in the confidence-threshold of context-based classifier



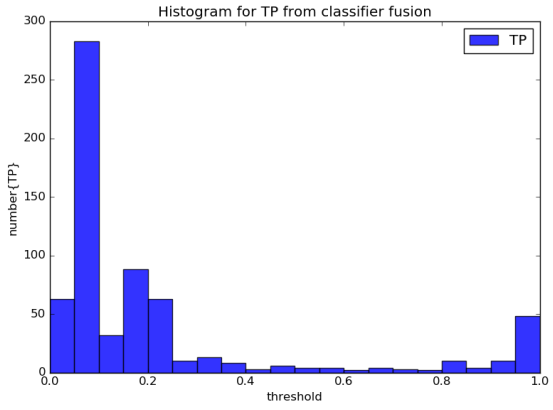
(b) Change in FP with increase in the confidence-threshold of context-based classifier



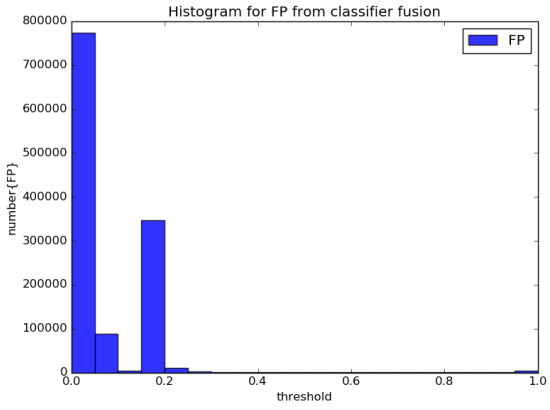
(c) Change in TP with increase in the confidence-threshold of Fast RCNN



(d) Change in FP with increase in the confidence-threshold of Fast RCNN



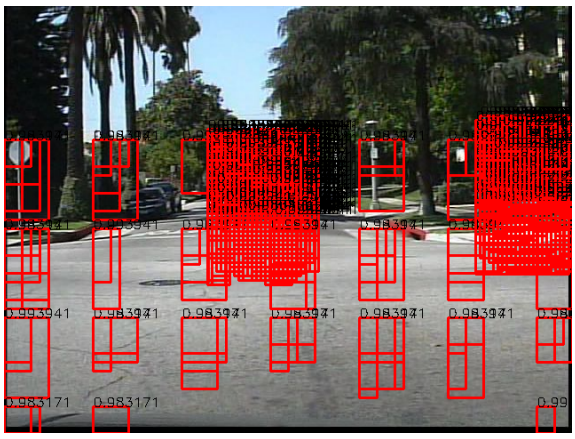
(e) Change in TP with increase in the confidence-threshold of Fusion classifier (Fast-RCNN + Context-based)



(f) Change in FP with increase in the confidence-threshold of Fusion classifier (Fast-RCNN + Context-based)

Figure 33: Histogram of true positives and false positives over training dataset

(a) detection by SVM-HOG classifier $\psi_{SVM-HOG}$



(b) Detection by decision fusion ψ_H classifier

Figure 34: Sample Detection image from $\psi_{SVM-HOG}$ and decision fusion classifier ψ_H

7.3 Comparison of deep and shallow approach

In this section comparison of different modelled classifiers that are employed in the design procedure is done. Table 9 provides the information regarding the best precision and recall of classifier which are obtained by finding least *euclidean distance* to the point $(1, 1)$ from the point in PR-space. F_1 scores as given by equation (7.4) [42] provides the harmonic mean of obtained precision and recall of each classifier (table 9). F_1 score gives rank and single measure of classification procedure's usefulness, therefore enabling one way to compare classifiers. According to the obtained F_1 scores from table 8, the score of proposed classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$ is ranked high compared to other classifiers. Whereas the F_1 scores of other classifiers are too low because, the F_1 score is balanced score provided on basis of precision and recall. Therefore those classifiers with low scores compare to $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$ as either very low precision or recall on considered Caltech dataset. Other metrics used to compare classifier performance are Area Under ROC Curve (AUC-ROC) and Aver-

age Precision (AP). Area under a PR curve gives rise to AP which allows to compare performance of classifiers. The PR curves also highlights the performance difference that are lost in ROC space [46, 47]. Table 10 shows the AUC-ROC and AP_{ped} for the different classifier. The AUC-ROC of $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$ is high compared to other considered classifiers, but the AP_{ped} is less than that of Fast-RCNN. This is due to the high number of FPR from the fused context-based classifier ψ_{AR}. As it is evident form table 10, the proposed fused classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$ dominates in ROC space and further modelling the context-based classifier with sophisticated information relative to object will improve the AP of the classifier. The high degree of skewness in the Caltech dataset should be consider, this skewness allows the considered and designed classifiers to have low values (table 8 and 10).

$$F_1 score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7.4}$$

Table 8: Comparison of F_1 scores on basis of PR-curve for different classifiers models used in the designed procedure

Classifier	$F_1 Score$
Context-based classifier ψ_{AR}	0.01172
Fast-RCNN $\psi_{Fast-RCNN}$	0.0916
SVM-HOG $\psi_{SVM-HOG}$	0.0011965
Fusion classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$	0.0256
Fusion classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$	0.2454

Table 9: Comparison of Precision and Recall on basis of PR-curve for classifiers used in the designed procedure

Classifier	**Precision**	**Recall**
Context-based classifier ψ_{AR}	0.59%	98%
Fast-RCNN $\psi_{Fast-RCNN}$	71%	4.9%
SVM-HOG $\psi_{SVM-HOG}$	21.1%	0.06%
Fusion classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$	1.3%	99%
Fusion classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$	17.7%	40%

Table 10: Comparison of area under ROC curve and average precision for classifiers used in the design procedure

Classifier	**ROC-AUC**	AP_{ped}
Context-based classifier ψ_{AR}	0.66	0.5%
Fast-RCNN $\psi_{Fast-RCNN}$	0.70	19.6%
Fusion classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$	0.756	7.5%

Table 11 shows the consumed time to detection. Time consumed for region hypothesis generation using brute-force method is not consider, only the time take for featuring engineering and classification is shown in table 11. As it is observed Fast-RCNN consumes maximum time, it is because of the computation complexity involved in convolution and pooling layer for feature map generation at each layers before final inner product linear classification layer. The context-based classifier and deep fusion classifier consumes only $59msec$ and $35msec$ respectively, which is very negligible. This shows that fusion does not add up to the delayed latency.

Table 11: Comparison of time consumed

Classifier	**Time consumed**
Context-based classifier ψ_{AR}	$\approx 59msec$ for ≈ 2300 region proposals
Fast-RCNN $\psi_{Fast-RCNN}$	$\approx 25.97sec$ for ≈ 2300 region proposals
SVM-HOG $\psi_{SVM-HOG}$	$\approx 140msec$ using sliding window
Fusion classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$	$\approx 10msec$ using sliding window
Fusion classifier $\psi_H(\xi_{Fast-RCNN}, \xi_{AR})$	$\approx (35msec)$ for ≈ 2300 region proposals

Chapter 8

Conclusion and Future work

Conclusion

This work focused on integrating context-based information explicitly to Deep learning detector (Fast-RCNN) and shallow based detector (SVM-HOG) to address the problem of pedestrian detection. Fusing of context-based classifier at the post processing stage of Fast-RCNN and SVM-HOG was proposed and evaluated on different metrics. It is perceived from the obtained evaluation results that, the utilized classifiers ($\psi_{SVM-HOG}, \psi_{AR}$) in designing shallow approach as described in section 6.3.2 leads to very poor performance with F_1 score of 0.0256. This is due to the adapted SVM-HOG classifier ($\psi_{SVM-HOG}$) which is trained on Daimler detection dataset [26] has very low recall and precision on Caltech pedestrian dataset [1], therefore this elevates the ambiguity in representation of appropriate feature engineering involved in shallow learning. Whereas, the utilized deep architecture Fast-RCNN [9] fine-tuned on PASCAL VOC dataset [7] shows better performance compared to the $\psi_{SVM-HOG}$ used in shallow approach. The feature engineering task involved in deep learning is inherent and allows to have high generalization compared to $\psi_{SVM-HOG}$. Comparing the F_1 scores from the table 8, the proposed method of fusing the context-based classifier with the deep network Fast-RCNN as improved the usefulness and performance from $F_1 = 0.0916$ to $F_1 = 0.2452$.

The context-based classifier ψ_{AR} is governed on just scale feature (aspect-ratio) of an object, therefore it has high false positive rate compared to other classifier. After fusing ψ_{AR} classifier with Fast-RCNN in post-processing stage it induces high false positives allowing the average precision (AP) of fusion classifier $\psi_H(\delta_{Fast-RCNN}, \delta_{AR})$ to drop from AP=19.6% to AP=7.5%. Hence to increase the average precision of fused model $\psi_H(\delta_{Fast-RCNN}, \delta_{AR})$ the contextual informations like spatiotemporal and semantic information along with aspect-ratio that are relative to pedestrian should be considered during modelling context-based classifier ψ_{AR}. The table 11 displays the computation intensity involved by Fast-RCNN, most of the time consumed is for the feature engineering task involved inside the deep architecture. Also it is evident that, the context based classifier ψ_{AR} and the time taken by fusion classifier $\psi_H(\delta_{SVM-HOG}, \delta_{AR})$ is very negligible. To conclude, integrating context

information with deep learning will provide an sophisticated object detector that address the problem of object detection in the area of computer vision and machine learning.

Future work

Future work aims at integrating more context information and even addressing the problem of occlusion. Designing of end-to-end deep models that have the ability to learning intuitive information implicitly for data for object detection task will be focused. Deep Learning models are associated with millions of parameters and require billions of floating point operations per second, which limits the ability to inference on low power mobile platforms for real time application. It is important to focus on dedicated hardware such as Nvidia Drive PX/ PX2 [48] and highly optimized BLAS functions which makes it easy to deploy Deep Learning models that can operate in real time. Considering the fact that Deep Learning models are highly redundant, *Deep Compression* techniques will be focused to decrease the redundancy and reduce the computation expenses without performance degradation.

Bibliography

[1] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[3] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun, "Pedestrian detection with unsupervised multi-stage feature learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[8] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "Ssd: Single shot multibox detector," *arXiv preprint arXiv:1512.02325*, 2015.

[14] https://github.com/rbgirshick/fast rcnn, "Fast-rcnn pre-trained model."

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.

[16] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: a machine learning research library," *arXiv preprint arXiv:1308.4214*, 2013.

[17] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[20] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," 2009.

[21] Y. Zheng, C. Shen, R. Hartley, and X. Huang, "Effective pedestrian detection using center-symmetric local binary/trinary patterns," *arXiv preprint arXiv:1009.0892*, 2010.

[22] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1751–1760, IEEE, 2015.

[23] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 794–801, IEEE, 2009.

[24] G. Bradski *et al.*, "The opencv library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[25] A. Ess, B. Leibe, and L. Van Gool, "Depth and appearance for mobile scene analysis," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[26] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[27] G. Hirtz, "Stereo vision." Lecture notes Computer Vision TU Chemnitz, June 2014.

[28] N. Dalal, *Finding people in images and videos*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[30] T. Joachims, "Making large scale svm learning practical," tech. rep., Universität Dortmund, 1999.

[31] A. Karpathy, "Convolutional neural networks for visual recognition." Lecture notes CS231n, Winter 2015.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[33] Y. B. Ian Goodfellow and A. Courville, *Deep Learning*. 2016. Book in preparation for MIT Press.

[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, pp. 818–833, Springer, 2014.

[35] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[36] S. R. Gunn *et al.*, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, 1998.

[37] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, 2006.

[38] A. R. Webb, *Statistical pattern recognition*. John Wiley & Sons, 2003.

[39] K. Kashin, "Statistical inference: Maximum likelihood estimation." Notes Statistics, Spring 2014.

[40] E. Zivot, "Maximum likelihood estimation," 2001.

[41] M. D. Happel and P. Bock, "Analysis of a fusion method for combining marginal classifiers," in *International Workshop on Multiple Classifier Systems*, pp. 137–146, Springer, 2000.

[42] I. I. Jose A. Lozano, Guzmán Santafé, "Classifier performance evaluation and comparison." International Conference on Machine Learning and Applications (ICMLA 2010), December 2010.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[44] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[45] M. D. Happel and P. Bock, "Analysis of a fusion method for combining marginal classifiers," in *International Workshop on Multiple Classifier Systems*, pp. 137–146, Springer, 2000.

[46] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 451–466, Springer, 2013.

[47] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision*, pp. 297–312, Springer, 2014.

[48] http://www.nvidia.de/object/drive-px de.html, "The ai computer for self-propelled cars."