HOG and Gabor Filter Based Pedestrian Detection Using Convolutional Neural Network

Fahim Ahmed
(Student ID: 140217)

Badhon Ahmed Topu
(Student ID: 140229)

Computer Science and Engineering Discipline
Khulna University
Khulna-9208, Bangladesh

Khulna University
Computer Science and Engineering Discipline

The undersigned hereby certify that Fahim Ahmed and Badhon Ahmed Topu have read and recommend to the Computer Science and Engineering Discipline for acceptance of a thesis entitled "HOG and Gabor Filter Based Pedestrian Detection Using Convolutional Neural Network" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science Engineering (CSE).

Dated: July 19, 2018

S.M. Mohidul Islam
Assistant Professor
Computer Science and Engineering Discipline
Khulna University, Bangladesh

Thesis Supervisor

Dr. MD. Anisur Rahman
Professor
Computer Science and Engineering Discipline
Khulna University, Bangladesh

Head of the Discipline

Abstract

Pedestrian detection is an essential research topic due to its major importance especially in the fields of automotive, surveillance and robotics. In spite of having the tremendous improvements, pedestrian detection is still an open challenge and searching for more and more accurate algorithms. In the last few years, deep learning more specifically Convolutional Neural Networks emerged as the state of the art in terms of accuracy for a number of computer vision tasks such as image classification, object detection, and segmentation. In this paper, we propose a pedestrian detection system based on Gabor filter, Histogram of Oriented Gradient and Convolutional Neural Networks. We have taken INRIA and Daimler dataset, two most used datasets for pedestrian detection. By applying this method on those two dataset, we have achieved comparatively good accuracy close to the state of the art approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Pedestrian detection is an established instance of object detection. It is useful technique which provides safety for both road drivers and also pedestrians themselves. There exists various kinds of technology for detecting pedestrians using either hardware or software.

The pedestrian detection system can be very useful for monitoring traffic flow, on-board and on-site vehicle drivers assisting system, or the intelligent pedestrian crossing system, etc. By using computer vision, pedestrians can be detected and segmented from complex background. Because of its versatile applications in various sectors such as car safety, surveillance, and robotics, it has attracted much attention in the last years. But pedestrian detection from image or video with enough perfection is still a big problem in computer vision field.

Pedestrians detection is challenging task. Generally, detection and identification manually which is very expensive and time consuming. Moreover, the accuracy of detecting an unknown person manually, is almost tough for a man. Most of crime investigation authority use to investigate a video footage after any crime have occurred. So the rate of solving problem is not good enough.

If there exist any responsive system that detect pedestrian instantly, the rate of crime can be reduced. So, a fast and accurate pedestrians detection system is very essential. Many researchers did several researches on detection and recognition of pedestrian but there work is not too much satisfactory till now. Previous researches where some provide pedestrian detection for clear straight image and some provide detection method those are not fast.

1.2 Motivation

As pedestrian detection is very useful and important topic, many researches were done for improving detection accuracy. Most used methods are Support Vector Machine (SVM), AdaBoost, Decision Tree, Neural Network and many other established Machine learning approaches to detect a pedestrian. For feature extraction Histogram of Oriented Gradient (HOG), Gabor filter, Haar, Kalman filer etc. Using SVM and HOG good accuracy was

obtained. From all the mentioned method, Convolutional Neural Network a Deep learning approach is gaining much popularity in recent years [10]. Moreover, Deep learning method is best for its detection accuracy [10]. Convolutional Neural Network is specially designed for the Computer Vision field. It works better on image data[4]. HOG is a widely used feature extraction method and Gabor filter is also used for feature extraction. Together this two method works well [11]. We have used the Gaussian filter for removing noise and Gabor and HOG for extracting features. The convolutional neural network has worked tremendously well with these two feature extractors.

Summary:

In this chapter, we have discussed about the background and motivation of pedestrian detection. In the last few years, many research was done on pedestrian detection. We have discussed about the process, importance, and application of the detection in daily life. In the next chapter, we will discuss about literature survey which is related to the object and pedestrian detection.

1.3 Objectives

The main objectives of our work are-

- To develop robust pedestrian detection system.
- To find comparatively better detection accuracy.

1.4 Thesis Organization

This thesis has been organized into five chapters. Each chapter provides different concepts of our research.

Chapter I (Introduction): The background of our research is given in this chapter. The objective and motivation are also within this section.

Chapter II (Literature Survey): This chapter presents basic concepts and methods about pedestrian detection and recognition system.

Chapter III (Related Works): This chapter presents the related work and drawbacks of existing system.

Chapter IV (Methodology): This chapter presents working method and system architecture of our work.

Chapter V (Experimental Results): This chapter presents experimental results of our work.

Chapter VI (Conclusion and Future Direction): This chapter presents the summary of our research.

Chapter 2

Literature Survey

2.1 Introduction

In past few years and current year a great number of research have been done on pedestrian detection and recognition. Most of the studies started with collecting the desired dataset from internet and various office. Then they applied various filtering technique to remove noise data and used various segmentation technique. Finally, they extracted features and applied various classifier algorithm. Here we review several types of analysis that most researches have used on multi pedestrian detection and recognition.

2.2 Various Filtering Techniques

2.2.1 Mean Filtering

Mean filtering is a simple, intuitive and easy to implement method of smoothing images i.e reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images. Mean filtering is usually used as a convolution filter. Like other convolutions it is based around a kernel, which represent the shape and size of the neighborhood to be sampled when calculating the mean [12, 13, 14].

2.2.2 Median Filtering

Median filter is normally used to reduce noise in image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image. The median filter considers each pixel in the image in the term and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used [12, 13, 14, 15].

2.2.3 Gaussian Filtering

The Gaussian smoothing operator is a 2D convolution operator that is used to blur images and remove detail and noise. In this sense, it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian. The idea of Gaussian smoothing is to use 2-D distribution as a point-spread function and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discreet approximation to the Gaussian function before we can perform the convolution.

In 2-D, an isotropic Gaussian has the form:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

The effect of Gaussian smoothing is to blur an image, in a similar fashion to the mean filter. The Gaussian outputs a weighted average of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter [13, 16, 17, 18].

2.3 Feature Extraction Methods

In machine learning, pattern recognition and image processing feature extraction is a special form of dimensionality reduction. When the input data is to an algorithm is too large to be processed and it is suspected to be redundant then the input data will be transformed into a reduced representation set of features which is also called feature vector. Transforming input data into a set of features is called feature extraction. These features play a fundamental role in classification. In image processing, there are some visual features color, texture and shape. These features are also called global feature. Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) are local features of image. The features are necessary for differentiating one class of object from another.

2.3.1 Gabor Filter

Gabor filters are band pass filters which are used in image processing for feature extraction, texture analysis, and stereo disparity estimation [19]. The impulse response of these filters is created by multiplying a Gaussian envelope function with a complex oscillation. Gabor showed that these elementary functions minimize the space (time)-uncertainty product. By extending these functions to two dimensions it is possible to create filters which are selective for orientation. Under certain conditions the phase of the response of Gabor filters is approximately linear. This property is exploited by stereo approaches which use the phase-difference of the left and right filter responses to estimate the disparity in the stereo images [19, 20]. It was shown by several researchers that the profile of simple-cell receptive fields in the mammalian cortex can by described by oriented twodimensional Gabor functions.

Complex

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\bigg(- \frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \bigg) exp\bigg(i\bigg(2\pi \frac{x'}{\lambda} + \psi \bigg) \bigg) \tag{2.2}$$

Real

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\bigg(- \frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \bigg) \cos\bigg(i\bigg(2\pi \frac{x'}{\lambda} + \psi \bigg) \bigg) \tag{2.3}$$

Imaginary

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\bigg(- \frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \bigg) \sin\bigg(i\bigg(2\pi \frac{x'}{\lambda} + \psi \bigg) \bigg) \tag{2.4}$$

Where

$$x = x \cos\theta + y \sin\theta \tag{2.5}$$

and

$$y = x \sin\theta + y \cos\theta \tag{2.6}$$

In this equation, = wavelength of the sinusoidal factor, = orientation of the normal to the parallel stripes of a Gabor function, = phase offset, = standard deviation of the Gaussian envelope and = spatial aspect ratio.

2.3.2 Local Binary Pattern

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis.

The basic idea for developing the LBP operator was that two-dimensional surface textures can be described by two complementary measures: local spatial patterns and gray scale contrast. The original LBP operator [21] forms labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number. This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation [22]. The LBP operator was extended to use neighborhoods of different sizes[23].

2.3.3 Haar Wavelet

The Haar wavelet is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis. Wavelet analysis is similar to Fourier analysis in that it allows a target function over an interval to be represented in terms of an orthonormal basis. The Haar sequence is now recognized as the first known wavelet basis and extensively used as

a teaching example. The Haar sequence was proposed in 1909 by Alfrd Haar [24]. Haar used these functions to give an example of an orthonormal system for the space of square-integrable functions on the unit interval [0, 1]. The study of wavelets, and even the term "wavelet", did not come until much later. As a special case of the Daubechies wavelet, the Haar wavelet is also known as Db1. The Haar wavelet is also the simplest possible wavelet. The technical disadvantage of the Haar wavelet is that it is not continuous, and therefore not differentiable. This property can, however, be an advantage for the analysis of signals with sudden transitions, such as monitoring of tool failure in machines [25].

2.3.4 Histogram of Oriented Gradients

The Histogram of Oriented Gradient (HOG) feature descriptor is popular for object detection [26].

Compute a Histogram of Oriented Gradients (HOG) by

1. Global image normalization
2. Computing the gradient image in x and y
3. Computing gradient histograms
4. Normalizing across blocks
5. Flattening into a feature vector

The first stage applies an optional global image normalization equalization that is designed to reduce the influence of illumination effects. In practice we use gamma (power law) compression, either computing the square root or the log of each color channel. Image texture strength is typically proportional to the local surface illumination so this compression helps to reduce the effects of local shadowing and illumination variations. The second stage computes first order image gradients. These capture contour, silhouette and some texture information, while providing further resistance to illumination variations. The locally dominant color channel is used, which provides color invariance to a large extent. Variant methods may also include second order image derivatives, which act as primitive bar detectors - a useful feature for capturing, e.g. bar like structures in bicycles and limbs in humans.

The third stage aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in pose or appearance. The adopted method pools gradient orientation information locally in the same way as the SIFT feature [27]. The image window is divided into small spatial regions, called cells. For each cell we accumulate a local 1-D histogram of gradient or edge orientations over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic orientation histogram representation.

The fourth stage computes normalization, which takes local groups of cells and contrast normalizes their overall responses before passing to next stage. Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram energy over local groups of cells that we call blocks. The result is used to normalize each cell in the block.

The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector for use in the window classifier.

2.3.5 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is an image descriptor for image-based matching and recognition developed by David Lowe [27, 28]. Experimentally, the SIFT descriptor has been proven to be very useful in practice for image matching and object recognition under realworld conditions. In its original formulation, the SIFT descriptor [27, 28] comprised a method for detecting interest points from a grey-level image at which statistics of local gradient directions of image intensities were accumulated to give a summarizing description of the local image structures in a local neighborhood around each interest point, with the intention that this descriptor should be used for matching corresponding interest points between different images. This method for detecting interesting points in the SIFT operator can be seen as a variation of a scaleadaptive blob detection method proposed by Lindeberg [29], where blobs with associated scale levels are detected from scale-space extrema of the scale-normalized Laplacian. The scalenormalized Laplacian is normalized with respect to the scale level in scale-space and is defined as

$$\nabla^2_{norm} L(x, y; s) = s\big(L_{ss} + L_{yy}\big) = s\left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}\right) = s\nabla^2(G(x, y; s) * f(x, y)) \tag{2.7}$$

From smoothed image values L(x,y;s) computed from the input image f(x,y) by convolution with Gaussian kernels

$$G(x, y; s) = \frac{1}{2\pi s} e^{-\frac{(x^2+y^2)}{2s}} \tag{2.8}$$

Of different widths $s = \sigma^2$, where σ denotes the standard deviation and s the variance of the Gaussian kernel. Then, the scale-space extrema are detected from the points $(x, y; s)$ in scale-space at which the scale-normalized Laplacian assumes local extrema with respect to space and scale. In a discrete setting, such comparisons are usually made in relation to all neighbours of a point in a 333 neighbourhood over space and scale. The difference-of-Gaussians operator constitutes an approximation of the Laplacian operator

$$DOG(x, y; s) = L(x, y; s + \Delta s) - L(x, y; s) \approx \frac{\Delta s}{2} L(x, y; s) \tag{2.9}$$

which by the implicit normalization of the differences-of-Gaussian responses, as obtained by a self-similar distribution of scale levels used by Lowe, also constitutes an approximation of the scale-normalized Laplacian with $\Delta s \nabla^2 L = (k^2 - 1)t\nabla^2 L = (k^2 - 1)\nabla^2_{norm} L$, thus implying

$$DOG(x, y; s) \approx \frac{(k^2 - 1)}{2} \nabla^2_{norm} L(x, y; s) \tag{2.10}$$

It can be shown that this method for detecting interest points leads to scale-invariance in the sense that

1. The interest points are preserved under scaling transformations and

2. The selected scale levels are transformed in accordance with the amount of scaling [29]. Hence, the scale values obtained from these interest points can be used for normalizing local neighbourhoods with respect to scaling variations [30] which is essential for the scale-invariant properties of the SIFT descriptor; see also for an overview of the scale-space theory on which these image operations are based. The Laplacian operation is rotationally invariant. Therefore,

3. These interest points will also be rotationally invariant.

Another way of detecting scalespace extrema of the Laplacian efficiently for real-time implementation has been presented by Lindeberg and Bretzner [31] based on a hybrid pyramid. A closely related method for real-time scale selection has been developed by Crowley and Riff [32].

2.4 Classifier

2.4.1 Convolutional Neural Network

Convolutional networks were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

Convolutional Neural Networks recorded amazingly good performance in several tasks, including digit recognition, image classification and face recognition. The key idea behind CNNs is to automatically learn a complex model that is able to extract visual features from the pixel-level content, exploiting a sequence of simple operations such as filtering, local contrast normalization, non-linear activation, local pooling. For instance, the Viola-Jones [33] features come from the observation that the shape of a pedestrian is characterized by abrupt changes of pixel intensity in the regions corresponding to the contour of the body.

Conversely, Convolutional Neural Networks do not exploit human intuitions but only rely on large training datasets and a training procedure based on back propagation, coupled with an optimization algorithm such as gradient descent. The first layers of the network typically identify low-level concepts such as edges and details, whereas the final layers are able to combine low-level features so as to identify complex visual concepts. Convolutional Neural Networks are typically trained resorting to a supervised procedure that, besides learning ad-hoc features, defines a classifier as the last layer of the network. Despite being powerful and effective, the interpretability of such models is limited. Moreover, being very complex model consisting of up to hundreds of millions of parameters, CNNs need large annotated training datasets to yield accurate results [4].

Layer of Convolutional Neural Network

- **Convolutional layer**

  The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

- **Pooling layer**

  Another important concept of CNNs is pooling, which is a form of non-linear downsampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The pooling operation provides another form of translation invariance. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.

  Average pooling was often used historically but has recently fallen out of favor compared to max pooling, which works better in practice [34]. Due to the aggressive reduction in the size of the representation, the trend is towards using smaller filters [35] or discarding the pooling layer altogether [36].

  Pooling is an important component of convolutional neural networks for object detection based on Fast R-CNN [37] architecture.

- **ReLU layer**

  ReLU is the abbreviation of Rectified Linear Units. This layer applies the nonsaturating activation function

$$f(x) = max(0, x) \tag{2.11}$$

  It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent

$$f(x) = tanh(x) \tag{2.12}$$

$$f(x) = |tanh(x)| \tag{2.13}$$

  And the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

ReLU is often preferred to other functions, because it trains the neural network several times faster [38] without a significant penalty to generalization accuracy.

- **Fully connected layer**

  Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

- **Loss layer**

  The loss layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in [0,1] . Euclidean loss is used for regressing to real-valued labels $(-\infty, \infty)$.

Architecture of Convolutional Neural Networks

There are several architectures in the field of Convolutional Networks that have a name. The most common are:

- **LeNet**

  The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990s. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc [29].

- **AlexNet**

  The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer) [29].

- **ZF Net**

  The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller [29].

- **GoogLeNet**

  The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to matter much. There are also several followup versions to the GoogLeNet, most recently Inception-v4 [29].

- **VGGNet**

  The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Their pretrained model is available for plug and play use in Caffe. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters [29].

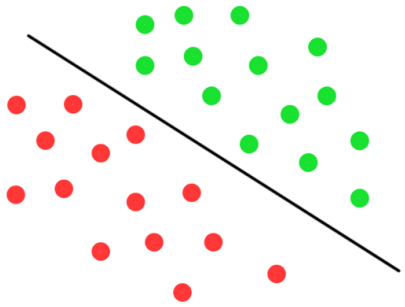
- **ResNet**

  Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 10, 2016). In particular, also see more recent developments that tweak the original architecture from Kaiming He et al. Identity Mappings in Deep Residual Networks (published March 2016) [39].

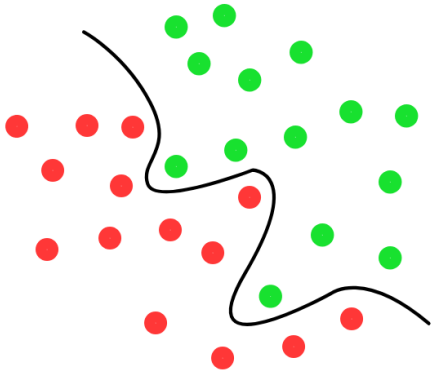
2.4.2 Support Vector Machines

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects



of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.



The illustration below shows the basic idea behind Support Vector Machines. Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation)[40].

Classification of SVM

Classification of SVM Type 1

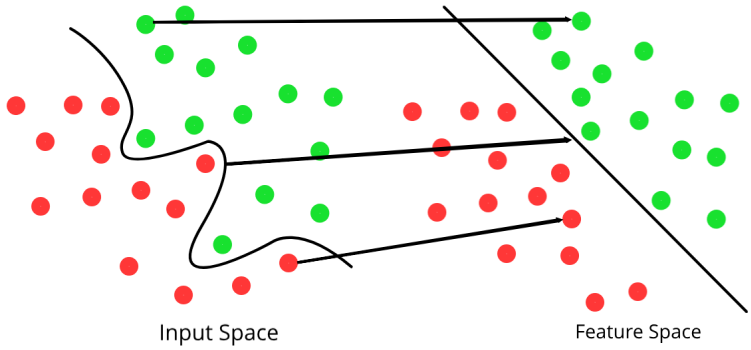
For this type of SVM, training involves the minimization of the error function:

$$\frac{1}{2}w^T w + C\sum_{i=1}^{N}\zeta_i \tag{2.15}$$

subject to the constraints:

$$y_i(w^T\phi(x_i)+b) \geq 1-\zeta_i \quad \text{and} \quad \zeta_i \geq 0, i = 1, ...N \tag{2.16}$$

where C is the capacity constant, w is the vector of coefficients, b is a constant, andrepresents parameters for handling nonseparable data (inputs). The index i labels the N



training cases. Note that $y\epsilon \pm 1$ represents the class labels and x_i represents the independent variables. The kernel ϕ is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting.

Classification of SVM Type 2

In contrast to Classification SVM Type 1, the Classification SVM Type 2 model minimizes the error function:

$$\frac{1}{2}w^T w - v\rho + \frac{1}{N}\sum_{i=1}^{N}\zeta_i \tag{2.17}$$

subject to the constraints:

$$y_i(w^T\phi(x_i) + b) \geq \rho - \zeta_i, \zeta_i \geq 0, i = 1, ..., N \quad \text{and} \quad \rho \geq 0 \tag{2.18}$$

In a regression SVM, you have to estimate the functional dependence of the dependent variable y on a set of independent variables x. It assumes, like other regression problems, that the relationship between the independent and dependent variables is given by a deterministic function f plus the addition of some additive noise.

2.4.3 Recurrent Neural Networks

The human brain is a recurrent neural network (RNN): a network of neurons with feedback connections. It can learn many behaviors / sequence processing tasks / algorithms / programs that are not learnable by traditional machine learning methods. This explains the rapidly growing interest in artificial RNNs for technical applications: general computers which can learn algorithms to map input sequences to output sequences, with or without a teacher. Our recent applications include adaptive robotics and control, handwriting recognition, speech recognition, keyword spotting, music composition, attentive vision, protein analysis, stock market prediction, and many other sequence problems.

Early RNNs of the 1990s could not learn to look far back into the past. Their problems were first rigorously analyzed on Schmidhuber's RNN [41] long time lag project by his former

PhD student Hochreiter. A feedback network called "Long Short-Term Memory" (LSTM, Neural Comp., 1997) [42] overcomes the fundamental problems of traditional RNNs, and efficiently learns to solve many previously unlearnable tasks involving:

1. Recognition of temporally extended patterns in noisy input sequences
2. Recognition of the temporal order of widely separated events in noisy input streams
3. Extraction of information conveyed by the temporal distance between events
4. Stable generation of precisely timed rhythms, smooth and non-smooth periodic trajectories

2.4.4 Artificial Neural Networks

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts.

These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers.

Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing.

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provides us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 is typical. They have a myriad of parts, sub-systems, and control mechanisms. They convey information via a host of electrochemical pathways.

These artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism. But for the software engineer who is trying to solve problems, neural computing was never about replicating human brains. It is about machines and a new way to solve problems [43].

Summary:

In this chapter, we have discussed various filtering technique, feature extraction method, and classifier. All of these elements are very useful in computer vision field, especially in detection and recognition type research. In the next chapter, we will discuss the works which are related to pedestrian detection.

Chapter 3

Related Works

Pedestrain detection and recognition is very important issue in the field of computer vision.It is useful technique which provides safety for both road drivers and also pedestrians themselves. Object detection more specifically pedestrain detection has received great attention during recent years. Some research papers related to this field are mentioned below:

3.1 Scale-aware Fast R-CNN for Pedestrian Detection [1]

Jianan Li et al. proposed the method based on Scale-Aware Fast R-CNN (SAF R-CNN). Their model introduced multiple built-in sub networks which detect pedestrians with scales from disjoint ranges. They could detect pedestrians of large and small sizes, respectively. Their architecture of proposed SAF R-CNN is based on popular Fast R-CNN detection framework due to its superior performance and computation efficiency in detecting general objects. They introduced a new detector called ACF detector which is a fast and effective sliding window based detector that performs quite well on rigid object detection. Their proposed model achieved 90.68% accuracy on Caltech dataset and 91.06% on INRIA dataset.

Limitations:

Their model gave best result on small-size pedestrian instances. It is not perfectly designed for detecting general object.

3.2 Multispectral Deep Neural Networks for Pedestrian Detection [2]

J Liu et al. used deep neural network specifically ConvNet and Fast R-CNN for detecting pedestrians. They designed four distinct ConvNet architecture based on different DNN stages. Their model can reduce the missing rate of baseline method Faster R-CNN by 11%, yielding a 37% overall missing rate on KAIST, which is also 3.5% lower than the other proposed fusion models. They achieved 83% accuracy on Caltech dataset.

Limitations:

They achieved a good accuracy only for multispectral (color and thermal images) pedestrian detection.

3.3 Pedestrian Detection using Augmented Training Data [3]

J. Nilsson et al used Histogram of Oriented Gradient (HOG) for extracting features from object and used Support Vector Machine (SVM) as a classifier. They used augmented dataset for pedestrian detection. Augmented images are generated by rendering virtual pedestrians onto real images backgrounds. On Daimler Mono Pedestrian benchmark data set, they got 81% detection accuracy.

Limitations:

They worked only on their generated augmented dataset but they didnt tested this on a popular dataset like INRIA or Caltech etc.

3.4 Deep Convolutional Neural Networks for Pedestrian Detection [4]

D. Tom et al. described a deep convolutional neural network for pedestrians detection. They used Deep learning and Histogram of Gradient (HOG) for features extraction and Support Vector Machine (SVM) as a classifier. According to the extracted features, dataset was trained by SVM. In their paper they introduced some novel contributions. Those are, they optimized most of the stages of the pedestrian detection pipeline, proposing novel solutions that significantly improve the detection accuracy, they approached state-of-the-art performance in terms of detection accuracy, outperforming both traditional methods based on handcrafted features and deep learning approaches, they proposed a lightweight version of their algorithm that runs in real-time on modern hardware, they validated their approach by implementing it on an NVIDIA Jetson TK1, a compact computational platform based on a Graphics Processing Unit that is being adopted as the computational brain of several car prototypes featuring modern safety systems. Since they used deep learning and Locally Decorrelated Channel Features (LDCF), their accuracy is good enough.

Limitations:

Their work unable to optimize the LDCF for region proposal and they couldnt implement their pipeline entirely on GPU.

3.5 Multi-Target Detection in CCTV Footage for Tracking Application Using Deep Learning Techniques [5]

A. Dimou et al. described those methodologies to improve the efficiency of deep learning based multi-target detectors in challenging CCTV footage are proposed. In their work, they conducted with the use of heterogeneous training data and data augmentation was explored to improve their detection rate in challenging CCTV scenes. Moreover, they proposed to use the objects spatial transformation parameters to automatically model and predicted the evolution of intrinsic camera parameters and accordingly tune the detector for better performance. The proposed approaches were tested on publicly available datasets and real-world CCTV videos. The RNN was used to predict the affine transformation of the objects and dynamically modify the parameters of the detector. Experimental validation of the proposed concept was performed on real CCTV videos. The proposed framework was applicable to any type of objects but experiments will focus on pedestrians. For VOC2007 dataset their proposed method showed 59.44% efficiency on No Blur background, 31.71% on Blur 5px background and 23.50 on Blur 10px background. Limitations: Their proposed method showed different efficiency on basis of blur background that leads to dependency on color factor and moving criteria.

3.6 Object Recognition and Detection with Deep Learning for Autonomous Driving Applications [6]

Yakup Demir et al. described about the object detection and recognition using deep learning. In this work, they proposed a hybrid system using both the Convolution Neural Network (CNN) and the Support Vector Machine (SVM) for object recognition and pedestrian detection. They took the real environment embankments such as variations in light conditions, partial occlusion, the presence of shadows, and surrounding background clutters as parameter. They presented a LMCNN-SVM system to handle these challenges in their paper. In their system, they used a pretrained AlexNet architecture and a new CNN architecture including nine layers. They divided whole images into patches and employed the CNN to extract their discriminative features. Then they applied PCA to the features obtained from the CNN in order to decorrelate and reduce them. Finally, they imported them to the input of the SVM classifiers to increase the generalization ability of the system and, finally, effectively fused the images by using a majority voting rule.

They observed that the LM-CNN-SVM system achieved the highest values of accuracy, 89.80% and 92.80% for 15 and 30 images per class, respectively. They experienced that their proposed method will continue to improve object recognition and detection in terms of both accuracy and speed in using for real-time applications.

3.7 Pedestrian and Part Position Detection using a Regression-based MultipleTask Deep Convolutional Neural Network [7]

Takayoshi Yamashita et al. proposed a method to detect both the pedestrian and their position simultaneously using a regression-based deep convolutional neural network (DCNN). Their research says that this simultaneous detection method is possible to train efficient features for both tasks, because it is attention to head and leg regions from given labels. In that experiments, their method improves the performance of pedestrian detection compared with the DCNN which detects only pedestrian. The proposed approach also improves the detection accuracy of the head and leg positions compared with the methods that detect only these positions. Using the results of position detection and the camera parameters, their method achieves distance estimation to within 5% error.

3.8 Pedestrian Detection Based on Deep Convolutional Neural Network with Ensemble Inference Network [8]

Hiroshi Fukui et al. proposed the methods based on Convolutional Neural Networks (CNN) that achieves high accuracy in various fields. They proposed two techniques for improvement of pedestrian detection by random selection of the units based on the Dropout. Random Dropout that is randomly setting corresponding value of the units to zero with flexible rate. Ensemble Inference Network (EIN) constructs multiple networks that have different structure in fully connected layer with decision manner of final output. They achieved the comparable performance to state-of-theart methods in Deep Learning approach, even the structure of proposed methods are significant simple.

Limitations:

Their proposed method has additional computational cost for real time processing.

3.9 Pedestrian Movement Direction Recognition Using Convolutional Neural Networks [9]

Alex Dominguez-Sanchez and Miguel Cazorla described their method for recognition the movement direction of pedestrian using convolution neural network. They had presented a method to differentiate the motion of pedestrians in real life environments. By building a novel inputfiltered image based on the post-processing of static recorded video frames, they had managed to successfully distinguish three different pedestrian movement directions. Moreover, they had demonstrated how the results could be enhanced even further by searching for the best hyperparameters once the CNN had been fine-tuned for their specific problem, in that case tuning the momentum and weight decay CNN parameters.

They had also presented an evaluation of the current state-of-the-art CNNs, with ResNet being the best-performing CNN for their image recognition problem (94% accuracy in the validation set and 79% in the test set).

Limitation Their work is not still a better and robust use of data augmentation.

Summary:

In this chapter, we have discussed most recent papers on the object and pedestrian detection. We have shown the existing methods, performance, and limitations of those works. After reading this chapter we may have an overview of the whole process of detection mechanisms. In the next chapter, we will discuss the methodology of our proposed work.

Chapter 4

Methodology

4.1 Introduction

The proposed working model system is shown in Figure 4.1 . To detect pedestrian from images, the preprocessing steps and feature vectors are extracted from the image. Then the image passes through the classifier where the pre-trained model and weights come from the database. The model then predicts the result if there any pedestrian contain in the image or not.

Different combinations of classifier model techniques are investigated to get better result.

4.2 Training Phase

We have taken some positive sample and negative samples for the training phase. In the positive samples where pedestrian contains and in the negative samples where no pedestrian contains in the sample. The samples pass through the preprocessing and the feature extraction method. After doing preprocessing and feature extraction the samples divided into a training set and validation set where the training set was used for training the classifier and validation set was used for checking the classifier accuracy during training. By checking accuracy the validation set is improving the classifier. Then the training set and validation set is fed into the classifier. After finishing the training the model and weights are stored in the database.

4.2.1 Preprocessing

In our preprocessing steps we have done several things.

- **Image Resize**

  The collected image is resized to 140x300. Then we have sent it for further processing.

- **Filter**

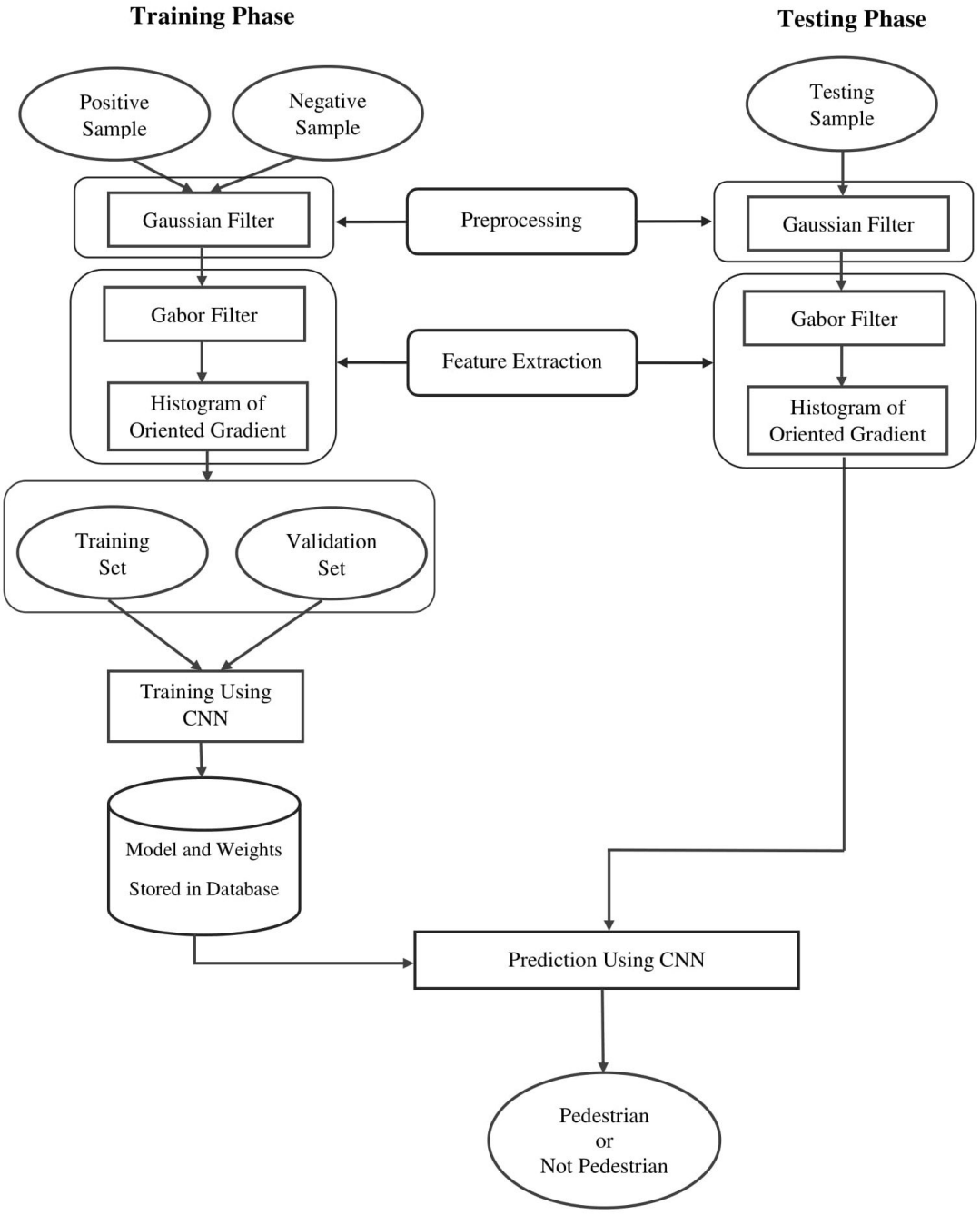


Figure 4.1: System Architecture

As our data does not contain salt and pepper noise , we have used the Gaussian filte to remove noise. The median filter cannot give us a better result. We need a clear image for pedestrian detection. So we have used the Gaussian filter to reduce noise in the image so that we can preserve every useful detail in the image.

The equation of Gaussian filter is given in equation 4.1

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

Here,
x and y = Intensity of the input image,
σ = Standard deviation of the Gaussian distribution.
The preprocessed image was sent for feature extraction.

4.2.2 Feature Extraction

Feature extraction describes the relevant shape information contained in a pattern so that the task of classifying the pattern is made easy by a formal procedure. Features are the machine understandable codes that describe the characteristics of different objects present in an image. They may or may not be understandable to human every time. Feature plays a very important role in the area of image processing. Before getting features, various image preprocessing is applied to the sampled image. After that, feature extraction techniques are applied to get features that will be useful in classifying and recognition of images.

- **Gabor Filter**

  We have tried to segment the parts from a whole scenery image. When the images will be big enough, then we can use big sized kernels. If its like texture then it is preferred to use a smaller size. In our system, we have used a 21x21 kernel size. As we have resized every image 140x300 so we have used big size kernel.

  We have converted the preprocessed color image into a grayscale image. Then we apply the Gabor filter to the grayscale image. After applying Gabor filter the image was sent for HOG feature extraction.

- **Histogram of Oriented Gradients (HOG)**

  In HOG feature extraction method the image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. HOG detect the direction when it enters from high intensity to low intensity and then draws the direction. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block.

  For computing the gradient of the image we use a filter 16x16 cells. The filter describes a patch of an image is that it provides a compact representation. The gradient of this patch contains 2 values (magnitude and direction).

In the previous step, we have created a histogram based on the gradient of the image. Gradients of an image are sensitive to overall lighting. If we make the image darker by dividing all pixel values by 2, the gradient magnitude will change by half, and therefore the histogram values will change by half. Ideally, we want our descriptor to be independent of lighting variations. In other words, we would like to normalize the histogram so they are not affected by lighting variations.

Now the 16x16 cells slide through the whole image and return the HOG extracted image.

- **Histogram of Oriented Gradients and Gabor Filter (HOGG)**

  A new method for extracting feature is HOGG (Histogram of Oriented Gradients and Gabor), which is comparatively much efficient than all other feature extraction method for pedestrian detection [44, 11, 45]. It also gives a better result than only HOG or only Gabor. If we working with a video then the initial step of this method is background subtraction in order to detect moving objects. This process has been done through an algorithm that calculates the background image with the average of the n first frames. After that, the background image is updated by adding one image each m frames. In that case, we need to consider the frame per second (fps) rate of the database used, where values are like n = 15 and m = 7. Then the result of the moving objects detection is introduced to the Gabor module.

  As we have worked with images, first we have applied Gabor filter. Next, the HOG algorithm is applied to the resulting images in order to extract relevant features. Lastly, a classifier is applied using the extracted features to determine if the original image is a person or not.

4.3 Classifier

4.3.1 Convolutional Neural Network

In our methodology we have worked with the convolutional neural network. Convolutional Neural Network recorded amazingly good performance in several tasks, including digit recognition, image classification and object detection and face recognition. The main advantage of CNN is highly accurate in image recognition and detection. The key idea behind CNN is to automatically learn a complex model that is able to extract visual features from the pixel-level content, exploiting a sequence of simple operations such as filtering, local contrast normalization, non-linear activation, local pooling. Convolutional Neural Networks do not exploit human intuitions but only rely on large training datasets and a training procedure based on back propagation, coupled with an optimization algorithm such as gradient descent. The training procedure aims at automatically learning both the weights of the filters so that they are able to extract visual concepts from the raw image content and a suitable classifier.

Model 1 In our first convolutional neural network architecture, we have used three convolutional layers and three hidden layers. In this architecture, the input shape is 64x64. We sent it to the convolutional layer that has 32 filters with 3x3 filter size. We use Rectified Linear Unit (ReLU) activation function that is the positive part of its argument. That means if any negative value wants to pass through layer it can't. It only passes the positive value or zero. The function of ReLU is given below,

$$f(x) = max(0, x) \tag{4.2}$$

Then we sent it to another two convolutional layer that has 64 and 128 filters with 3x3 filter size. After finishing the convolutional layer we convert the feature map into a flattened array. Then we sent it to the hidden layer that is fully connected with each other. In all three hidden layer, we use 128 neurons with activation function ReLU then one output layer with a sigmoid function that is also fully connected with the previously hidden layer. Here in the output layer, the reason we use sigmoid function is that it exists the value between 0 and 1. As our model is a binary classifier and the output will be zero or one like a predictive model so we use a sigmoid function here. The sigmoid function is given below,

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

If the value of output layer is zero then we can say there is no pedestrian contain in the image if the value of output layer is one then we can say there is one or more pedestrian in the image. The model is given in Figure 4.2

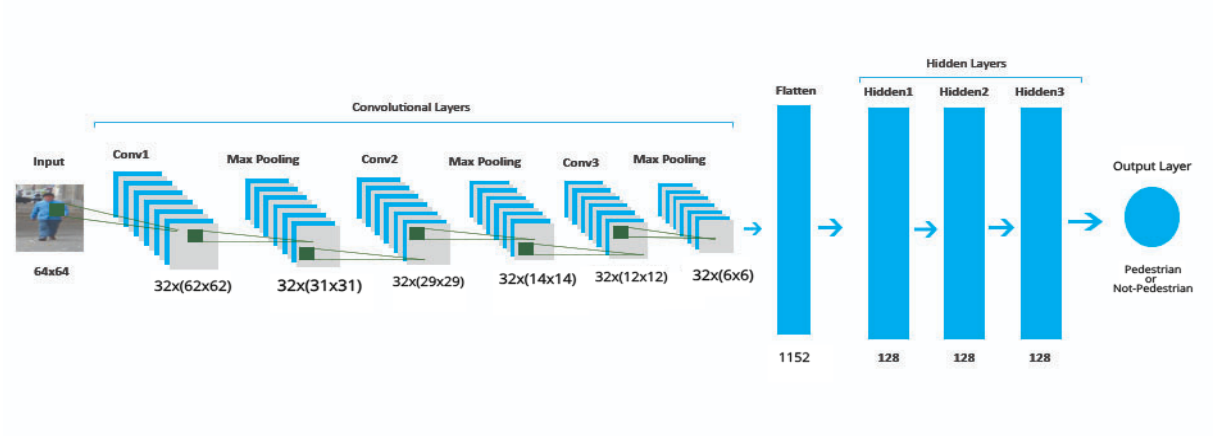


Figure 4.2: Model 1

Model 2 Like our previous convolutional neural network model, we have made some change to get better accuracy and also to compare model one with another. Here we three convolutional layer and three hidden layers. The convolutional layer is the same as the previous one. The model is given in Figure 4.3

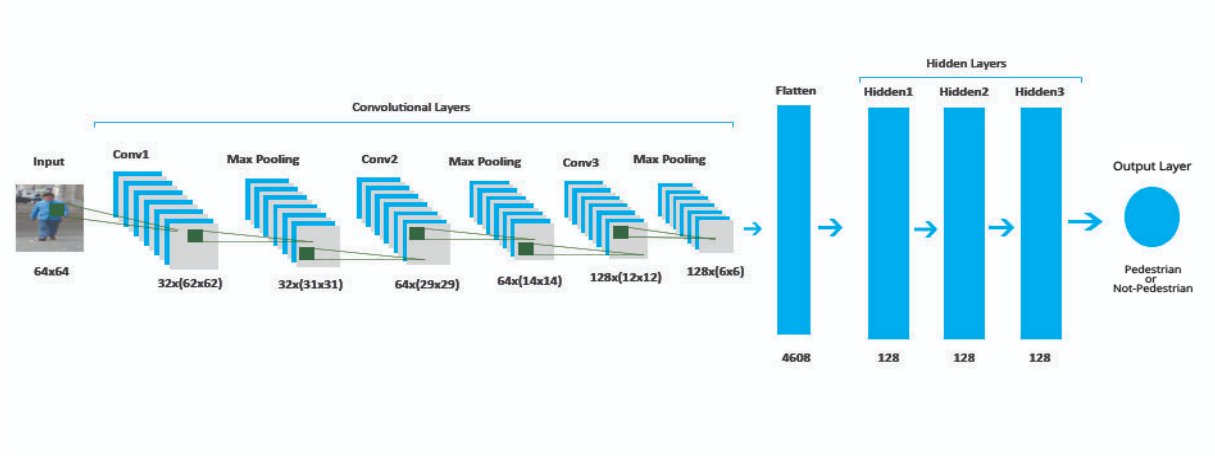


Figure 4.3: Model 2

Model 3 ious convolutional neural network model, we have made some change to get better accuracy and also to compare model one with another. Here we have used four convolutional layer and six hidden layers. The convolutional layer is the same as the previous one. The model is given in Figure 4.4

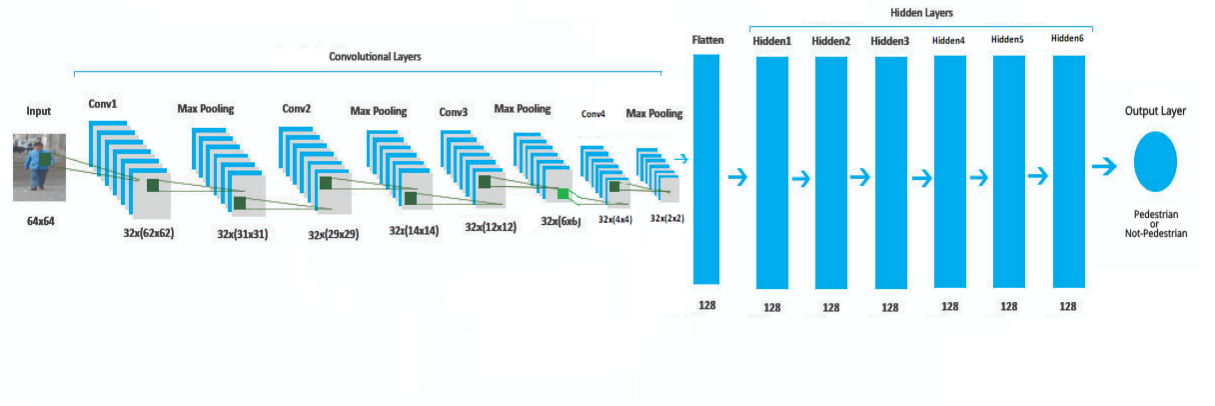


Figure 4.4: Model 3

After finishing the training we have stored the trained model and weights in the database.

4.4 Testing Phase

We have taken some positive and negative sample for testing purpose. The preprocessing and the feature extraction method is the same as the training phase. Then the extracted features have been sent to the classifier to predict the result. We have loaded the pre-trained model and weights then we have sent it to the classifier. If the predicted result is zero then

there is no pedestrian in the image. If the predicted result is one then the image contains at least one pedestrian.

Summary:

In this chapter, we have shown the total process of our proposed work step by step with essential diagram and related figure. We have thoroughly discussed the system architecture of our classifier. We have also discussed about the filter and feature extractor which we have used in our method. In the next chapter, we will discuss the experimental result of our work.

Chapter 5

Experimental Results

5.1 Introduction

In our thesis INRIA, DAIMLER Mono Pedestrian Classification Benchmark Dataset and PennFudanPed have used for training and testing. INRIA dataset contains 5203 images, where we have used 3633 images for training and 1570 images for testing. DAIMLER Mono Pedestrian Classification Benchmark Dataset contains 22404 images, where we have used first 5000 images for training and last 2000 images for testing. The PennFudanPed dataset contains 170 images and these images is used for testing purpose. Each dataset (except PennFudanPed) contains two classes pedestrian and not pedestrian.

The dataset are freely downloadable from:
INRIA - [46]
Daimler Mono Pedestrian Classification Benchmark - [47]

Table 5.1: Categories Of Images In Datasets

Name Of Dataset	Name Of Class		Number Of Images
INRIA	Pedestrian (3535)	Not-Pedestrian (1668)	5203
Daimler Mono Pedestrian Classification Benchmark	Pedestrian (4000)	Not-Pedestrian (3000)	7000
PennFudanPed	Pedestrian (170)	-	170

5.2 Dataset Preparation

In our working system we have used three dataset. The distribution of datasets for training, testing and validation images.

5.2.1 INRIA Dataset

INRIA dataset contains 5203 images in total where 3633 images for training and 1570 for testing. We have divided the training images into training-set and validation-set. In training-set contains 2422 images where 1610 images contain pedestrian and 810 images contain no-pedestrian. In testing it contains 1120 pedestrian and 450 no-pedestrian. Some of the sample images of INRIA dataset in shown in Figure 5.1

Figure 5.1: Sample Images Of INRIA Dataset

The images distribution of INRIA dataset is given in Figure 5.2

5.2.2 Daimler Mono Pedestrian Dataset

In Daimler Mono Pedestrian Classification Benchmark Dataset contains 22404 images but we working with 7000 images from this dataset. We have used 5000 images for training and 2000 images for testing. We have divided the training images into to part that is taininget and validationset. In trainingset it contains 3334 images where 2000 images are pedestrian and 1334 images are non-pedestrian. In validationset it contains 1666 images where 1000 images pedestrian and 666 are non-pedestrian. In testing phase there are 2000 images where 1000 images are pedestrian and 1000 images are non-pedestrian. Some of the sample images of Daimler Mono Pedestrian Dataset in shown in Figure 5.3

The images distribution of Daimler Mono Pedestrian Dataset is given in Figure 5.4

5.2.3 Daimler and INRIA dataset

Here we have used Daimler Mono Pedestrian Dataset and INRIA dataset together for training and testing. The total image we have used is 12203 where we have used 8633 images for training and 3570 images for testing. We have then divided the training images into training-set and testing-set. In training-set it contains 5756 images where 3610 images are pedestrian and 2146 images are non-pedestrian. In validation-set it contains 2877 images where 1805

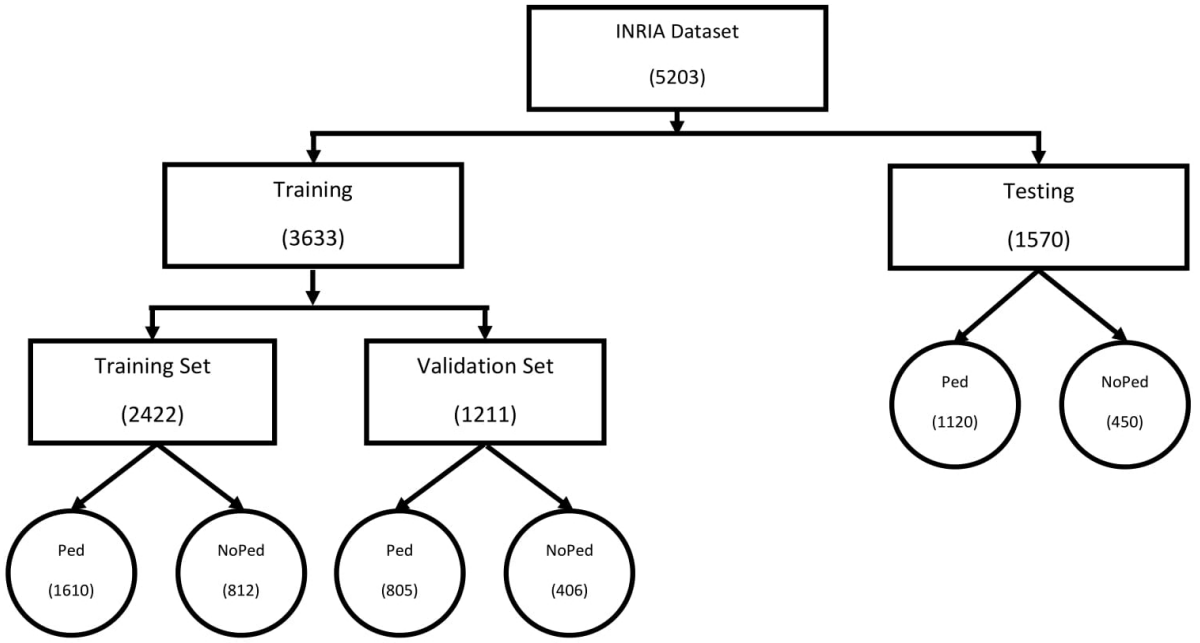


Figure 5.2: Distribution Of Images Of INRIA Dataset

Figure 5.3: Sample Images Of Daimler Mono Pedestrian Dataset

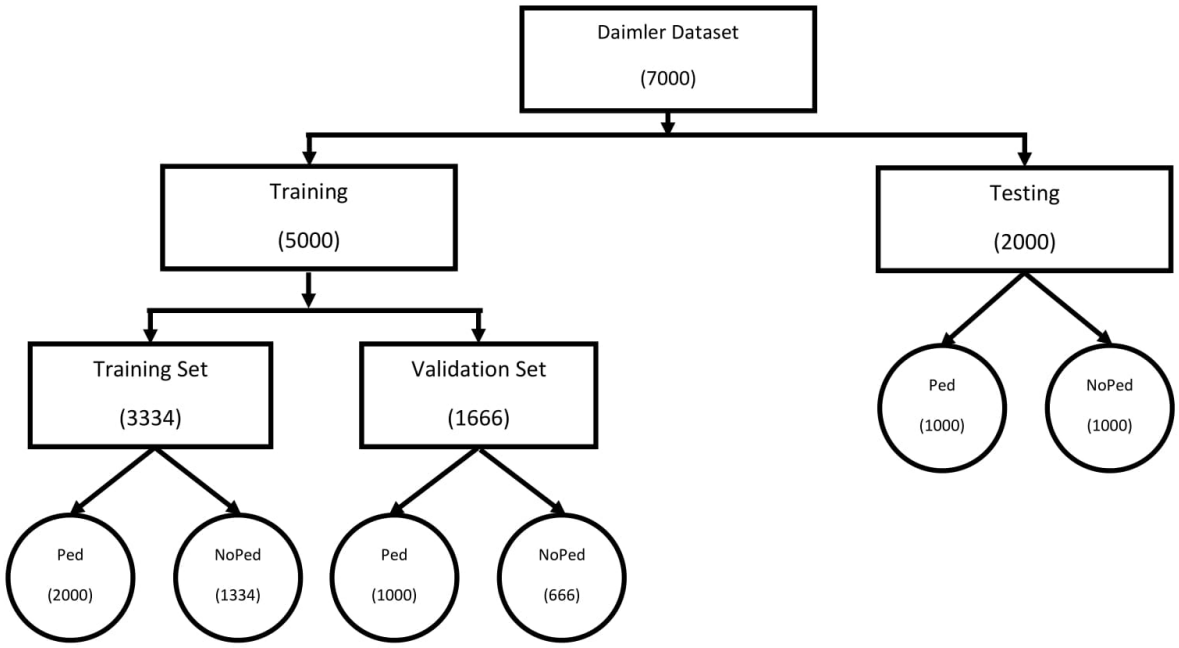


Figure 5.4: Distribution Of Images Of Daimler Mono Pedestrian Dataset

images are pedestrian and 1072 images are non-pedestrian. The images distribution of Daimler and INRIA dataset is given in Figure 5.5

5.3 Preprocessing Result

5.3.1 Gaussian Filter

We have resized the images of all datasets mentioned in Table 5.1 into 140 x 300. Then we have applied Gaussian filter. The output of Gaussian filter is given in Figure 5.6

5.4 Feature Extraction Result

5.4.1 Gabor Filter

We have applied Gabor filter on Gaussian filtered image. Gabor filter is used for extracting highly oriented features. The output form of Gabor filtered image is shown in Figure 5.7

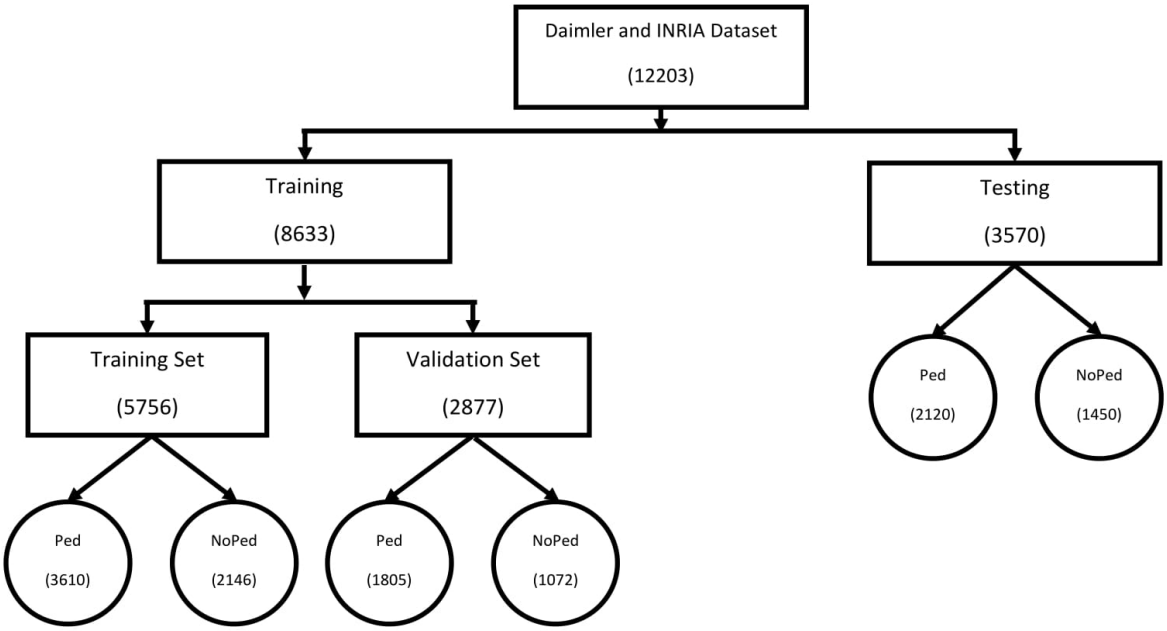


Figure 5.5: Distribution Of Images Of Daimler and INRIA Dataset

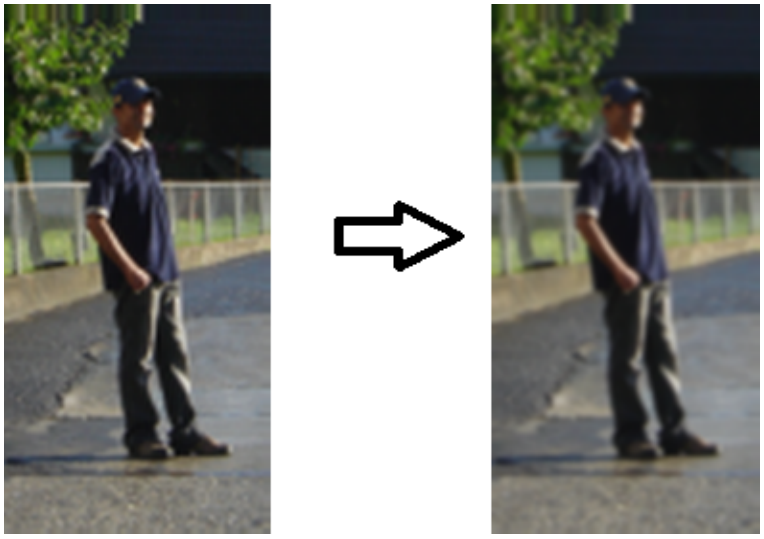


Figure 5.6: Output Image after applying Gaussian Filter

Figure 5.7: Output image after applying Gabor Filter

5.4.2 Histogram of Oriented Gradients (HOG)

HOG detects the direction when pixels intensity enters from high to low and then draws the direction. The output form of HOG image is shown in Figure 5.8

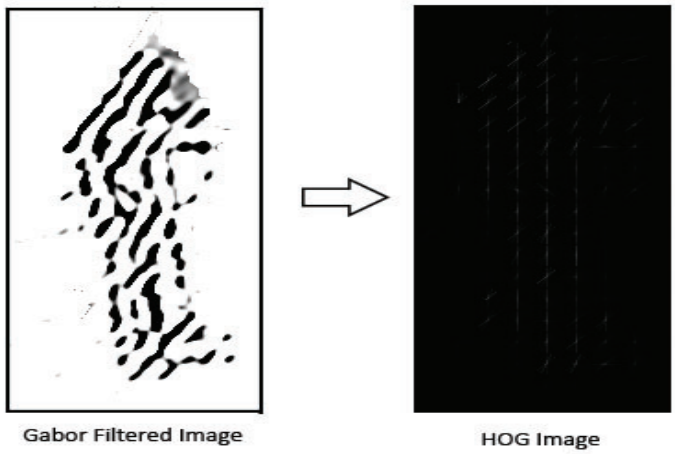


Figure 5.8: Output image after applying HOG

5.4.3 HOGG (Histogram of Oriented Gradients and Gabor Filter)

After applying gaussian filter in the image we have applied gabor filter and then HOG method. This is the method of HOGG. The resulting output in given in Figure 5.9

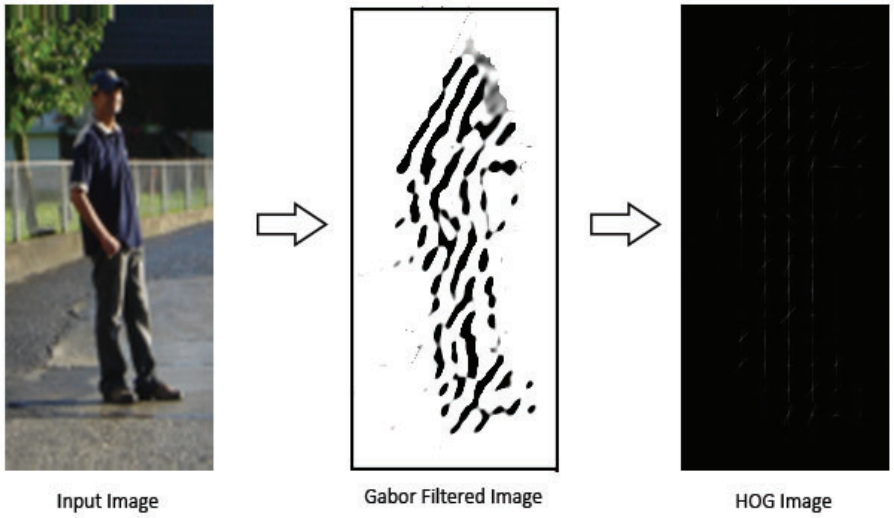


Figure 5.9: Output image after applying HOGG

5.5 Accuracy Calculation

For calculating the accuracy, precision, recall first we have took the true positive, true negative, false positive and false negative from the confusion matrix. Then we have calculated accuracy, precision, recall using the confusion matrix. For three dataset and for each dataset we have got different confusion matrix.

Accuracy means the number of correct results from the whole dataset divided by all sample.

$$accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalseNegative + FalsePositive + TrueNegative} \tag{5.1}$$

Error rate means the number of wrong results from the whole dataset divided by all sample.

$$errorrate = \frac{FalsePositive + FalseNegative}{TruePositive + FalseNegative + FalsePositive + TrueNegative} \tag{5.2}$$

Precision means the number of correct results (or true positives, where, a pedestrian who is detected as a pedestrian) divided by the number of all positive results (that is, the sum of pedestrians and non- pedestrians detected as pedestrians). Precision shows the probability that a retrieved object was a pedestrian [44].

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5.3}$$

Recall means the number of correct results divided by the sum of pedestrians detected as pedestrians, and pedestrians not detected as pedestrians . Recall shows the probability that a pedestrian was retrieved [44].

$$recall = \frac{TruePositive}{TurePositive + FalseNegative} \tag{5.4}$$

The F-measure measures the accuracy of a test by considering both the Precision and the Recall. F-measure is very useful to avoid unbalanced systems[44].

$$F - score = \frac{2 * precision * recall}{precision + recall} \tag{5.5}$$

5.5.1 INRIA Dataset

We have run CNN model several times with changes of convolutional layer, convolutional layer feature map size, hidden layer, input shape, number of neurons and we have got three best result according to the INRIA dataset. After testing with the dataset we have got a confusion matrix given in Figure 5.10

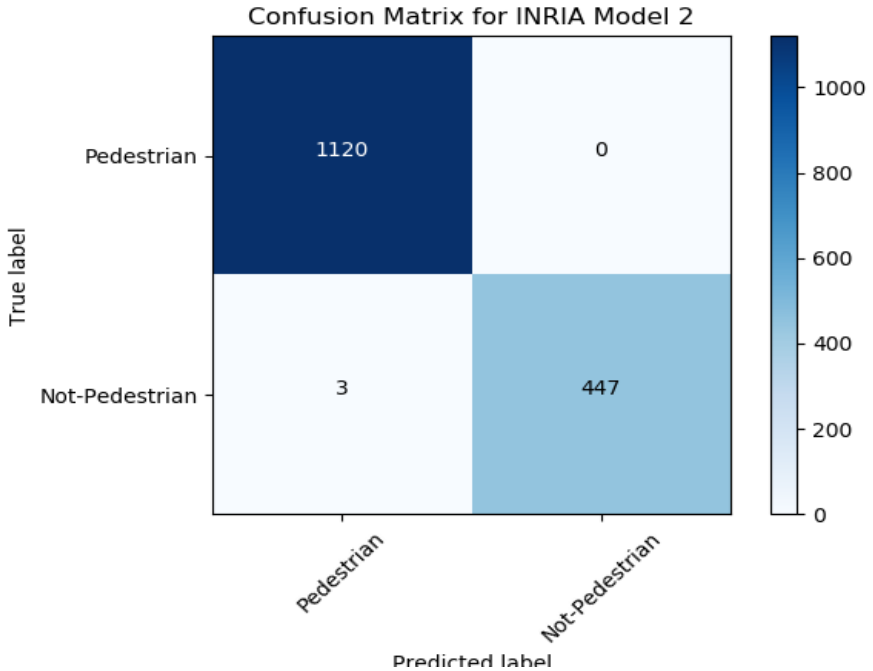


Figure 5.10: Confution Matrix for Model 2 using INRIA Datasaet

The data for training and testing is given in Table 5.2

Table 5.2: Accuracy Measure for INRIA Dataset

Model	Input Shape	Convolutional Layer	Hidden Layer	Precision	Recall	Accuracy
1	64x64	1-32(3x3)	1-128	99.56 %	100 %	99.68 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
2	64x64	1-32(3x3)	1-128	99.73 %	100 %	99.81 %
		1-64(3x3)	1-128			
		1-128(3x3)	1-128			
3	64x64	1-32(3x3)	1-128	98.68 %	100 %	99.04 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
			1-128			
			1-128			

5.5.2 Daimler Dataset

We have also run CNN model several times and get three best result. We have also changed the CNN parameter to get better accuracy. The confusion matrix in given in Figure 5.11

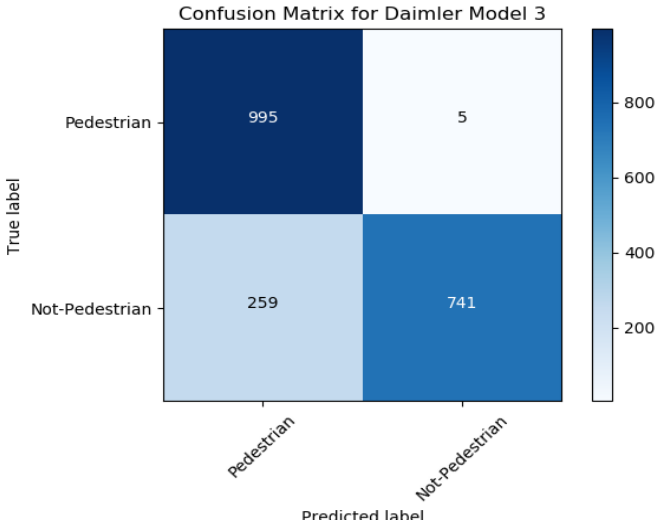


Figure 5.11: Confution Matrix for Model 2 using Daimler Dataset

The data for training and testing is given in Table 5.3

Table 5.3: Accuracy Measure for Daimler Dataset

Model	Input Shape	Convolutional Layer	Hidden Layer	Precision	Recall	Accuracy
1	64x64	1-32(3x3)	1-128	81.16 %	89.6 %	84.4 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
2	64x64	1-32(3x3)	1-128	79.35 %	99.5 %	86.8 %
		1-64(3x3)	1-128			
		1-128(3x3)	1-128			
3	64x64	1-32(3x3)	1-128	100 %	53.7 %	76.85 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
			1-128			
			1-128			

5.5.3 Daimler and INRIA Dataset

We have combined the two dataset Daimler and INRIA then we have run CNN model several times and we have taken best three model. The confusion matrix of our best model is given in 5.12

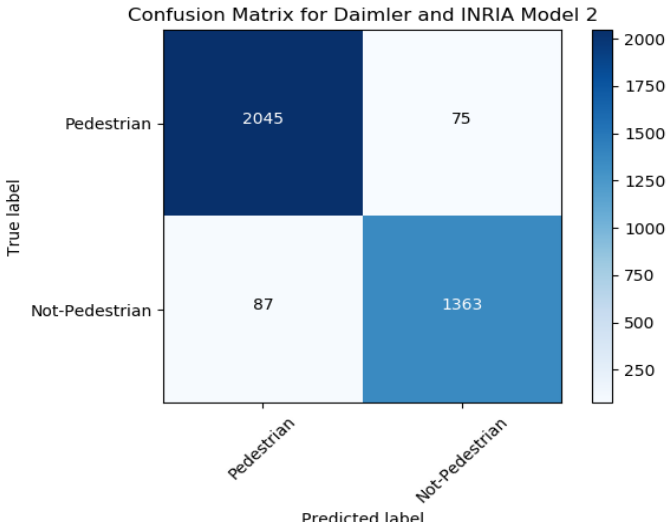


Figure 5.12: Confution Matrix for Model 2 using Daimler and INRIA Dataset

he data for training and testing is given in Table 5.4

Table 5.4: Accuracy Measure for Daimler and INRIA Dataset

Model	Input Shape	Convolutional Layer	Hidden Layer	Precision	Recall	Accuracy
1	64x64	1-32(3x3)	1-128	97.66 %	84.76 %	89.74 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
2	64x64	1-64(3x3)	1-128	95.92 %	96.46 %	95.46 %
		1-32(3x3)	1-128			
		1-128(3x3)	1-128			
3	64x64	1-32(3x3)	1-128	98.07 %	88.82 %	92.32 %
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
		1-32(3x3)	1-128			
			1-128			
			1-128			

5.6 Comparison

The objectives of HOGG-CNN techniques are to improve detection accuracy and compare with existing techniques and methodology. In our research we have worked with INRIA and Daimler dataset using Gabor filter, Histogram of Oriented Gradients and CNN. For all three model we have run our CNN several times and take the three best results. We have run all of our CNN model in Intel(R) Core(TM) i5-4200M CPU with 2.50 GHz and 4GB RAM. We have implemented the system in Python. For these purpose we have used OpenCV for preprocessing and feature extraction and keras and tensorflow for training and testing the model. We have achived some result from our model and we have compraed with the other existing method that is given in 5.5

Table 5.5: Comparison with Existing Works

Dataset	Work Reference	Approach	Accuracy	Precision	Recall	Remarks
INRIA	Sun et. al. [48]	HOG and SVM	-	82%	63%	
	Yamashita et. al. [7]	DCNN	94.5%	-	-	
	Li et. al. [1]	R-CNN	91.06 %	-	-	
	Liu et. al. [2]	DNN	96.5 %	-	-	
	Present Work	**HOG + Gabor and CNN**	**99.81 %**	**99.73%**	**100%**	**Best**
Daimler	Dominguez et. al. [9]	HOG and Resnet	79%	-	-	
	Nilsson et al. [3]	HOG and SVM	81 %	-	-	
	Martelli et. al.[49]	HOG and SVM, LSVM	96.82 %	-	-	
	Present Work	**HOG + Gabor and CNN**	**86.8 %**	**99.5%**	**79.35%**	

Summary:

In this chapter, we have shown various calculation with necessary equations. We have also shown the accuracy, precision and recall table. We have discussed about the datasets we used and their distribution in training and testing phase. We have drawn the confusion matrix for the different model on different datasets and showed the comparison among related works.

Chapter 6

Conclusion and Future Direction

6.1 Conclusion

In our thesis we have applied Gaussian filter for removing noise and HoGG (Gabor + HoG) is used for extracting features. We have used Convolutional Neural Network as a classifier which very much efficient on Computer Vision field. Our main goal is to increase the detection accuracy compared to existing methods. We have used INRIA, DAIMLER Mono Pedestrian Classification Benchmark and PennFudanPed datasets for training and testing purposes. These datasets are mostly used dataset in pedestrian detection sector. We have obtained 99.81% accuracy on INRIA dataset and 86.8% accuracy on DAIMLER Mono Pedestrian Classification Benchmark dataset. Our model showed comparatively good accuracy in INRIA dataset as compared to DAIMLER Mono Pedestrian Classification Benchmark and PennFudanPed.

6.2 Limitations and Future Direction

Our proposed structure can not give satisfactory result on DAIMLER Mono Pedestrian Classification Benchmark dataset. We have run all of our CNN model in Intel(R) Core(TM) i5-4200M CPU with 2.50 GHz and 4GB RAM, for this reason execution time is too much higher. If we use more powerful device with highly configured GPU and RAM, execution time will be lessen and we can get more accurate result.

In future, we will work on pedestrian position detection in an frame using bounding box. We may also work on pedestrian movement detection so that we can develop a robust automated car driving system.

References

[1] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware fast r-cnn for pedestrian detection," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 985–996, 2018.

[2] J. Liu, S. Zhang, S. Wang, and D. N. Metaxas, "Multispectral deep neural networks for pedestrian detection," *arXiv preprint arXiv:1611.02644*, 2016.

[3] J. Nilsson, P. Andersson, I. Y. Gu, and J. Fredriksson, "Pedestrian detection using augmented training data," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 2014, pp. 4548–4553.

[4] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal processing: image communication*, vol. 47, pp. 482–489, 2016.

[5] A. Dimou, P. Medentzidou, F. Álvarez García, and P. Daras, "Multi-target detection in cctv footage for tracking applications using deep learning techniques," in *2016 IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP)*. IEEE, 2016, p. 932.

[6] A. Uçar, Y. Demir, and C. Güzeliş, "Object recognition and detection with deep learning for autonomous driving applications," *Simulation*, vol. 93, no. 9, pp. 759–769, 2017.

[7] T. Yamashita, H. Fukui, Y. Yamauchi, and H. Fujiyoshi, "Pedestrian and part position detection using a regression-based multiple task deep convolutional neural network," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 3500–3505.

[8] H. Fukui, T. Yamashita, Y. Yamauchi, H. Fujiyoshi, and H. Murase, "Pedestrian detection based on deep convolutional neural network with ensemble inference network," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015, pp. 223–228.

[9] A. Dominguez-Sanchez, M. Cazorla, and S. Orts-Escolano, "Pedestrian movement direction recognition using convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3540–3548, 2017.

[10] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "How far are we from solving pedestrian detection?" in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] D. Moctezuma, C. Conde, I. M. de Diego, and E. Cabello, "Person detection in surveillance environment with hogg: Gabor filters and histogram of oriented gradient," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1793–1800.

[12] R. D. Boyle and R. C. Thomas, *Computer vision: A first course*. Blackwell Scientific Publications, Ltd., 1988.

[13] E. Davies, "Machine vision: Theory, algorithms and practicalities. 1990," *Google Scholar*, 1997.

[14] D. Vernon, "Machine vision-automated visual inspection and robot vision," *NASA STI/Recon Technical Report A*, vol. 92, 1991.

[15] P. Lei, "Adaptive median filtering," in *Seminar Report, Machine Vision*, vol. 140, 2004.

[16] B. Horn, B. Klaus, and P. Horn, *Robot vision*. MIT press, 1986.

[17] R. M. Haralick and L. G. Shapiro, *Computer and robot vision*. Addison-wesley Reading, 1992, vol. 1.

[18] R. Gonzalez, "R. woods digital image processing," *Reading, MA: Addison-Welsley*, 1992.

[19] A. D. Jepson and M. R. Jenkin, "The fast computation of disparity from phase differences," in *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR'89., IEEE Computer Society Conference on*. IEEE, 1989, pp. 398–403.

[20] T. D. Sanger, "Stereo disparity computation using gabor filters," *Biological cybernetics*, vol. 59, no. 6, pp. 405–418, 1988.

[21] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[22] T. Ojala and M. Pietikäinen, "Unsupervised texture segmentation using feature distributions," *Pattern recognition*, vol. 32, no. 3, pp. 477–486, 1999.

[23] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[24] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910.

[25] B. Lee, "Application of the discrete wavelet transform to the monitoring of tool failure in end milling using the spindle motor current," *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 4, pp. 238–243, 1999.

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

[27] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

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

[29] T. Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[30] K. Ikeuchi, *Computer vision: A reference guide*. Springer Publishing Company, Incorporated, 2014.

[31] T. Lindeberg and L. Bretzner, "Real-time scale selection in hybrid multi-scale representations," in *International Conference on Scale-Space Theories in Computer Vision*. Springer, 2003, pp. 148–163.

[32] J. L. Crowley and O. Riff, "Fast computation of scale normalised gaussian receptive fields," in *International Conference on Scale-Space Theories in Computer Vision*. Springer, 2003, pp. 584–598.

[33] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *null*. IEEE, 2003, p. 734.

[34] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101.

[35] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.

[36] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

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

[39] "Convolutional neural networks for visual recognition," http://cs231n.github.io/convolutional-networks/, accessed: 2018-01-08.

[40] "Support vector machines," http://www.statsoft.com/Textbook/ Support-Vector-Machines, accessed: 2018-01-08.

[41] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] "Artificial neural networks," http://www.psych.utoronto.ca/users/ reingold/courses/ai/cache/neural2.html, accessed: 2018-01-08.

[44] C. Conde, D. Moctezuma, I. M. De Diego, and E. Cabello, "Hogg: Gabor and hog-based human detection for surveillance in non-controlled environments," *Neurocomputing*, vol. 100, pp. 19–30, 2013.

[45] F. Guojuan, L. Bo, M. Wanquan, and L. Xuqing, "Hogg: Gabor and hog-based human detection," in *Information Technology in Medicine and Education (ITME), 2016 8th International Conference on.* IEEE, 2016, pp. 562–566.

[46] "Inria person dataset," http://pascal.inrialpes.fr/data/human/, accessed: 2018-02-10.

[47] "Daimler mono pedestrian classification benchmark," http://www.lookingatpeople.com/download-daimler-ped-class-benchmark/index.html, accessed: 2018-02-15.

[48] Y. Sun, Z. Liu, and B. Ding, "Multi-pedestrian detection in crowded places on quadrotor's view," in *Intelligent Robot Systems (ACIRS), 2017 2nd Asia-Pacific Conference on.* IEEE, 2017, pp. 62–65.

[49] S. Martelli, M. San Biagio, and V. Murino, "Latent subcategory models for pedestrian detection with partial occlusion handling," in *Advanced Video and Signal Based Surveillance (AVSS), 2015 12th IEEE International Conference on.* IEEE, 2015, pp. 1–6.