M.Tech Signal Processing And Digital Design Megha Kataria 2019

PEDESTRIAN WALKING DIRECTION DETECTION USING HYBRID CNN-SVM MODEL

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SIGNAL PROCESSING AND DIGITAL DESIGN

Submitted by:

MEGHA KATARIA

2K17/SPD/07

Under the supervision of

DR. RAJESH ROHILLA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

MAY, 2019

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Megha Kataria, 2K17/SPD/07, student of M.Tech. Signal Processing And Digital Design, hereby declare that the project Dissertation titled "Pedestrian Walking Direction Detection Using Hybrid Convolutional Neural Network – Support Vector Machine Model" which is submitted by me to the Department of Electronics And Communication , Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **MEGHA KATARIA**

Date: 28/07/2019

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "Pedestrian Walking Direction Detection Using Hybrid Convolutional Neural Network – Support Vector Machine Model" which is submitted by Megha Kataria, 2K17/SPD/07, Department Of Electronics And Communications , Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **DR. RAJESH ROHILLA**

Date: 28/07/2019 **SUPERVISOR**

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my advisor Dr. Rajesh Rohilla for the continuous support of my M.tech thesis, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. You have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a researcher. Your advice on both research as well as on my career have been invaluable**.** I could not have imagined having a better advisor and mentor for my M.tech study.

I would also like to thank my friends for their constant support and motivation throughout and their readiness for an active discussion at all times.

Last but not the least, I would like to thank my family: my parents Mr. Anil Kumar and Mrs. Sanjeeta , for giving birth to me at the first place and supporting me spiritually throughout my life.

(MEGHA KATARIA)

2K17/SPD/07

ABSTRACT

Pedestrian movement direction recognition is an important factor in autonomous driver assistance and security surveillance systems. Pedestrians are the most crucial and fragile moving objects in streets, roads, and events, where thousands of people may gather on a regular basis. People flow analysis on zebra crossings and in shopping centers or events such as demonstrations are a key element to improve safety and to enable autonomous cars to drive in real life environments. This thesis focuses on deep learning techniques such as hybrid Convolutional Neural Networks (CNN) – Support Vector Machine (SVM) model to achieve a reliable detection of pedestrians moving in a particular direction. We propose a CNN-based technique that leverages current pedestrian detection techniques (histograms of oriented gradients-linear SVM) to generate a sum of subtracted frames (flow estimation around the detected pedestrian), which are used as an input for the hybrid CNN – SVM model.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS , ABBREVIATIONS

CNN: Convolutional Neural Network

SVM: Support Vector Machine

ADAS: Autonomous Driver Assistance Systems

GPUs: Graphics Processing Units

WHO: World Health Organization

ACC: Adaptive Cruise Control

HOG: Histogram of Oriented Gradients

ANN: Artificial Neural Networks

KF: Kalman Filters

ReLU: Rectified Linear Unit

CHAPTER 1. INTRODUCTION

Urbanization, and in particular the emergence of megacities, reflects the growth in the global population and economy. Mobility and transport form the backbone for society’s increasing prosperity and people’s greater participation in it. At the same time, managing traffic in these urban agglomerations represents a special challenge because more and more people live not only in the towns themselves, but also in the surrounding areas, and they commute between home and work every day. The towns continue to expand, and urban agglomerations themselves turn into cities

In developing countries and emerging economies in particular, many towns and cities are growing so fast that infrastructural measures and urban planning can hardly keep up. The most visible consequence is long traffic queues and too few parking spaces. This means that time is lost and it also annoys the drivers. If the traffic does not flow, trade and productivity also suffer, which has negative effects on the economy.

Growth and prosperity and participation in them require people and goods to be mobile. We will need innovative solutions for secure mobility and transport to act as the motors driving the future. It is no longer possible simply to “carry on as before.” In view of population growth and urbanization, traffic flows, which are precisely what up to now has been the basis for greater prosperity for more people, would be restricted. Automated driving has the potential to contribute to resolving the challenges accompanying global development trends. The primary objective is to make road traffic even safer. The challenge lies in accomplishing this while the volume of traffic continues to rise, especially in the “megacities.

The technology in driver assistance systems potentially offers additional significant reductions in the number of accidents and traffic jams. Adaptive cruise control systems, for example, improve the traffic flow and make a major contribution to avoiding accidents. Automated driving makes traffic not only safer, but also more efficient and comfortable. Optimized traffic flow and less congestion bring about a decisive reduction in CO2 emissions. During rush hour in particular, drivers of automated passenger cars and commercial vehicles gain a newly won freedom and enjoy a better quality of driving. Furthermore, they can leave parking to their vehicles. In short, automated driving functions support drivers and offer them greatly improved driving comfort and flexibility. Yet despite all this progress in automation, drivers remain in control. Just as today they can already decide whether they would like to have an assistance system in their vehicle, they can also decide while driving whether they want to use an assistance system that has been installed, such as adaptive cruise control.

TRAFFIC control, risk detection and Autonomous Driver Assistance Systems (ADAS) are key elements for the development of future intelligent transportation systems. Furthermore, dynamic pedestrian movement in traffic environments makes it necessary to develop people flow analysis and movement intention recognition systems. In recent years, Convolutional Neural Networks (CNN) and other deep learning techniques have demonstrated impressive performance in many computer vision problems and therefore we believe they could be the perfect approach for the aforementioned problems. Moreover, computer vision and machine learning techniques have been transformed due to the rapid evolution and remarkable performance of Graphics Processing Units (GPUs), which has enabled the development of deep learning-based systems. In this work, our objective is the detection and recognition of pedestrian intention on streets, zebra crossings or road junctions, so as to be able to alert drivers or monitoring systems about possible risk situations.

1.1 NEED FOR PEDESTRIAN MOVEMENT DETECTION

1.1.1 PEDESTRIAN SAFETY

With rapid economic development, the use of automobiles has greatly increased in developing countries, especially in China, India, and Vietnam, where vehicles are replacing bicycles as the dominant transportation mode. Facing this great change, the space allocated to automobiles has been expanded, thus alleviating traffic congestion, which encroaches on the space for cyclists and pedestrians and constrains bicycling and walking. Consequently, potential conflicts of vehicles, cycles, and pedestrians not only exacerbate travel delays but also increase the randomness of pedestrian movements, substantially threatening pedestrian safety. In a recent traffic safety report released by the World Health Organization (WHO), road collisions are the world’s leading cause of preventable death; over 1.25 million people die annually on the roads (especially at intersections) because of traffic collisions . In some ways, active transportation users, such as cyclists and pedestrians, are more vulnerable to injuries than other road users due to their labile speed and direction . Therefore, a walkable city for people is regarded as one ultimate goals of future cities . Urban planners have taken several actions to encourage walking, such as configuring special walkable lanes, designing good walking interfaces, building friendly walkable infrastructures, etc. These actions also highlight the necessity of pedestrian tracking and simulation.

Traditionally, pedestrian safety inspection largely relies on historical collision records. However, due to the lack of detailed and precise historical data and the infrequent occurrence of collisions, the task of inspection often cannot be fully accomplished. Recently, the use of pedestrian conflicts as an alternative for collisions to analyze road safety has attracted significant interest . Pedestrian conflicts can provide detailed information concerning road dynamics at intersections, allowing the detection of the series of events that lead to collisions . Pedestrian conflict analysis can be conducted by detecting and tracking moving traffic objects or flawed design elements that may be causing safety issues . The introduction of computer vision algorithms has greatly strengthened pedestrian conflict and violation analysis by automating the extraction of accurate movements of traffic objects, overcoming many shortcomings of manual pedestrian analysis techniques .

1.1.2 ADVANCED DRIVER ASSISTANCE SYSTEM (ADAS)

Today a large number of driver assistance systems is available for almost all vehicles. They ensure stability in critical situations, maintain a safe distance to the vehicle in front, and support the driver while parking. Monitoring the surroundings in all directions requires data and information from the vehicle’s sensors (ultrasound, radar, cameras). The capabilities of the sensors and the data processing by the control units are continually growing, and highly advanced software is used to analyze this information in fractions of a second. In the future, passenger cars and commercial vehicles will have a complete image of the surroundings in real time.

Radar sensors that are usually located in the front and rear of the vehicle can detect other vehicles and obstacles. The rear sensor detects traffic approaching from behind and vehicles that are overtaking. The traffic in front is monitored by longrange radar. The short-range radar surveys the vehicle's immediate surroundings. Cameras are used, for instance to recognize lane markings, traffic signs, traffic lights, and other road users. Ultrasound sensors have been installed in vehicles from the beginning of the nineties, to help drivers maneuver into parking spaces. Since then, their range of functions has increased markedly. They can measure parking spaces while the vehicle is in motion, and detect vehicles driving in an adjacent lane

In the past, radar, cameras, and ultrasound sensors were used for separate functions, but now all the relevant data can be linked intelligently and simultaneously by sensor fusion. That makes automated driving possible in the first place. Special attention is paid to functional safety.

The inclusion of redundancies and plausibility checks – that is, the system's internal check on whether the environmental data have been recorded correctly – prevents erroneous interpretation of the data. The signals from the vehicle sensors are compared with one another. Only if the data are consistent will the system actuate the steering and the engine.

The automated driving functions include "highway driving," which in the case of highly automated driving will be used up to a defined speed on highways and similar roads. The driver can choose when to activate the system and does not have to monitor it continuously. This takes away some of the stress of driving, and in certain situations they will be prompted in good time to resume the task of driving. In the case of fully automated driving, the driver does not have to monitor the system at all. In the distant future, in built-up areas the driving function "urban driving" will make it possible to drive on various routes without the driver intervening at all. In this case the driver will be free to use the time on the road as they choose

DEVELOPMENTS IN COMMERCIAL VEHICLE TECHNOLOGY

The information and functions both of tried-and-tested and of future driver assistance systems are being bundled into an overall system in commercial vehicles just as in passenger cars. The systems include adaptive cruise control (ACC), the lane keeping assistant, and the emergency braking system – to name but a few examples. Then there are also innovations such as digital 3-D maps. With their aid, the vehicle's handling is adapted to the features of the road immediately ahead. So a truck can accelerate while approaching an incline in order to build up momentum and ultimately reach the brow of the hill more economically. These systems can very easily be expanded by car-to-X communication. Sharing information with other road users results in additional improvements in safety and efficiency.

Automated driving functions are especially attractive to fleet operators. Fuel consumption and emissions fall considerably because the traffic can flow more evenly. This means that higher average speeds are possible without the need to increase the top speed. Transport times are more predictable and there is less wear on the engines in trucks using the new features, owing to the smoother driving style.

Relieving the stress on drivers is an important factor in further development. Today's truck drivers are subject to extreme demands. When driving in very dense traffic, they have to remain attentive at all times and are often under time pressure. In the more distant future they will be able to rely completely on the technological systems in their truck, and the truck itself will drive to its destination safely and efficiently thanks to its sensor systems and the sharing of data with its surroundings. This will make a lot of things easier for the drivers, who will be able to turn their attention to other tasks, such as flexible organization of the current route or planning future journeys.

1.1.3 LAND USE AND TRANSPORTATION PLANNING

The study of pedestrian movement is crucial for land-use and transportation planning which mostly concentrates on improving the connection between urban places and public transportation. Land use planning influences pedestrian movement behaviour in terms of the element of accessibility that is controlled by urban structure, activities, and street networks, all of which make different cities display the unique urban forms. This is especially the case in the high-density areas such as a central business district or a major transit station district where urban form is planned to support the use of land and potential accessibility between people and places.

Urban structure plays a key role in providing available paths for pedestrian flows through urban areas. Public spaces, sidewalks, and street crossings all influence the direction of crowd movement along with the surrounding conditions that have an impact on people making decisions on which access path they select. Pedestrians create their own path to reach their desired destination through their own choices of transit access routes, which are generated by their estimation and perception of the quickest route whilst also considering secondary factors that include the surroundings of the built-up environment such as the attractiveness of facilities while avoiding negative features. Urban network analysis is useful for describing the interaction between urban structures and street networks which leads to the impact prediction on the project evaluation. The computer analysis is available for the transportation planning on providing important spatial information which is precise data on spatial structure that enables urban planners to see the whole picture of the planning area as well as to understand the impact on both existing and future structures that might be assigned as a result of future policy.

Survey methods used to identify pedestrian movement characteristics are questioned in terms of their ability to obtain precise data for proceeding to the next step of spatial analysis. As the development of data technology assists spatial survey methods with lower financial and time costs, surveyors or analysts nowadays are able to conduct data collection processes via their handheld device. Although counting the number of people passing by particular area is considered as a traditional method for the study of transportation study, there are not always definitions of precise pedestrian movement or behaviour due to limitations in the data collection process. The integration of data surveying methods which are able to solve such limitations by relying on more accessible devices and computer software are needed to improve these collection methods for tracking the actual walking movement which seems to be more precise in terms of identifying how pedestrians actually react to the pedestrian infrastructure within specific land-use conditions.

1.2 LITERATURE REVIEW

Until 2012, most recognition, segmentation and classification image problems were approached by extracting hand designed features and applying specific algorithms for those particular features. For example, if a number plate on a car needed to be detected, we segmented the image by looking for straight lines, then corners and finally the image was reduced until we had an area similar to the geometry of a number plate. In essence, we looked for the particular features that could solve a specific problem.

A common hand-crafted feature used for pedestrian detection is the Histogram of Oriented Gradients (HOG) . The main idea behind this descriptor is that local object appearance and shape within an image can be described by the intensity distribution of gradients or edge directions. The image is divided into small connected areas, and for the pixels within each area, a histogram of gradient directions is generated.

Recent work in this area has added a local sub-descriptor called Colour Self Similarity (CSS) where colour histograms are compared within a HOG detected window, and for example, colour histograms from the two arms have a high similarity.

In addition, extensive research has been done on pedestrian detection , where more than sixteen different detectors were benchmarked against several public datasets. The features were mainly based on window-sliding techniques and detection was performed using support vector machines (SVM) for classification. Moreover, other approaches based on the Adaboost work of Viola and Jones , and many others based on HOG and variants of the same method.

Since 2012, new approaches for pedestrian detection and related problems have emerged with the advent of deep learning techniques. Deep learning is a new way of applying machine learning algorithms, where neural networks are being made deeper and deeper by the addition of tens, or even hundreds, of layers. Specifically, in computer vision, much work was done in this regard before 2012, using multi-layer neural networks but obtaining poor results. Recognition of characters was conducted using a CNN with a deeper layer structure. However, it was after 2012, with the proposal of Alex Krizhevsky CNN, AlexNet, when the real capabilities of CNNs became clear. These methodologies were first used at the Imagenet Competition where the novel techniques, of deep multi-layer neural networks, were accelerated using GPUs. Since then, new and better hardware has appeared. This increases the possibility of bigger and deeper CNNs, providing better classification accuracy and making the training of existing deep networks an affordable scientific tool in terms of training time.

Computer vision research groups focused on pedestrian detection have also benefited from the rise of CNNs, and recent analyses have proved that better and more reliable results can be achieved . However, our work focuses, not just on pedestrian detection, but also on pedestrian movement direction recognition, analyzing, for example, whether the pedestrian moved to the left, right or to the front of the scene. There are few studies in this area. Enzweiler and Gavrila and Gandhi and Trivedi focused on that aspect using the HOG descriptor and SVM as a classifier while Mogelmose et al. used pedestrian tracking techniques and trajectory analysis for estimating pedestrian direction.

In general, the estimation of pedestrians' trajectories have traditionally been addressed using naive movement models based on human gait estimation and analysis of simple heuristics based on that information . Other traditional approaches have focused on the use of Kalman Filters (KF) to estimate pedestrian trajectories. Most of these existing techniques produced poor results due to the impossibility of properly handling and adapting to changes in pedestrians' movements . More recently, a more complex method based on Artificial Neural Networks (ANN) has been proposed for pedestrian trajectory estimation and intention recognition . This work is able to estimate pedestrian trajectory based on pedestrian head detection and the use of its position for tracking along the sequence. Other existing works in the literature make use of features extracted from a dense optical compensated with ego-motion techniques (car movement) . Using this approach, they are not only able to estimate a pedestrian's path but also to roughly estimate pedestrian intentions towards specific situations such as crossing at intersections .

Finally, it is worth mentioning the existence of related works addressing this problem from a different perspective. Most of these works are based on the information gathered by inertial measurement units (IMUs) and similar technologies (accelerometers, gyroscopes, etcetera). These types of approaches are very intrusive from the pedestrian viewpoint and do not provide enough information to distinguish between different pedestrians' actions. After reviewing state-of-the-art techniques we can conclude that even though in recent years great progress has been made in pedestrian recognition systems, more research is still required on systems and new techniques that can provide better classification accuracy, improved performance and ease of integration in current ADAS and security surveillance systems.

1.3 ORGANIZATION OF THE DISSERTATION

In this work, we contribute to the literature on pedestrian walking direction recognition with the proposal of a hybrid CNN- SVM based system. The CNNs have been trained with a novel dataset that was recorded in different scenarios. Pedestrians were video recorded and the CNNs were fed with output images produced as a result of several image operations at pixel level from this input video. The main purpose of this additional image processing was to visually highlight image characteristics that may be relevant for pedestrian trajectory recognition. Then the features were extracted from the last layer of the CNN architecture and the SVM was trained using these features. This trained SVM is then used to classify the test data.

To the best of our knowledge, not much work has been done on the classification of pedestrians according to their motion direction using deep learning techniques.

The key contributions of our work are as follows:

• We propose a novel pipeline for pedestrian movement direction recognition, which provides high recognition rates in the proposed dataset.

• We have evaluated state-of-the-art hybrid Convolutional Neural Network – Support Vector Machine models for the problem presented and carried out a performance evaluation providing quantitative metrics.

In chapter 2 , we briefly present Convolutional Neural Networks and other techniques like optical flow , histograms of oriented Gradients , Support Vector Machines and multiclass Support Vector Machines that we used in this work.. In Chapter 3 , we describe the dataset we used in our work, and then we describe the proposed CNN based approach for the pedestrian movement direction estimation. Chapter 4 presents the results (accuracy and run time) for the evaluated hybrid CNN Architecture and Chapter 5 draws conclusions and indicates future scope of the work.

CHAPTER 2. TECHNIQUES USED

2.1 CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network (CNN or ConvNet) is one of the most popular algorithms for deep learning, a type of machine learning in which a model learns to perform classification tasks directly from images, video, text, or sound.

CNNs are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They learn directly from image data, using patterns to classify images and eliminating the need for manual feature extraction. Applications that call for object recognition and computer vision such as self-driving vehicles and face-recognition applications rely heavily on CNNs.

Using CNNs for deep learning has become increasingly popular due to three important factors:

- CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.
- CNNs produce state-of-the-art recognition results.
- CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.

CNNs provide an optimal architecture for image recognition and pattern detection. Combined with advances in GPUs and parallel computing, CNNs are a key technology underlying new developments in automated driving and facial recognition.

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object. CNNs perform feature identification and classification of images, text, sound, and video.

2.1.1 LAYERS OF CNN

The first step of creating and training a new convolutional neural network (ConvNet) is to define the network architecture.The network architecture can vary depending on the types and numbers of layers included. The types and number of layers included depends on the particular application or data. A CNN has the following layers:

IMAGE INPUT LAYER :

An image input layer inputs images to a network and applies data normalization. The size of an image corresponds to the height, width, and the number of color channels of that image.

CONVOLUTIONAL LAYER :

A 2-D convolutional layer applies sliding convolutional filters to the input. The convolutional layer consists of various components.

FILTERS AND STRIDES :

A convolutional layer consists of neurons that connect to subregions of the input images or the outputs of the previous layer. The layer learns the features localized by these regions while scanning through an image. For each region while training a dot product of the weights and the input is computed, and then a bias term is added. A set of weights that is applied to a region in the image is called a filter. The filter moves along the input image vertically and horizontally, repeating the same computation for each region. In other words, the filter convolves the input. The step size with which the filter moves is called a stride. The number of weights in a filter is h * w * c, where h is the height, and w is the width of the filter, respectively, and c is the number of channels in the input.

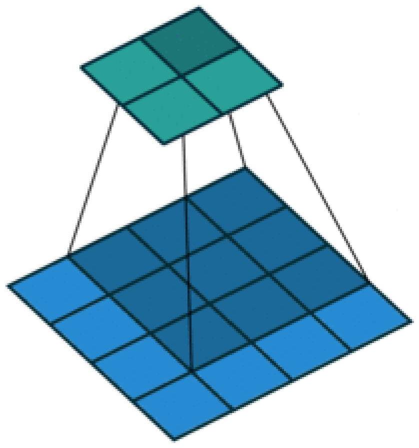


Figure 2.1 : A 3-by-3 filter scanning through the input. The lower map represents the input and the upper map represents the output.[33]

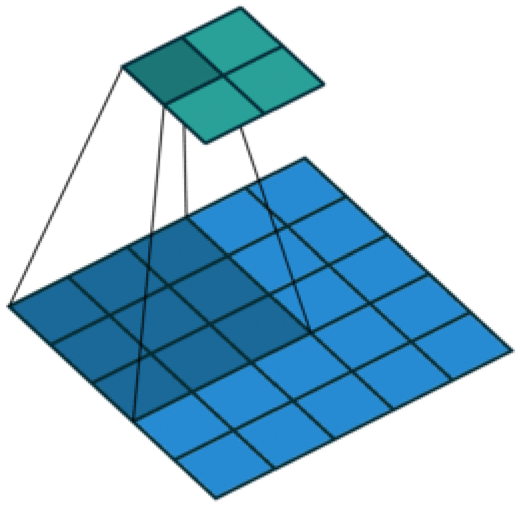


Figure 2.2 : A 3-by-3 filter scanning through the input with a stride of 2. The lower map represents the input and the upper map represents the output.[33]

FEATURE MAPS :

As a filter moves along the input, it uses the same set of weights and the same bias for the convolution, forming a feature map. Each feature map is the result of a convolution using a different set of weights and a different bias. Hence, the number of feature maps is equal to the number of filters. The total number of parameters in a convolutional layer is ((h*w*c + 1)*Number of Filters), where 1 is the bias.

ZERO PADDING :

You can also apply zero padding to input image borders vertically and horizontally. Padding is rows or columns of zeros added to the borders of an image input. By adjusting the padding, you can control the output size of the layer.

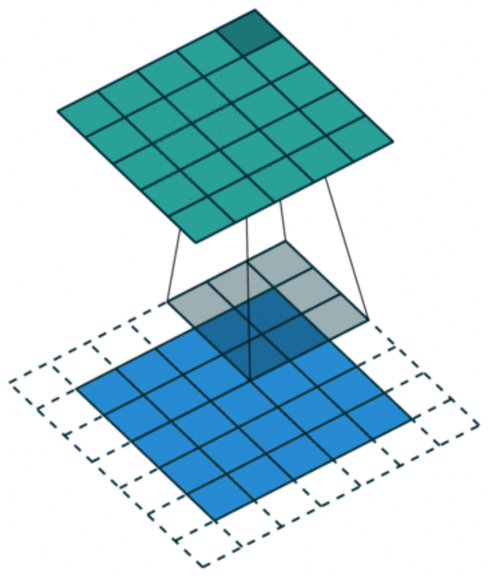


Figure 2.3 : This image shows a 3-by-3 filter scanning through the input with padding of size 1. The lower map represents the input and the upper map represents the output.[33]

OUTPUT SIZE :

The output height and width of a convolutional layer is (Input Size – ((Filter Size – 1)*Dilation Factor + 1) + 2*Padding)/Stride + 1. This value must be an integer for the whole image to be fully covered.

NUMBER OF NEURONS :

The product of the output height and width gives the total number of neurons in a feature map, say Map Size. The total number of neurons (output size) in a convolutional layer is Map Size*Number of Filters.

BATCH NORMALIZATION LAYER :

A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers.

The layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by a learnable offset β and scales it by a learnable scale factor γ. β and γ are themselves learnable parameters that are updated during network training.

Batch normalization layers normalize the activations and gradients propagating through a neural network, making network training an easier optimization problem.

ALGORITHM:

Batch normalization normalizes its inputs x_i by first calculating the mean μ_B and variance σ_B^2 over a mini-batch and over each input channel. Then, it calculates the normalized activations as

$$\widehat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}. \qquad (2.1)$$

Here, ϵ (the property Epsilon) improves numerical stability when the mini-batch variance is very small. To allow for the possibility that inputs with zero mean and unit variance are not optimal for the layer that follows the batch normalization layer, the batch normalization layer further shifts and scales the activations as

$$y_i = \gamma \widehat{x}_i + \beta. \qquad (2.2)$$

Here, the offset β and scale factor γ (Offset and Scale properties) are learnable parameters that are updated during network training.

When network training finishes, the batch normalization layer calculates the mean and variance over the full training set and stores them . When you use a trained network to make predictions on new images, the layer uses the trained mean and variance instead of the mini-batch mean and variance to normalize the activations.

RELU LAYER :

A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero. Convolutional and batch normalization layers are usually followed by a nonlinear activation function such as a rectified linear unit (ReLU), specified by a ReLU layer. A ReLU layer performs a threshold operation to each element, where any input value less than zero is set to zero, that is,

$$F(x) = \left\{ \begin{array}{l} x, x >= 0 \\ 0, x < 0 \end{array} \right\} \qquad (2.3)$$

There are other nonlinear activation layers that perform different operations and can improve the network accuracy for some applications.

MAX AND AVERAGE POOLING LAYERS :

A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region. An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region. Pooling layers follow the convolutional layers for down-sampling, hence, reducing the number of connections to the following layers. They do not perform any learning themselves, but reduce the number of parameters to be learned in the following layers. They also help reduce overfitting. Pooling layers scan through the input horizontally and vertically in step sizes. If the pool size is smaller than or equal to the stride, then the pooling regions do not overlap. For nonoverlapping regions (Pool Size and Stride are equal), if the input to the pooling layer is n-by-n, and the pooling region size is h-by-h, then the pooling layer down-samples the regions by h . That is, the output of a max or average pooling layer for one channel of a convolutional layer is n/h-by-n/h.

FULLY CONNECTED LAYER :

A fully connected layer multiplies the input by a weight matrix and then adds a bias vector .The convolutional (and down-sampling) layers are followed by one or more fully connected layers.

As the name suggests, all neurons in a fully connected layer connect to all the neurons in the previous layer. This layer combines all of the features (local information) learned by the previous layers across the image to identify the larger patterns. For classification problems, the last fully connected layer combines the features to classify the images. This is the reason that the output size of the last fully connected layer of the network is equal to the number of classes of the data set. A fully connected layer multiplies the input by a weight matrix W and then adds a bias vector b.

OUTPUT LAYERS :

SOTMAX AND CLASSIFICATION LAYERS :

A classification layer computes the cross entropy loss for multi-class classification problems with mutually exclusive classes. For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer. For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, trainNetwork takes the values from the softmax function and assigns each input to one of the K mutually exclusive classes using the cross entropy function for a 1-of-K coding scheme

$$\text{loss} = -\sum_{i=1}^{N}\sum_{j=1}^{K} t_{ij} \ln y_{ij}, \tag{2.4}$$

where N is the number of samples, K is the number of classes, t_{ij} is the indicator that the ith sample belongs to the jth class, and y_{ij} is the output for sample i for class j, which in this case, is the value from the softmax function. That is, it is the probability that the network associates the ith input with class j.

2.1.2 CASE STUDIES

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

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).

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.

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.

VGGNet. The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. 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.

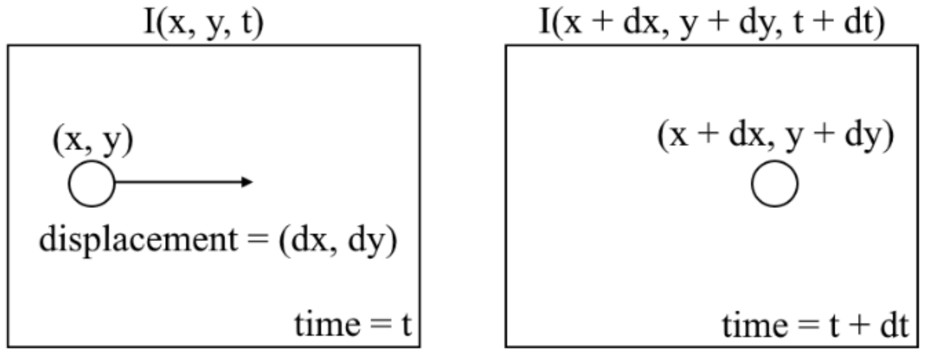
ResNet. Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. The reader is also referred to Kaiming's presentation (video, slides), and some recent experiments that reproduce these networks in Torch. 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).

2.2 OPTICAL FLOW

Recent breakthroughs in computer vision research have allowed machines to perceive its surrounding world through techniques such as object detection for detecting instances of objects belonging to a certain class and semantic segmentation for pixel-wise classification.

However, for processing real-time video input, most implementations of these techniques only address relationships of objects within the same frame (x,y) disregarding time information (t). In other words, they re-evaluate each frame independently, as if they are completely unrelated images, for each run. However, if we do need the relationships between consecutive frames, for example, we want to track the motion of vehicles across frames to estimate its current velocity and predict its position in the next frame we use optical flow.

Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera. The problem of optical flow may be expressed as:



where between consecutive frames, we can express the image intensity I as a function of space (x,y) and time (t). In other words, if we take the first image I(x,y,t) and move its pixels by (dx,dy) over t time, we obtain the new image I(x+dx,y+dy,t+dt).

First, we assume that pixel intensities of an object are constant between consecutive frames.

$$I(x,y,t) = I(x + \delta x , y + \delta y , t + \delta t) \tag{2.1}$$

Second, we take the Taylor Series Approximation of the RHS and remove common terms.

$$I(x + \delta x , y + \delta y , t + \delta t) = I(x,y,t) + \partial I/\partial x\,(\delta x) + \partial I/\partial y\,(\delta y) + \partial I/\partial t\,(\delta t) + \dots = 0 \tag{2.2}$$

Third, we divide by dt to derive the optical flow equation:

$$\partial I/\partial x\,(u) + \partial I/\partial y\,(v) + \partial I/\partial t = 0 \tag{2.3}$$

where $u=dx/dt$ and $v=dy/dt$

dI/dx, dI/dy, and dI/dt are the image gradients along the horizontal axis, the vertical axis, and time. Hence, we conclude with the problem of optical flow, that is, solving u(dx/dt) and v(dy/dt) to determine movement over time. You may notice that we cannot directly solve the optical flow equation for u and v since there is only one equation for two unknown variables. We will implement some methods such as the Lucas-Kanade method to address this issue.

2.2.1 SPARSE VS DENSE OPTICAL FLOW

Sparse optical flow gives the flow vectors of some "interesting features" (say few pixels depicting the edges or corners of an object) within the frame while Dense optical flow, which gives the flow vectors of the entire frame (all pixels) - up to one flow vector per pixel. Dense optical flow has higher accuracy at the cost of being slow/computationally expensive.

2.2.2 LUCAS-KANADE: SPARSE OPTICAL FLOW

Lucas and Kanade proposed an effective technique to estimate the motion of interesting features by comparing two consecutive frames in their paper An Iterative Image Registration Technique with an Application to Stereo Vision. The Lucas-Kanade method works under the following assumptions:

1. Two consecutive frames are separated by a small time increment (dt) such that objects are not displaced significantly (in other words, the method work best with slow-moving objects).
2. A frame portrays a “natural” scene with textured objects exhibiting shades of gray that change smoothly.

In a nutshell, we identify some interesting features to track and iteratively compute the optical flow vectors of these points. However, adopting the Lucas-Kanade method only works for small movements (from our initial assumption) and fails when there is large motion.

2.2.3 OPTICAL FLOW USING DEEP LEARNING

While the problem of optical flow has historically been an optimization problem, recent approaches by applying deep learning have shown impressive results. Generally, such approaches take two video frames as input to output the optical flow (colour-coded image)**,** which may be expressed as:

$$(u , v) = f (I_{t-1} , I_t) \quad (2.4)$$

where u is the motion in the x direction, v is the motion in the y direction, and f is a neural network that takes in two consecutive frames I_{t-1} (frame at time = $t-1$and I_t(frame at time = t) as input.

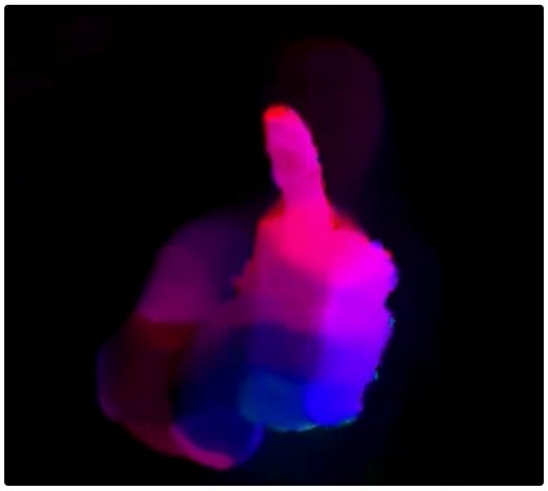


Figure 2.4 : Output of a deep learning model: colour-coded image; colour encodes the direction of pixel while intensity indicates their speed**.**[35]

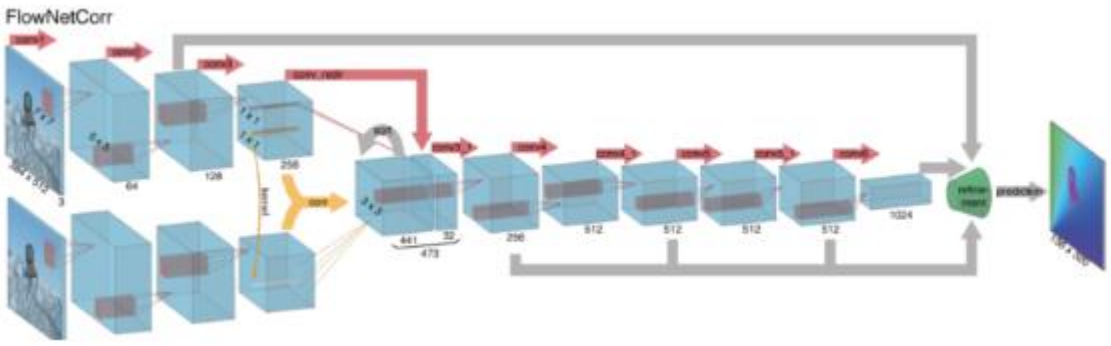


Figure 2.5 : Architecture of FlowNetCorr, a convolutional neural network for end-to-end learning of optical flow.[35]

Computing optical flow with deep neural networks requires large amounts of training data which is particularly hard to obtain. This is because labeling video footage for optical flow requires accurately figuring out the exact motion of each and every point of an image to subpixel accuracy. To address the issue of labeling training data, researchers used computer graphics to simulate massive realistic worlds. Since the worlds are generated by instruction, the motion of each and every point of an image in a video sequence is known.

Solving optical flow problems with deep learning is an extremely hot topic at the moment, with variants of FlowNet, SPyNet, PWC-Net, and more each outperforming one another on various benchmarks.

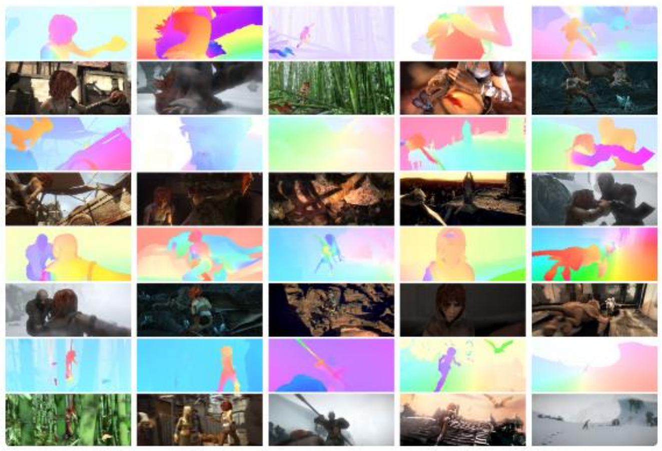


Figure 2.6 Synthetically generated data for training Optical Flow Models – MPI-Sintel dataset.[35]

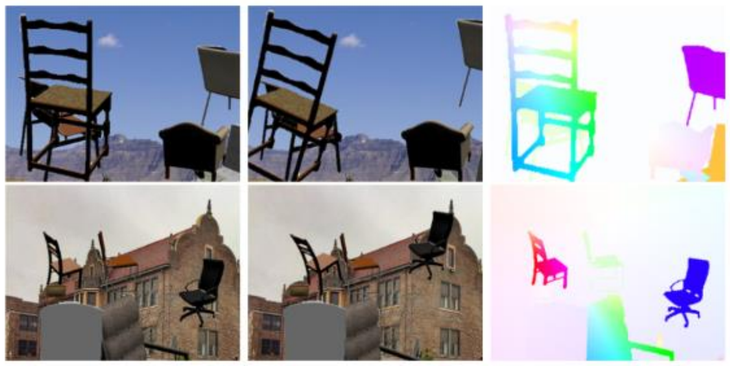


Figure 2.7 Synthetically generated data for training Optical Flow Models – Flying Chairs dataset[35]

2.2.4 APPLICATIONS OF OPTICAL FLOW

SEMANTIC SEGMENTATION

The optical flow field is a vast mine of information for the observed scene. As the techniques of accurately determining optical flow improve, it is interesting to see applications of optical flow in junction with several other fundamental computer visions tasks. For example, the task of semantic segmentation is to divide an image into series of regions corresponding to unique object classes yet closely placed objects with identical textures are often difficult for single frame segmentation techniques. If the objects are placed separately, however, the distinct motions of the objects may be highly helpful where discontinuity in the dense optical flow field correspond to boundaries between objects.

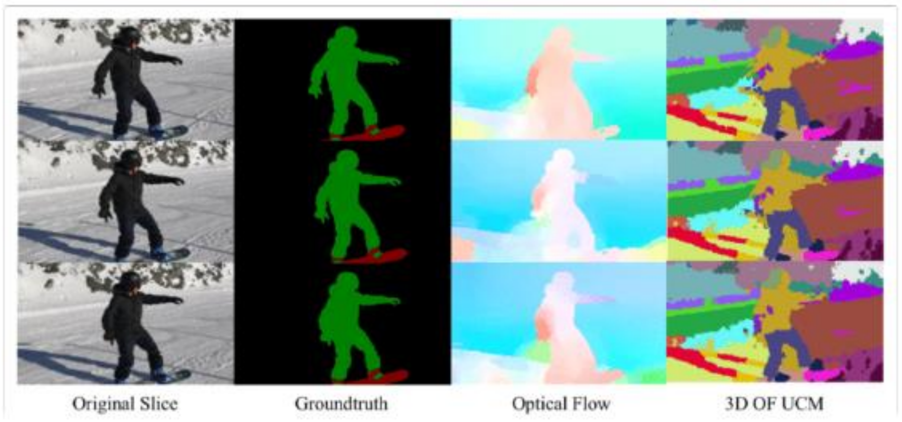


Figure 2.7 : Semantic segmentation generated from optical flow.[35]

OBJECT DETECTION AND TRACKING

Another promising application of optical flow may be with object detection and tracking or, in a high-level form, towards building real-time vehicle tracking and traffic analysis systems. Since sparse optical flow utilizes tracking of points of interest, such real-time systems may be performed by feature-based optical flow techniques from either from a stationary camera or cameras attached to vehicles.

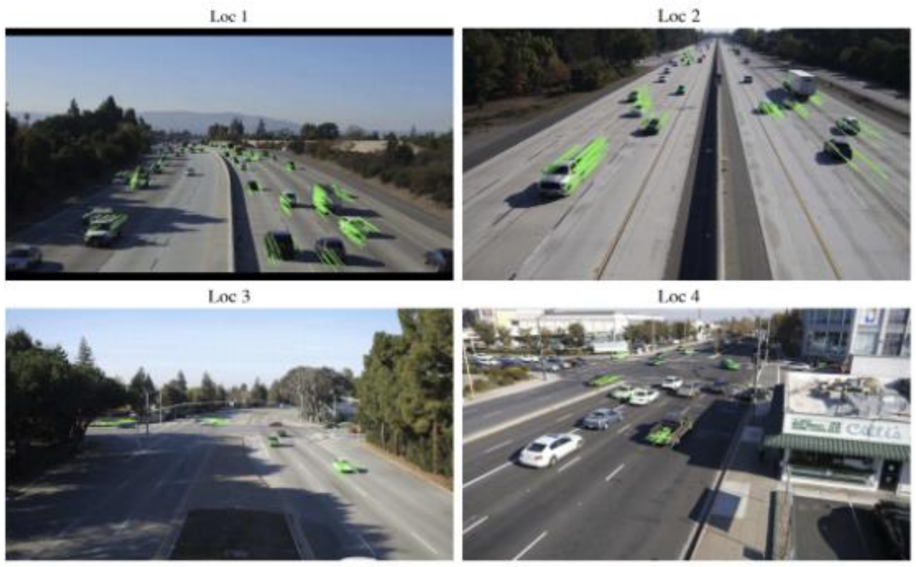


Figure 2.8 : Real-time tracking of vehicles with optical flow. [35]

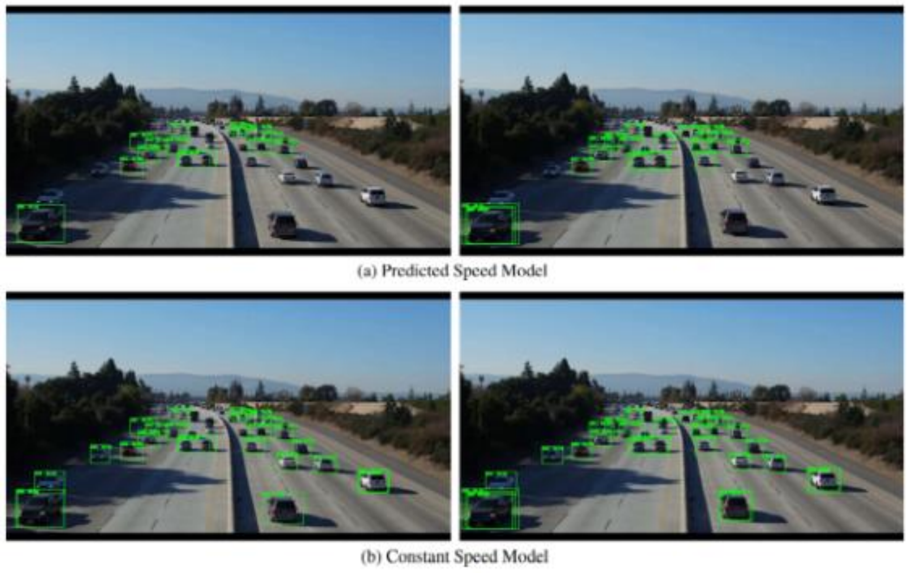


Figure 2.9 : Optical flow can be used to predict vehicle speeds[35]

Fundamentally, optical flow vectors function as input to a myriad of higher-level tasks requiring scene understanding of video sequences while these tasks may further act as building blocks to yet more complex systems such as facial expression analysis, autonomous vehicle navigation, and much more. Novel applications for optical flow yet to be discovered are limited only by the ingenuity of its designers.

2.3 HISTOGRAMS OF ORIENTED GRADIENTS

Histogram of oriented gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI).

Implementation of the HOG descriptor algorithm is as follows:

- Divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell.
- Discretize each cell into angular bins according to the gradient orientation.
- Each cell's pixel contributes weighted gradient to its corresponding angular bin.
- Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms.
- Normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

Computation of the HOG descriptor requires the following basic configuration parameters:

- Masks to compute derivatives and gradients
- Geometry of splitting an image into cells and grouping cells into a block
- Block overlapping
- Normalization parameters

According to Dalal the recommended values for the HOG parameters are:

- 1D centered derivative mask [-1, 0, +1]
- Detection window size is 64x128
- Cell size is 8x8
- Block size is 16x16 (2x2 cells)

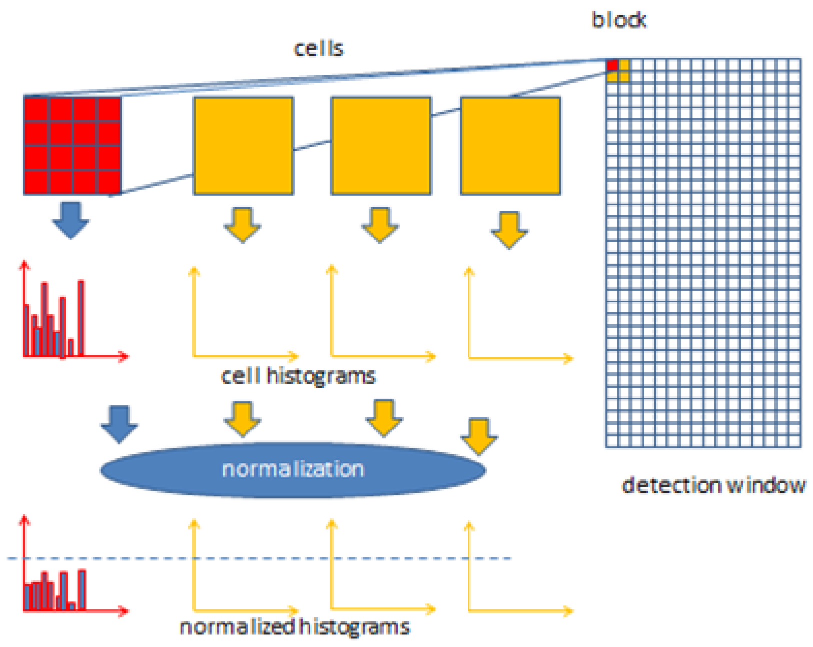


Figure 2.10 : The HOG algorithm implementation scheme [32]

2.4 SUPPORT VECTOR MACHINE (SVM)

In machine learning, **support-vector machines** (**SVMs**, also **support-vector networks**) are supervised learning models with associated learning algorithms that analyze data used for classification. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier . An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

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).

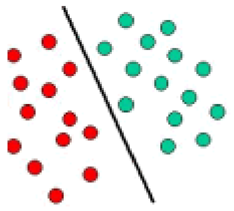


Figure 2.11 : Example of linear SVM[36]

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.

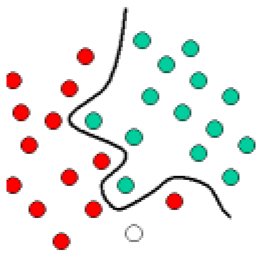


Figure 2.12 : Seperating lines to distinguish between objects.[36]

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). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

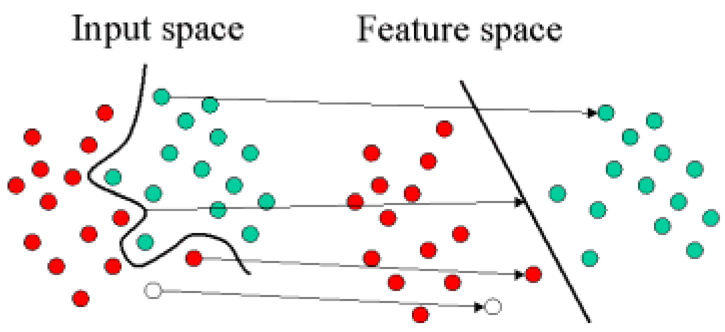


Figure 2.13 : Original objects mapped to a new feature space[36]

2.5 MULTICLASS SVM :

In multi class or multinomial classification is the problem of classifying instances into one of three or more classes. (Classifying instances into one of two classes is called binary classification.) SVM are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies. One such strategy is one vs rest.

ONE VS REST :

One-vs.-rest (or one-vs.-all, OvA or OvR, one-against-all, OAA) strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.

In pseudocode, the training algorithm for an OvA learner constructed from a binary classification learner L is as follows:

```
Inputs:
    • L, a learner (training algorithm for binary classifiers)
    • samples X
    • labels y where $y_i \in \{1, \dots K\}$ is the label for the sample $X_i$
Output:
    • a list of classifiers $f_k$ for $k \in \{1, \dots, K\}$
Procedure:
    • For each k in $\{1, \dots, K\}$
        o Construct a new label vector z where $z_i = y_i$ if $y_i = k$ and $z_i = 0$ otherwise
        o Apply L to X, z to obtain $f_k$
```

Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score:

CHAPTER 3 DATASET

In this section we present our pedestrian movement direction recognition dataset. This dataset is used for the training and evaluation of the proposed system. To test the proposed CNN-based system we needed a specific dataset, designed to feed our network with images of pedestrians moving in different directions and in different scenarios, boardwalks, zebra crossings, sidewalks, etc. Video was recorded with a camera capturing at 30fps, 640×480 resolution. Videos were recorded in a static and dynamic manner, using a regular and a handheld tripod. There are no other datasets that provide information about pedestrian movement direction. Most existing datasets related to this topic, such as the:

Caltech benchmark Dataset consists of approximately 10 hours of 640x480 30Hz video taken from a vehicle driving through regular traffic in an urban environment. About 250,000 frames (in 137 approximately minute long segments) with a total of 350,000 bounding boxes and 2300 unique pedestrians were annotated. The annotation includes temporal correspondence between bounding boxes and detailed occlusion labels.

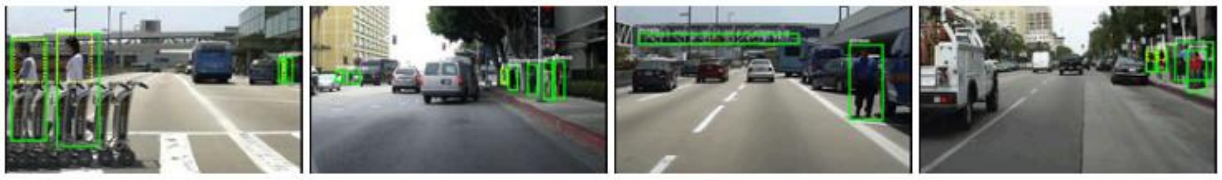


Figure 3.1 : Caltech dataset[37]

The Pascal database is a set of additional annotations for PASCAL VOC 2010. It goes beyond the original PASCAL semantic segmentation task by providing annotations for the whole scene. The statistics section has a full list of 400+ labels. The INRIA person dataset focus only on pedestrian detection, providing bounding boxes of the detected pedestrians, but no information about the pedestrian direction or intention is provided.

The Daimler dataset is the only one that not only provides ground truth information for pedestrian detection tasks but also pedestrian intention information. Four different pedestrian motion types are considered: crossing, stopping, starting to walk and bending in. Each video sequence has the previously mentioned label about pedestrian intention.

In our dataset, we considered different motion types, not previously included in existing datasets. We decided to recognize more generic motion types, such as moving left, right or to the front, which can be used to recognize higher-level motion types like those proposed in the Daimler dataset. This dataset was manually annotated, creating a ground truth split for training the proposed CNN architecture.

RELATED DATASETS

- GM-ATCI: Rear-View Pedestrians Dataset captured from a fisheye-lens camera.
- Daimler: Also captured in an urban setting, update of the older DaimlerChrysler dataset. Contains tracking information and a large number of labeled bounding boxes.
- NICTA: A large scale urban dataset collected in multiple cities/countries. No motion/tracking information, but significant number of unique pedestrians.
- ETH: Urban dataset captured from a stereo rig mounted on a stroller.
- TUD-Brussels: Dataset with image pairs recorded in an crowded urban setting with an onboard camera.
- INRIA: Currently one of the most popular static pedestrian detection datasets.
- PASCAL: Static object dataset with diverse object views and poses.
- CVC-ADAS: collection of pedestrian datasets including pedestrian videos acquired on-board, virtual-world pedestrians (with part annotations), and occluded pedestrians.
- USC: A number of fairly small pedestrian datasets taken largely from surveillance video.
- MIT: One of the first pedestrian datasets, fairly small and relatively well solved at this point

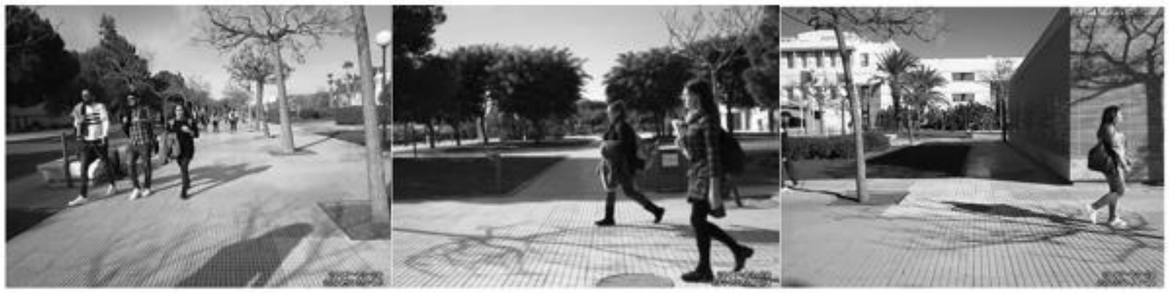


Figure 3.2 : Dataset images: Left, right and front samples

CHAPTER 4. PROPOSED METHOD

This section describes our method for pedestrian trajectory recognition. It is based on a CNN network trained with preprocessed data from a video feed as input. Once the CNN was defined, and the dataset was annotated with ground truth information, we started to feed the CNN with training data from the recorded dataset.

4.1 PREPROCESSING STEPS

In the proposed method, acquired video is passed through an image preprocessing pipeline for image filtering, obtaining our final added video frames , which were used as input data for the proposed CNN architecture. Only one of every six frames was used, as we saw that consecutive frames at 30 frames per second contributed no new relevant features to our network.

4.1.1 OPTICAL FLOW

To detect changes in scene or camera movements we use optical flow method. We are estimating optical flow using Lucas-Kanade derivative of Gaussian method. We create an optical flow object for estimating the direction and speed of moving objects using the Lucas-Kanade method. The object of optical flow stores the direction and speed of a moving object from one image or video frame to another. The properties of optical flow object are:

Vx — x component of velocity
x component of velocity, specified as an M-by-N matrix. If the input Vx is not specified, the default value of this property is set to 0-by-1 empty matrix.

Vy — y component of velocity
y component of velocity, specified as an **M**-by-**N** matrix. If the input Vy is not specified, the default value of this property is set to 0-by-1 empty matrix.

Orientation — Phase angles of optical flow

This property is read-only. Phase angles of optical flow in radians, specified as an M-by-N matrix of the same size and data type as the components of velocity. The phase angles of optical flow is calculated from the x and y components of velocity. If the inputs Vx and Vy are not specified, the default value of this property is set to 0-by-1 empty matrix

Magnitude — Magnitude of optical flow

This property is read-only. Magnitude of optical flow, specified as an M-by-N matrix of the same size and data type as the components of velocity. The magnitude of optical flow is calculated from the x and y components of velocity. If the inputs Vx and Vy are not specified, the default value of this property is set to 0-by-1 empty matrix.

Threshold -- The threshold of noise reduction

Threshold for noise reduction, specified as a positive scalar. As you increase this number, the movement of the objects has less impact on optical flow calculation. We have taken the threshold as 0.005 since we have to detect the movement of pedestrians.

We use the magnitude property of the object flow created in matlab to find if there is any movement in the scene or not. If there is movement in the scene then the magnitude is greater than 0.008. We have found this value after seeing the result over a video having movement and no movement in the scene.

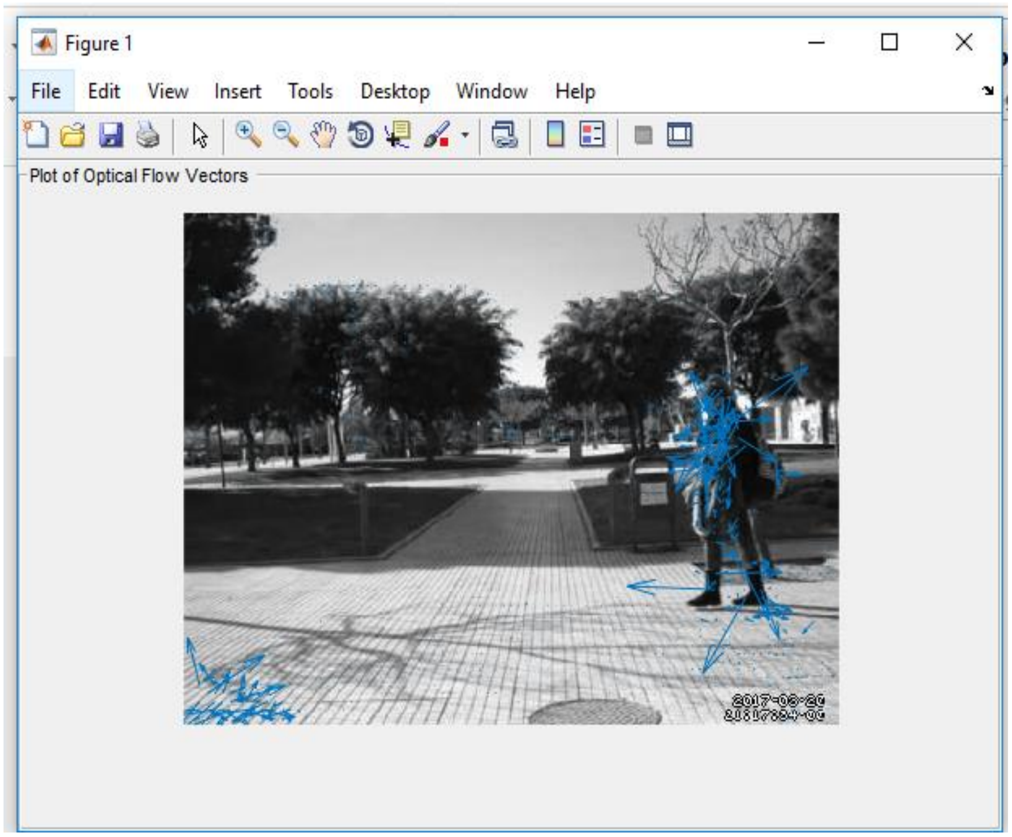


Figure 1
File Edit View Insert Tools Desktop Window Help
Plot of Optical Flow Vectors
2017-03-20

Figure 4.1 : Plot of optical flow vectors .

4.1.2 PEDESTRIAN DETECTION

Once we have detected movement in the scene we move onto the next step of pedestrian detection .We use Histograms of Oriented Gradients (HOG) features for classification and Support Vector Machine (SVM) as the classifier. If pedestrian is detected we move onto the next step.

4.1.3 IMAGE SUBTRACTION

After pedestrian detection we perform an image subtraction step , which is an absolute difference between two consecutive frames. In this way, the contour of the pedestrian shows a slight movement in a particular direction. Then, we continue processing the sequence obtaining the next subtracted frame.

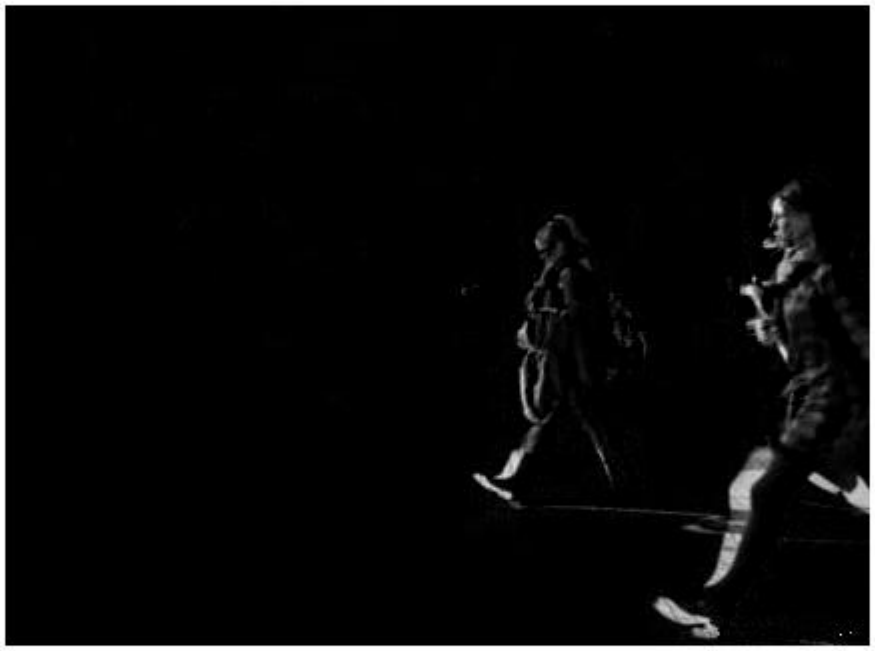


Figure 4.2: Result obtained after image subtraction.

4.1.4 IMAGE ADDITION

Finally, using consecutive subtracted frames, we perform the sum (pixel level) of these subtracted frames obtaining a similar black background frame. This output image is used as input data for the proposed CNN architecture, which automatically learns to extract features from these preprocessed images.

The images are then resized (downscaled) to 32x32 pixels. In addition, as the colour provides no further relevant information, we decided to convert the images to greyscale just after the acquisition step

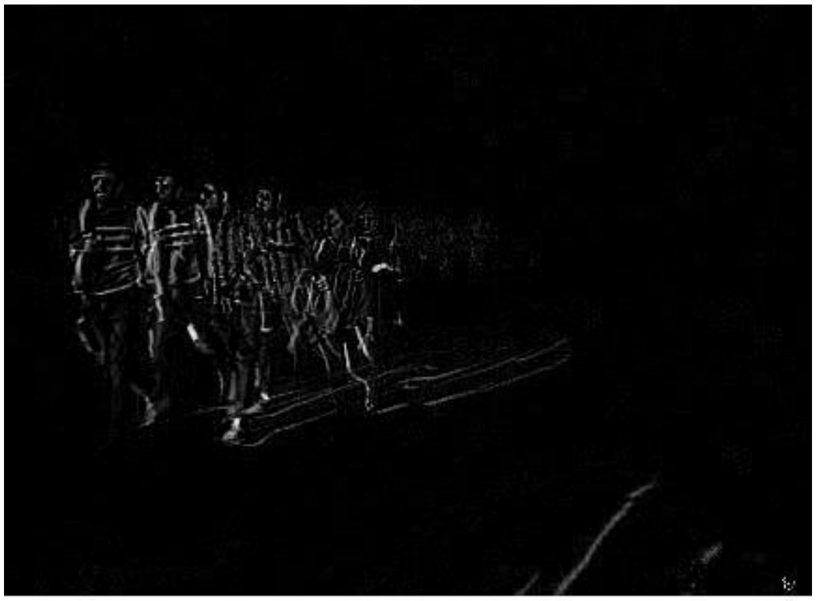


Figure 4.3: Sum of suubtrated frames

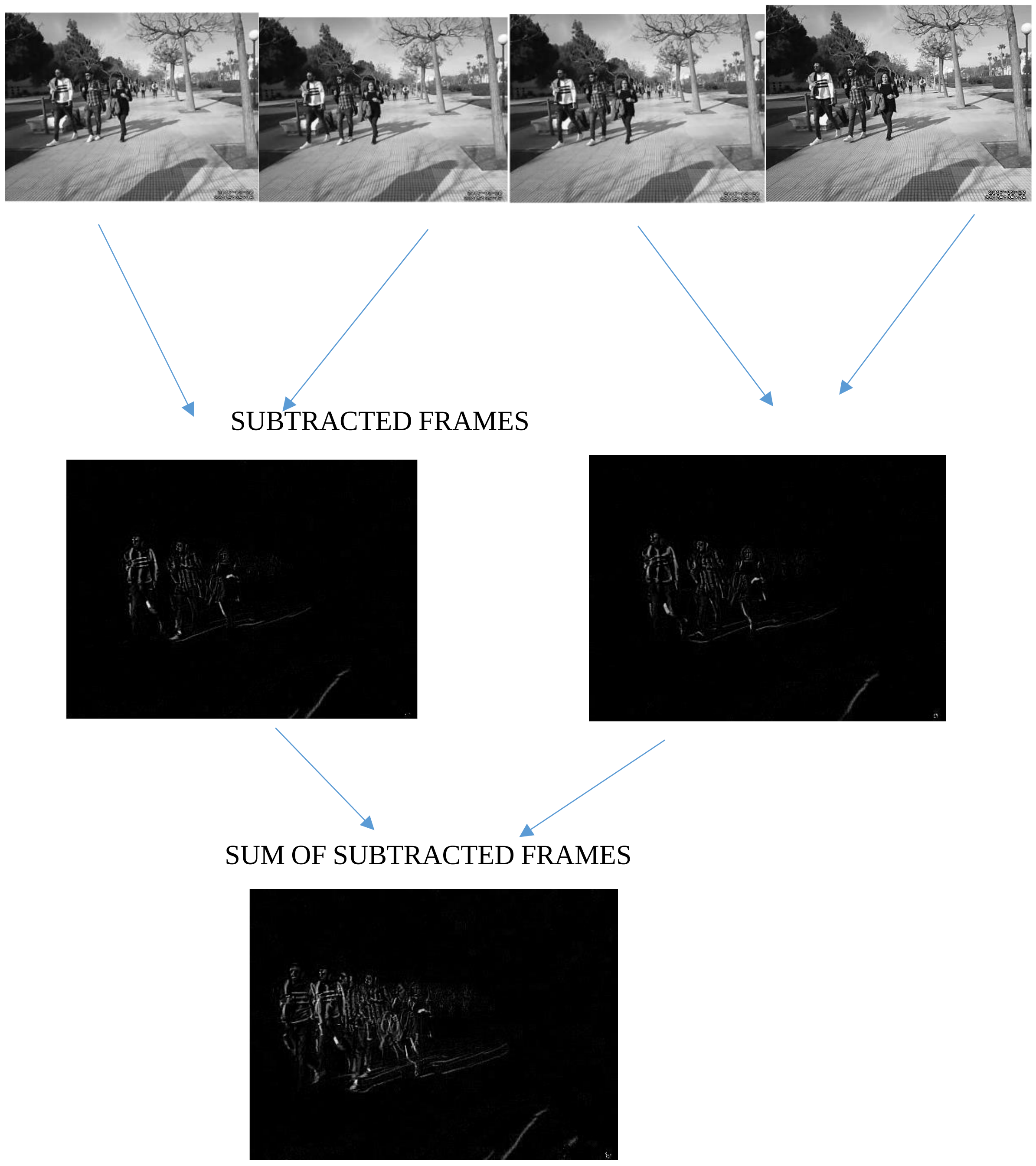


Figure 4.4: Preprocessing steps carried out on the images.

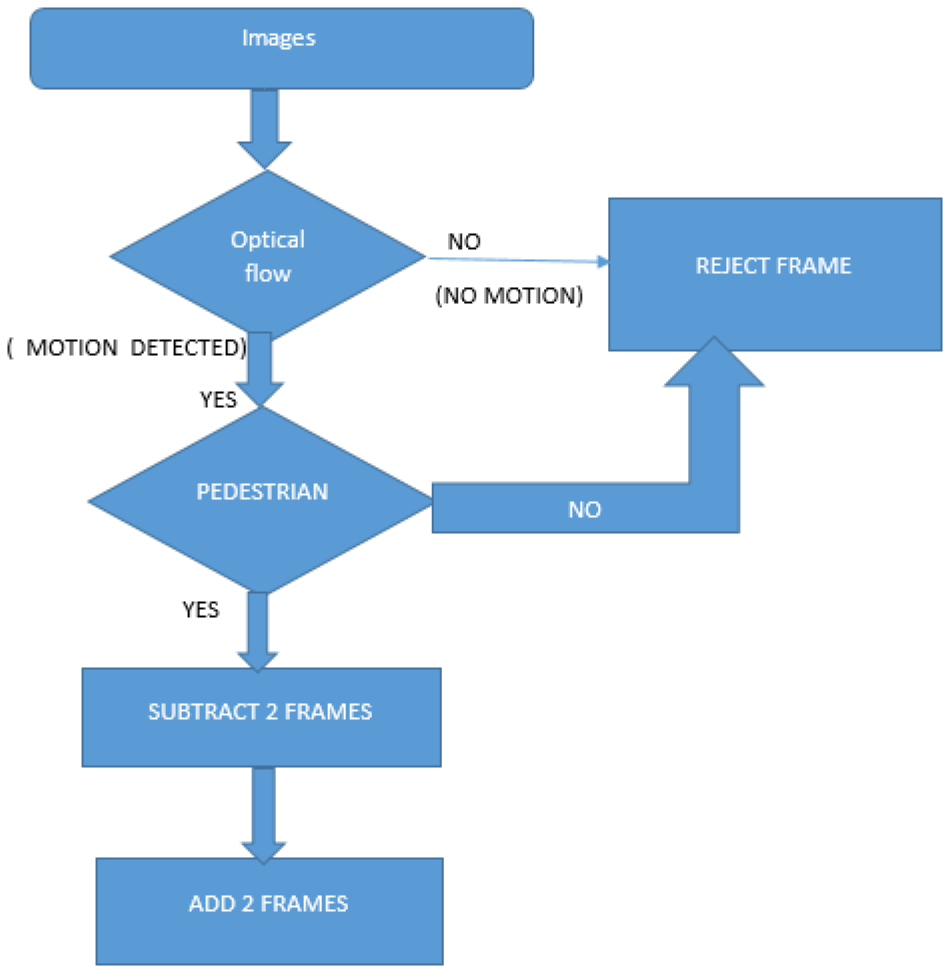


Figure 4.5: Flowchart of the preprocessing steps

4.2 HYBRID CNN TRAINING

The last step is the training of the hybrid CNN. The architecture of the hybrid CNN we used in this work is as follows:

Image Input Layer An image Input Layer is where you specify the image size, which, in this case, is 32-by-32-by-1. These numbers correspond to the height, width, and the channel size. The digit data consists of grayscale images, so the channel size (color channel) is 1.

Convolutional Layer – We have used a total of three convolutional layers the properties of which will be discussed one by one below:

For the first layer we have used a filter size of 2 that is a 2x2 filter and for the next two layers we used a filter size of three. The next argument we need to specify in this layer is the number of filters, which is the number of neurons that connect to the same region of the input. This parameter determines the number of feature maps. In the first layer we have used the number of filters as 8 , in the second layer the number of filters used is 16 and in the third layer it is 32 .

We want the spatial output size same as the spatial input size for this we use padding. We can calculate the padding using the formula but since we have used the parameter 'same' in our code matlab software does it automatically for us and we do not have to worry about the padding. In case we wanted the spatial input size and spatial output size different we could have specified the padding required to do so using the formula. In all the three layers we have used the padding parameter as 'same' since in all three layers we have kept the spatial input size and output size same. Next parameter we have specified is the step size for traversing the input image horizontally and vertically. We have used the default step size that is one. We could have initialized the weights and the learn rate factor for the layer but we have the default weight initializer which is the 'glorot' in our software and the default learn rate parameter of 1.

Batch Normalization Layer – This is the next layer after convolutional layer.

A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers. The layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by a learnable offset € and scales it by a learnable scale factor γ.

In this layer after network training finishes, the software calculates the input mean over the entire training data set. The layer uses this trained mean (in place of the mini-batch mean) to normalize the input during prediction. Similarly the the software calculates the input variance over the entire training dataset , and then this trained variance is used in place of the mini batch variance to normalize the input during prediction. The value of epsilon used is 1e – 5. The algorithm of this layer has been defined in Chapter 2.1.2 under the heading Normalization Layer.

ReLU Layer The batch normalization layer is followed by a nonlinear activation function. The most common activation function is the rectified linear unit (ReLU) and in our work also we have used a ReLU layer which is giving us good results.The ReLU layer does not change the size of its input.

There are other nonlinear activation layers that perform different operations and can improve the network accuracy for some applications. Other activation layers are as follows:

Leaky ReLU layer – It performs a threshold operation, where any input value less than zero is multiplied by a fixed scalar.

A clipped ReLU layer - It performs a threshold operation, where any input value less than zero is set to zero and any value above the clipping ceiling is set to that clipping ceiling.

An ELU activation layer - It performs the identity operation on positive inputs and an exponential nonlinearity on negative inputs.

A hyperbolic tangent (tanh) activation layer - It applies the tanh function on the layer inputs.

A PReLU layer - It performs a threshold operation, where for each channel, any input value less than zero is multiplied by a scalar learned at training time.

We tested our code for all these layers but got the best results for ReLU layer .

Max Pooling Layer – The next layer we have created after the ReLU layer is the max pooling layer . This basically does the downsampling operation , it reduces the spatial size of the feature map and reduces redundant spatial operation**.** Down-sampling makes it possible to increase the number of filters in deeper convolutional layers without increasing the required amount of computation per layer. One way of downsampling is max pooling which we have used here. In this layer we specify the pool size which gives the maximum value of the rectangular region given by poolsize. In our work we have taken a poolsize of 2 that is a 2x2 filter which is convolved across the entire feature map to give the maximum value out of the rectangular region. The stride we have taken is 2 . if the stride is taken as different from the poolsize then the there is overlapping of the the downsampling operation . Here we have set the default padding as ‘same’ that is the software automatically calculates the padding required for the input feature map. After all the ReLU layer in each block we have used this max pooling layer .

Fully Connected Layer The convolutional and down-sampling layers are followed by one or more fully connected layers. This layer combines all the features learned by the previous layers across the image to identify the larger patterns. The last fully connected layer combines the features to classify the images. We have named it as fc layer.

SVM CLASSIFIER :

After the fully connected layer in the CNN architecture is the softmax layer which is used to normalize the ouput of the fully connected layer , which is can be used as the classification probabilities by the classification layer. But instead of using the softmax layer as the classifier we are using an SVM classifier. The architecture of the hybrid CNN & SVM model was designed by replacing the last layer (fc) of the CNN model with an SVM classifier in the testing phase. For output units of the fc layer in the CNN network, they are the estimated probabilities for the input sample. Each output probability is calculated by an activation function. The input of the activation function is the linear combination of the outputs from the previous f3 layer with trainable weights, plus a bias term. Looking at the output values of f3 is meaningless, but only makes sense to the CNN network itself; however, these values can be treated as features for any other classifiers. The prediction of the unknown sample is made by an SVM classifier instead of the fc layer. After the original CNN has been trained by the back-propagation algorithm, the outputs produced from Layer f3 are extracted as the new features. They are sent to the SVM classifier for training. Once the SVM classifier has been well trained, it conducts the recognition task with corresponding features from the testing data. Firstly, the processed image is sent to the input layer, and the original CNN with the output unit (fc) is trained with several epochs until the training process converges. Then, the SVM with an RBF kernel replaces the output layer fc. The SVM takes the outputs from the f3 layer as a new feature vector. Finally, the trained SVM makes new decisions on testing images with such automatically extracted features.

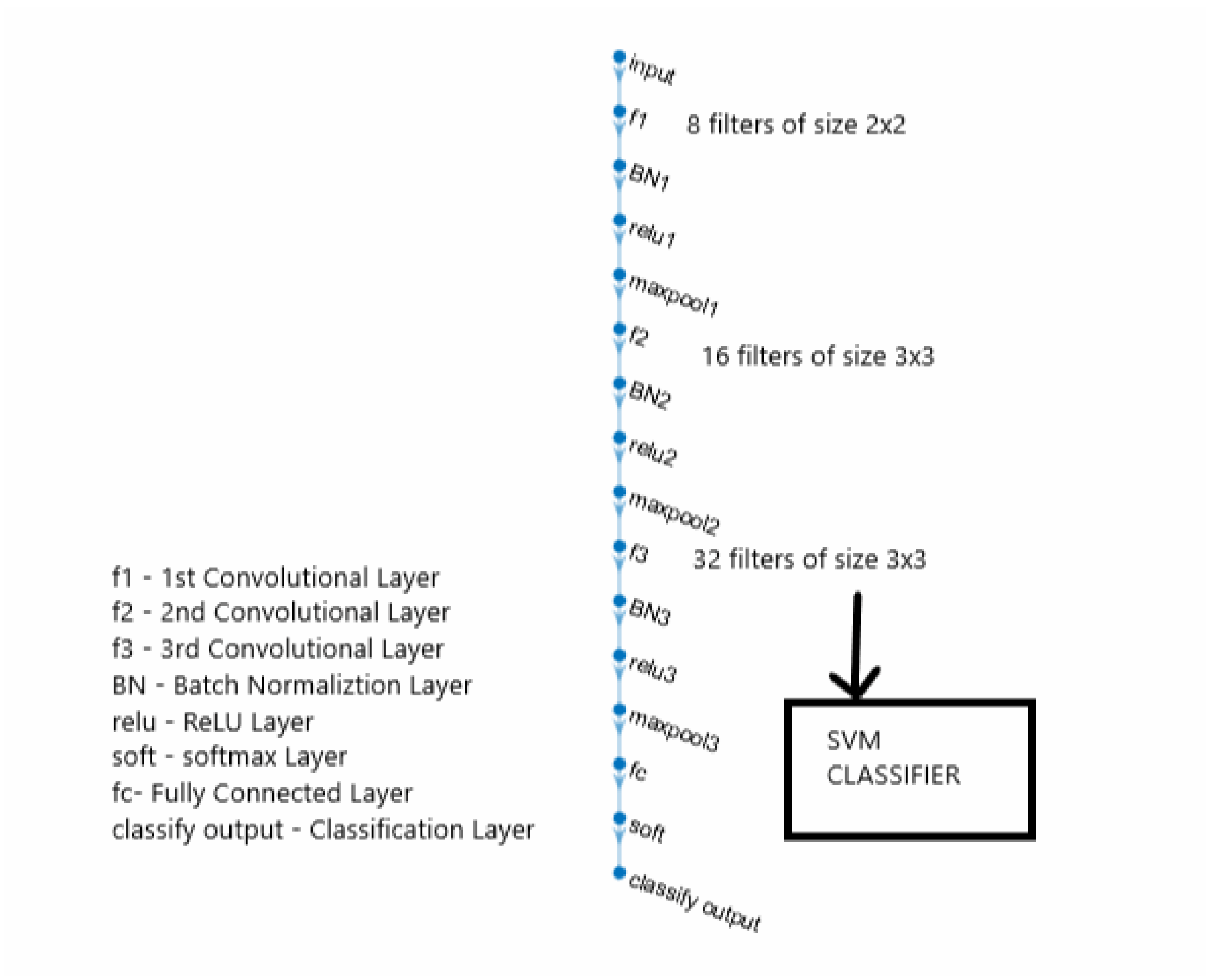


Figure 4.6 : Architecture of the hybrid CNN – SVM model

4.3 MERITS OF HYBRID CNN AND SVM MODEL :

The hybrid system compensates the limits of the CNN and SVM classifiers by incorporating the merits of both classifiers. Since the theoretical learning method of CNN is the same as that for the Multi-Layer Perceptron (MLP), it is an extension model of the MLP. The learning algorithm of MLP is based on the Empirical Risk Minimization, which attempts to minimize the errors in the training set. When the first separating hyperplane is found by the back-propagation algorithm, no matter whether it is the local or the global minima, the training process stops and the algorithm does not improve the separating hyperplane solution. Therefore, the generalization ability of MLP is lower than that of SVM. On the other hand, the SVM 27 classifier aims to minimize the generalization errors on the unseen data with a fixed distribution of the training set, by using the Structural Risk Minimization principle. The separating hyperplane is a global optimum solution. It is calculated by solving a quadratic programming problem, and the margin area between two classes of training samples reaches its maximum. As a result, the generalization ability of SVM is maximized. Due to the good generalization ability of the SVM, it should enhance the classification accuracy after its replacement of the fc output units from the CNN. The advantage of the CNN classifier is that it automatically extracts the salient features of the input image. The features are invariant in a certain degree to the shift and shape distortions of the input characters. This invariance occurs because CNN adopts the weight sharing technique on one feature map. On the contrary, the hand-designed feature extractor needs elaborately designed features. Therefore, the trainable features of CNN can be used instead of the hand-designed features to collect more representative and relevant information.

CHAPTER 5 EXPERIMENTS

In this section, we describe the experimental setup and the different experiments carried out for the validation of the proposed method. We evaluated the system using our proposed Convolutional Neural Networks. In order to evaluate the performance of our proposal in terms of accuracy, we used the dataset presented in Chapter 3. First, we tested the system using the hybrid Convolutional neural nework - SVM and performed an exhaustive experimental tuning process in order to boost the accuracy of the proposed system. Finally, we present qualitative results showing the accuracy of the proposed system using our Hybrid CNN architectures. We also compared the results with the work previously done in this field.

5.1 CNN EVALUATION

TRAINING OPTIONS :

The training options we have used in our code are as follows:

TRAINING ALGORITHM: We have trained our network using stochastic gradient descent with momentum (SGDM) with an initial learning rate of 0.01.The standard gradient descent algorithm updates the network parameters (weights and biases) to minimize the loss function by taking small steps at each iteration in the direction of the negative gradient of the loss, The stochastic gradient descent algorithm can oscillate along the path of steepest descent towards the optimum. Adding a momentum term to the parameter update is one way to reduce this oscillation . The stochastic gradient descent with momentum (SGDM) update is

$$\theta_{\ell+1}=\theta_\ell-\alpha\nabla E(\theta_\ell)+\gamma(\theta_\ell-\theta_{\ell-1}), \qquad (5.1)$$

where γ determines the contribution of the previous gradient step to the current iteration. We varied the momentum from .8 to .95 but .9 gave optimum results. We also used the rmsProp optimizer but the training is giving smoother results for SGDM so we used the latter.

Max epochs : Set the maximum number of epochs to 8. An epoch is a full training cycle on the entire training data set. We also experimented using the epoch value of 6 and 10 but maximum epoch of 8 gave optimum results. Also we shuffle the data after every epoch. For an epoch value of 6 we have the validation accuracy of 90.21% and classification accuracy of 82.24%. An epoch of 8 is giving validation accuracy of 91.48% and classification accuracy of 86.06%.it means for an epoch value of 8 hybrid CNN is giving good results on the test data. For an epoch value of 10 the validation accuracy is 95.74 % whereas the classification accuracy is 81% which means overfitting of CNN is taking place in this case.

Validation data : we have split the dataset randomly into training , test, and validation data in the ratio of 0.6, 0.1 , 0.3 respectively. So the network is trained on the training set that is the weights are updated on the training set and calculates the accuracy on validation data at regular intervals of time. The validation data is not used to update the network weights.

Validation frequency: The 'validation frequency' value is the number of iterations between evaluations of validation metrics. We have used a validation frequenvy of 30.

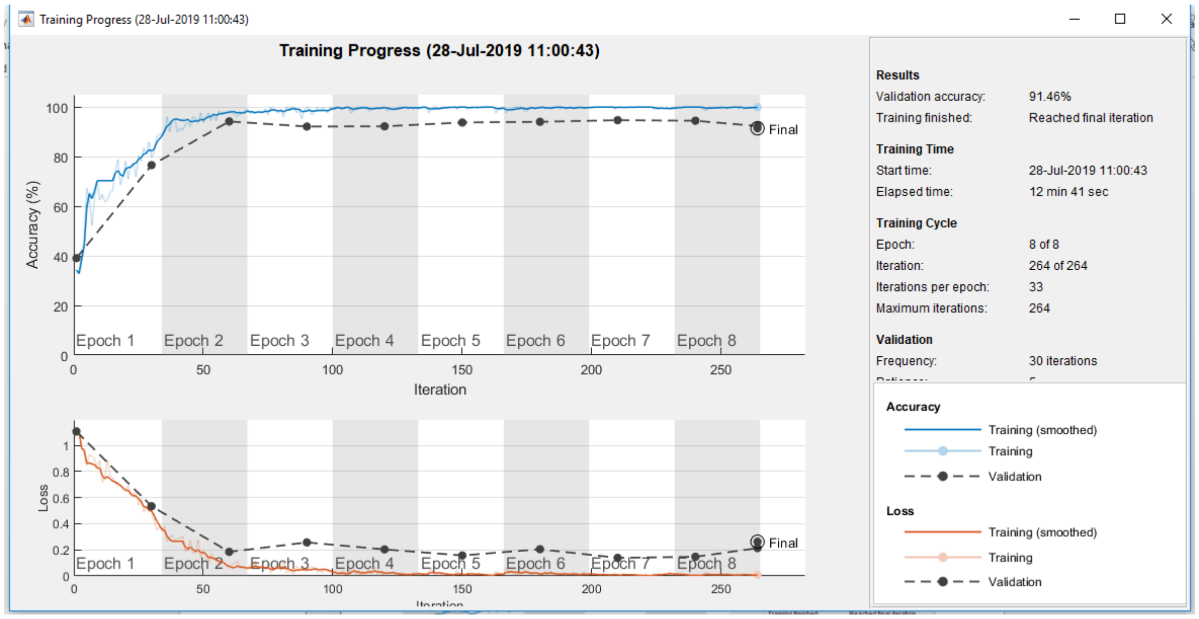
Validation patience : The 'Validation Patience' value is the number of times that the loss on the validation set can be larger than or equal to the previously smallest loss before network training stops. We have used a validation patience of 5. It could be default value as well which is inf , that is the training stops only after completing the number of epochs.

Mini batch size : A mini-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. We have used a mini batch size of 128 , which is giving us optimum results. On decreasing the batch size to 75 the validation accuracy as well as test accuracy drops down.

5.2 RESULTS

The accuracy of the evaluated hybrid CNN was quantified using the validation/test splits created during the dataset generation. After many different executions we get classification accuracy ranging from 74.8% to 87% . Best results were obtained for a value of 0.9 for the momentum parameter. It can be seen that the loss function in all the cases also tends to a minimum of between 0.1 and 0.3. We use the maximum epochs of 8 since it is giving better results on test data. The maximum iterations are 264 and iteration per epoch is 33.CNN architecture of 4 layers is giving best results. So the maximum validation accuracy is 91.46%

and maximum classification accuracy is 86.06% with the time taken for training as 12 min 41 sec.

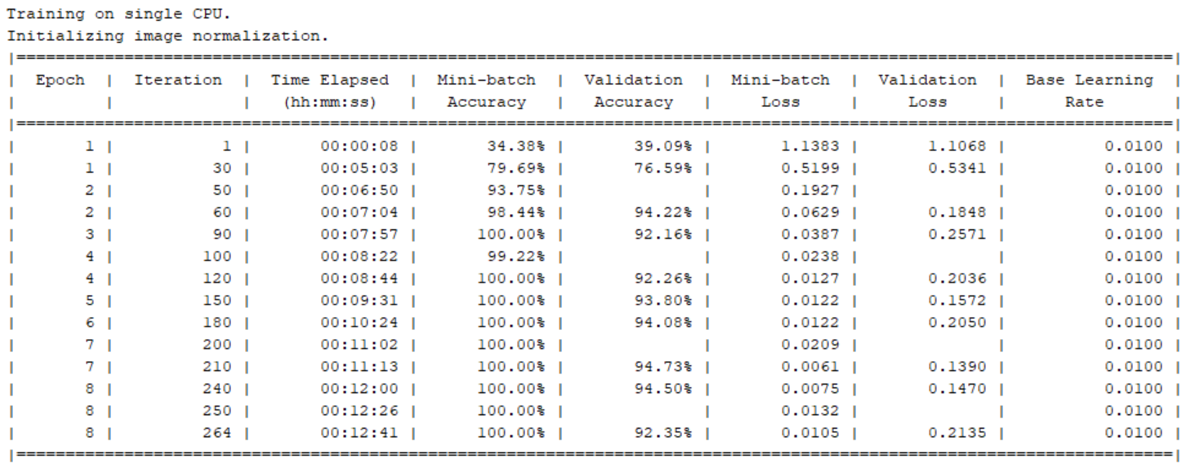


Results	
Validation accuracy:	91.46%
Training finished:	Reached final iteration
Training Time	
Start time:	28-Jul-2019 11:00:43
Elapsed time:	12 min 41 sec
Training Cycle	
Epoch:	8 of 8
Iteration:	264 of 264
Iterations per epoch:	33
Maximum iterations:	264
Validation	
Frequency:	30 iterations

Figure 5.1 : Training progress of proposed hybrid CNN

	FRONT	LEFT	RIGHT
FRONT	.9250	.0050	0
LEFT	.3026	.7684	.0789
RIGHT	.0211	.0105	.8884

Table 5.1 : Confusion Matrix of proposed hybrid CNN



```
Training on single CPU.
Initializing image normalization.
|======================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate       |
|======================================================================================================================|
|       1 |           1 |       00:00:08 |       34.38% |       39.09% |       1.1383 |       1.1068 |          0.0100 |
|       1 |          30 |       00:05:03 |       79.69% |       76.59% |       0.5199 |       0.5341 |          0.0100 |
|       2 |          50 |       00:06:50 |       93.75% |              |       0.1927 |              |          0.0100 |
|       2 |          60 |       00:07:04 |       98.44% |       94.22% |       0.0629 |       0.1848 |          0.0100 |
|       3 |          90 |       00:07:57 |      100.00% |       92.16% |       0.0387 |       0.2571 |          0.0100 |
|       4 |         100 |       00:08:22 |       99.22% |              |       0.0238 |              |          0.0100 |
|       4 |         120 |       00:08:44 |      100.00% |       92.26% |       0.0127 |       0.2036 |          0.0100 |
|       5 |         150 |       00:09:31 |      100.00% |       93.80% |       0.0122 |       0.1572 |          0.0100 |
|       6 |         180 |       00:10:24 |      100.00% |       94.08% |       0.0122 |       0.2050 |          0.0100 |
|       7 |         200 |       00:11:02 |      100.00% |              |       0.0209 |              |          0.0100 |
|       7 |         210 |       00:11:13 |      100.00% |       94.73% |       0.0061 |       0.1390 |          0.0100 |
|       8 |         240 |       00:12:00 |      100.00% |       94.50% |       0.0075 |       0.1470 |          0.0100 |
|       8 |         250 |       00:12:26 |      100.00% |              |       0.0132 |              |          0.0100 |
|       8 |         264 |       00:12:41 |      100.00% |       92.35% |       0.0105 |       0.2135 |          0.0100 |
|======================================================================================================================|
```

Figure 5.2: Training progress information chart.

5.3 COMPARISON WITH PREVIOUS WORK

We compare the proposed approach with the work proposed earlier for pedestrian movement detection. In one such work an experiment was carried out where they used the dense optical flow algorithm described in and the gradients of a Motion History Image (MHI) to detect direction of motion. Given consecutive images where a pedestrian has been detected, the optical flow is calculated and used to estimate the direction of the segmented pedestrian. It was empirically tested using different thresholds for segmentation and MHI computation, choosing the ones providing best performance (MHI duration = 0.05 milliseconds, segmentation threshold = 35 (HSV distance), Max time delta = 125000.0 and min time delta = 5). Optical flow global direction is calculated as a mean weighted direction using all moving pixels. This provides us with a value between 0 and 360 degrees. Finally, those values were discretized in three ranges (120 bin size), for each predefined direction: left, right and front. Table shows quantitative results on the proposed dataset using the approach described above. It can be observed that this approach performs poorly in the test set, obtaining an average accuracy of 51%, 39% and 40% respectively for each motion type.

	Front	Left	Right
Front	0.51	0.35	0.14
Left	0.36	0.39	0.25
Right	0.38	0.220	0.40

Table 5.2 : Confusion matrix results using optical flow and the gradients of a motion history image

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

We have presented a method to differentiate the motion of pedestrians in real life environments. By building a novel input-filtered image based on the post-processing of static recorded video frames, we have managed to successfully distinguish three different pedestrian movement directions. Additionally, it has been proved how hybrid CNN –SVM model can impressively perform in such a task by training them with a specialised dataset. We have also presented an evaluation of our model giving a validation accuracy of 91.46% and a test accuracy of 86.06% .

As future directions, we are working on a better and more robust use of data augmentation, which should provide a more robust model. Also we are planning to use this pipeline using other hybrid CNN and making use of Long Short Term Memory (LSTM) and RNN, to give more accurate results. In future we would also like to work on a model that detects the direction of movement of the pedestrian as well as his/her speed.

CHAPTER 7 REFERENCES

[1] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05) - Volume 1 - Volume 01, ser. CVPR ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.177

[2] S. Walk, N. Majer, K. Schindler, and B. Schiele, “New features and insights for pedestrian detection.” in CVPR. IEEE Computer Society, 2010, pp. 1030–1037.

[3] M. Enzweiler and D. M. Gavrila, “Monocular pedestrian detection: Survey and experiments,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 99, 2009.

[4] R. Benenson, M. Omran, J. Hosang, and B. Schiele, Ten Years of Pedestrian Detection, What Have We Learned? Cham: Springer International Publishing, 2015, pp. 613–627. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-16181-5 47

[5] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 4, pp. 743–761, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2011.155

[6] P. Viola and M. J. Jones, “Robust real-time face detection,” Int. J. Comput. Vision, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb

[7] D. G. Lowe, “Object recognition from local scale-invariant features,” in Proceedings of the International Conference on Computer VisionVolume 2 - Volume 2, ser. ICCV ’99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, 1998, pp. 2278–2324.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/ 4824-imagenet-classification-with-deep-convolutional-neural-networks. pdf

[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] M. Enzweiler and D. M. Gavrila, “Integrated pedestrian classification and orientation estimation.” in CVPR. IEEE Computer Society, 2010, pp. 982–989.

[12] T. Gandhi and M. M. Trivedi, “Image based estimation of pedestrian orientation for improving path prediction,” in 2008 IEEE Intelligent Vehicles Symposium, 2008.

[13] A. Mgelmose, M. M. Trivedi, and T. B. Moeslund, “Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations.” in Intelligent Vehicles Symposium. IEEE, 2015, pp. 330–335. [Online]. Available: http://dblp.uni-trier.de/db/conf/ivs/ivs2015.html#MogelmoseTM15

[14] T. F. Fugger, B. C. Randles, A. C. Stein, W. C. Whiting, and B. Gallagher, “Analysis of pedestrian gait andperceptionreaction at signalcontrolledcrosswalk intersections,” Transportation Research Record, vol. 1, 2000.

[15] M. Goldhammer, A. Hubert, S. Kohler, K. Zindler, U. Brunsmann, ¨ K. Doll, and B. Sick, “Analysis on termination of pedestrians gait at urban intersections,” in Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, 2014, pp. 1758 – 1763.

[16] N. Schneider and D. M. Gavrila, Pedestrian Path Prediction with Recursive Bayesian Filters: A Comparative Study. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 174–183. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40602-7 18

[17] C. G. Keller and D. M. Gavrila, “Will the pedestrian cross? a study on pedestrian path prediction,” IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 2, pp. 494–506, April 2014.

[18] S. Koehler, M. Goldhammer, S. Bauer, S. Zecha, K. Doll, U. Brunsmann, and K. Dietmayer, “Stationary detection of the pedestrian?s intention at intersections,” IEEE Intelligent Transportation Systems Magazine, vol. 5, no. 4, pp. 87–99, winter 2013.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Computer Vision and Pattern Recognition (CVPR), 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” CoRR, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] M. Lin, Q. Chen, and S. Yan, “Network in network,” CoRR, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[22] 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.

[23] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A ´ benchmark,” in CVPR, June 2009.

[24] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” Int. J. Comput. Vision, vol. 88, no. 2, pp. 303–338, June 2010. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0275-4

[25] 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 2005), 20-26 June 2005, San Diego, CA, USA, 2005, pp. 886–893. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.177

[26] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, High Accuracy Optical Flow Estimation Based on a Theory for Warping. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 25–36. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24673-2 3

[27] G. Farneback, ¨ Two-Frame Motion Estimation Based on Polynomial Expansion. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363– 370. [Online]. Available: http://dx.doi.org/10.1007/3-540-45103-X 50

[28] G. R. Bradski and J. Davis, “Motion segmentation and pose recognition with motion history gradients,” in Applications of Computer Vision, 2000, Fifth IEEE Workshop on. IEEE, 2000, pp. 238–244.

[29] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing coadaptation of feature detectors,” CoRR, vol. abs/1207.0580, 2012.

[30] Y. Nesterov, “A method for solving a convex programming problem with rate of convergence O(1/k2),” Soviet Math. Doklady, vol. 269, no. 3, pp. 543–547, 1983.

[31] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics, 2010.

[32]https://www.google.com/search?q=histograms+of+oriented+gradients+algorithm&safe=off&source=lnms&tbm=isch&sa=X&ved=0ahUKEwizkP6Jz93jAhUF73MBHTaHBuQQ_AUIESgB&biw=1366&bih=657#imgrc=3aIsLIbsOEpOKM:

[33]https://www.google.com/search?q=cnn+filters&safe=off&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi0me_rz93jAhU27nMBHWTlD9QQ_AUIESgB&biw=1366&bih=657

[35]https://www.google.com/search?safe=off&biw=1366&bih=657&tbm=isch&sa=1&ei=gr9AXazFKcuWwgPV3aS4DA&q=optical+flow+estimation&oq=optical+flow+&gs_l=img.1.9.35i39j0l9.3916.5644..10921...0.0..0.207.1234.0j7j1......0....1..gws-wiz-img.......0i24j0i8i30.UCkvEStyC5Q

[36]https://www.google.com/search?q=svm&safe=off&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjoxPzL1N3jAhW28HMBHcMQDxEQ_AUIEigC&biw=1366&bih=657

[37]https://www.google.com/search?safe=off&biw=1366&bih=657&tbm=isch&sa=1&ei=jr9AXe37NMXKvgSPybroBQ&q=caltech+pedestrian+dataset&oq=caltech+dat&gs_l=img.3.0.0i8i30j0i24l9.320842.327175..329346...0.0..2.818.3994.0j13j5-1j2......0....1..gws-wiz-img.....0..35i39j0j0i67.H_MGbp40OJI