PROJECT REPORT

ON

"REAL-TIME HUMAN ACTION RECOGNITION"

Submitted to

ADAMAS UNIVERSITY

In Partial Fulfilment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING

BY

ANIRBAN SAHA	**2015ETCS002**
SANJOY GOSWAMI	**2015ETCS020**
SAYANTAN ROY	**2015ETCS022**
SOUVIK GHOSH	**2015ETCS025**

UNDER THE GUIDANCE OF
MR. BIBHAS DAS
(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ADAMAS UNIVERSITY
BARASAT, KOLKATA - 700126

2018-2019

Undertaking

We declare that this report on the project titled "**Real-Time Human Action Recognition**" submitted to the Computer Science & Engineering Department, School of Engineering and Technology, Adamas University, Barasat is based on our original work. We have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, we accept that our degree may be unconditionally withdrawn.

Date: 18/05/2019
Place: Adamas University, Barasat

(Anirban Saha)
2015ETCS002

(Sanjoy Goswami)
2015ETCS020

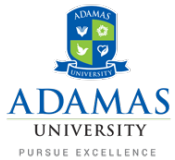
(Sayantan Roy)
2015ETCS022

(Souvik Ghosh)
2015ETCS025

ADAMAS UNIVERSITY

Department of Computer Science & Engineering

BARASAT, KOLKATA 700126



CERTIFICATE

This is to certify that the project entitled

"Real-Time Human Action Recognition"

submitted by

ANIRBAN SAHA	**2015ETCS002**
SANJOY GOSWAMI	**2015ETCS020**
SAYANTAN ROY	**2015ETCS022**
SOUVIK GHOSH	**2015ETCS025**

is a bonafide work carried out by them, in the partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (Computer Science & Engineering) at Adamas University, Barasat. This work is done during the year 2018-2019, under our guidance and has not formed the basis of any other degree.

Date: 18/05/2019

(Mr. BIBHAS DAS)
Project Guide

(Prof. (Dr.) AMITAVA MUKHERJEE)
HOD, CSE Department

Acknowledgement

We are profoundly grateful to **Mr. Bibhas Das** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its current stage.

We would like to express deepest appreciation towards **Prof. (Dr.) Amitava Mukherjee**, Head of Department of Computer Science & Engineering, Adamas University, whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the faculty members of Computer Science & Engineering Department who helped us directly or indirectly during this course of work.

(Anirban Saha)
2015ETCS002

(Sanjoy Goswami)
2015ETCS020

(Sayantan Roy)
2015ETCS022

(Souvik Ghosh)
2015ETCS025

ABSTRACT

Human Action Recognition is a challenging area of research in the field of computer vision due to its complex analytical structure. Many research have been carried out in this field over the past few years. Now, as it has become possible to recognize human action successfully, the researchers are trying to make the recognition system real-time in nature. But due to its complex video analysis, it has not been possible to make it real-time completely till date. Though some algorithms had been developed that will successfully recognize human action with the minimum delay possible in order to make the recognition system almost real-time in nature.

Once completed, this work will demonstrate an architecture that will be self sufficient to recognize human action in real-time. This will lead to a complete real-time system that is capable to recognize human action. This system can be utilized in large number of critical application including surveillance, medical, man-machine interfaces, robotics, etc. Utilization of this system in other applications, which includes further processing of the recognized human action, also cannot be ignored as it will speed up the processing of the required system.

Keywords: action recognition, human motion, recognition model, real-time human action, convolutional neural network

Contents

Chapter 1

Introduction

In this chapter, we are going to discuss a brief history of Artificial Intelligence and its related domains. We will also then have a quick look on the technologies that made machine capable of visualizing its environment. Through this discussion, we will get into the need and evolution of Human Action Recognition techniques as an overview, which will be discussed in the upcoming chapters.

1.1 Artificial Intelligence making its way out

Recent buzzing term "Artificial Intelligence" was first coined at a conference at Dartmouth College in Hanover, New Hampshire in the year 1956. It was defined as intelligence demonstrated by machines, in contrast to natural intelligence displayed by humans and animals. But achieving an artificially intelligent being wasn't so simple. After several reports criticizing progress in AI, government funding and interest in the field dropped off - a period from 1974-80 that became known as the "AI winter". The field later revived in the 1980s when the British government started funding it again in part to compete with efforts by the Japanese.

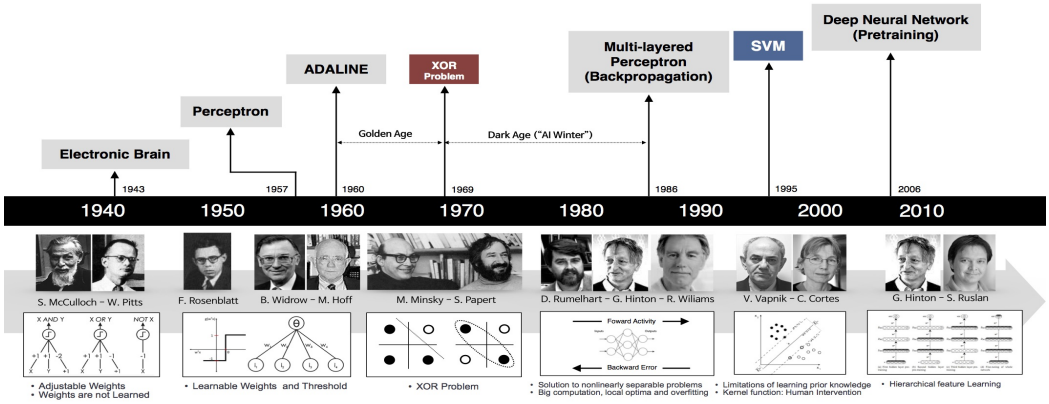


Figure 1.1: Artificial Intelligence Timeline

The urge of humans to make a machine capable to mimic all the natural behaviours of mankind has led to several complicated fields of study. Each of these fields helps in copying different parts of human intelligence. One of these areas include the image analysis perspectives of different intelligent systems which is considered to be the most complicated domain till date. This domain includes a wide range of topics starting from Computer Graphics, Image Processing, Machine Learning, Deep Learning, Computer Vision, and many more. Being the most complicated domain of AI, this is improving rapidly in recent days and has attracted many people including mathematicians and Computer Scientists towards it.

1.2 Computer Graphics, Image Processing and Computer Vision

Being the subjects under the same domain of Artificial Intelligence, these are intended to cover different interdependent aspects of an intelligent system. Computer Graphics, beside providing a basic platform, deals with preparing a sketch of the description given to the system as an input. Image Processing, on the other hand, focuses on different operations that can be performed over images in order to enhance the quality or extract certain useful properties of the image provided. Whereas, Computer Vision, being the most complicated but most focused area, concerns with modeling and replicating human vision using computer software and hardware.

In simple words, Computer Graphics converts a textual information to image. Image Processing manipulates a image where both input and output information are images. And Computer Vision tries to visualize a image and provide certain relevant textual information.

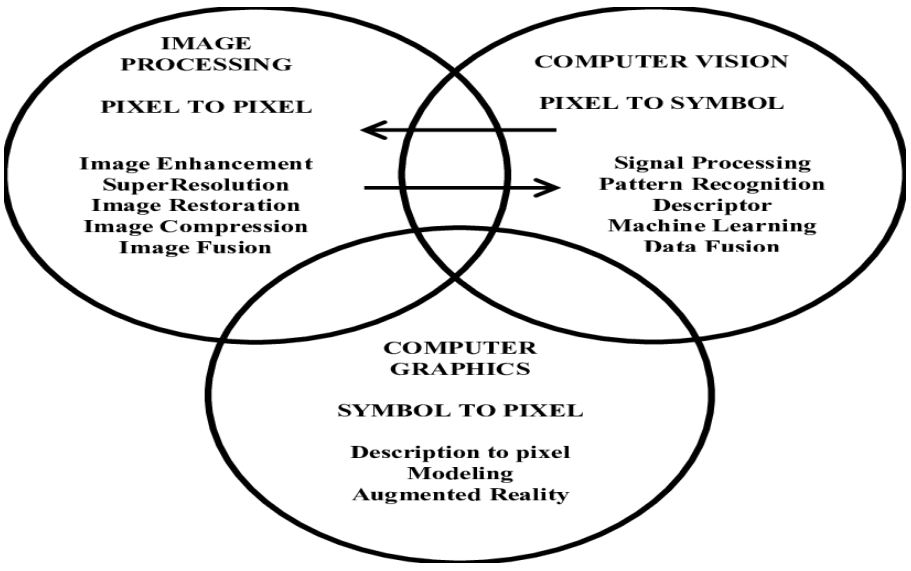


Figure 1.2: Interdependent Image Analysis Domains

1.3 Computer Vision: A Brief Overview

Computer vision is a discipline that studies how to reconstruct, interrupt and understand a 3d scene from its 2d images in terms of the properties of the structure present in scene. It is divided into three basic categories that are as following:

- **Low-level vision**: includes process image for feature extraction.
- **Intermediate-level vision**: includes object recognition and 3D scene Interpretation
- **High-level vision**: includes conceptual description of a scene like activity, intention and behavior.

Figure 1.3: Introduction to Computer Vision

It has huge applications in technology development fields in recent era, such as Robotics, Medical, Security, Transportation and Industrial Automation, to name a few. Application of Computer Vision in Robotics includes Localization, Navigation, Human-Robot Interaction (HRI), etc. Medical Science takes the help of Computer Vision in diagnostics, 3D human organ reconstruction, Vision-guided robotics surgery, and in many more fields. Whereas, Security, being the concerned area, adopts Computer Vision techniques for Bio-metrics, Surveillance, etc.One of the major surveillance application of computer vision is Human Action Recognition (H.A.R.). It has attracted the attention of a huge number of researchers.

Overall tasks performed by Computer Vision algorithms includes Recognition, Motion Analysis, Scene Reconstruction, Image Restoration, etc. The system having an inbuilt computer vision module undergoes several typical functions which are as follows:

- Image Acquisition
- Pre-Processing
- Feature Extraction
- Detection or Segmentation
- High-level Processing
- Decision Making

1.4 Human Action Recognition: A Brief Idea

Human Action Recognition is an area of study in the field of computer vision which aims to analyse video or image sequence in order to classify different actions performed by human that are detected in the given video or image sequence. In other words, it aims to track, understand, and analyze human movements that is continuously captured by sensors.

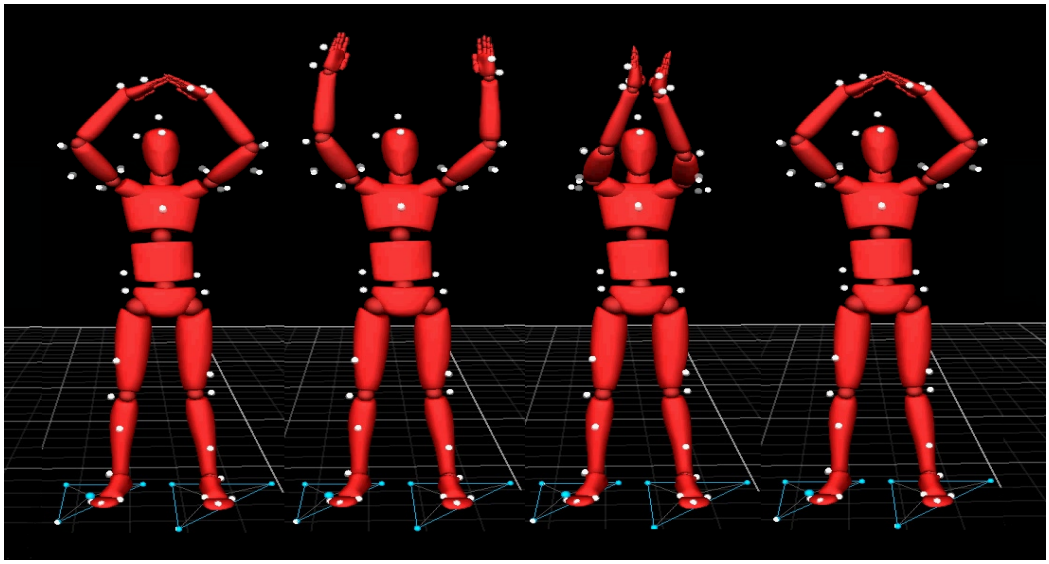


Figure 1.4: Human Action Analysis

Chapter 2

Human Action Recognition

In this chapter, we are going to discuss an overview on the topic Human Action Recognition (H.A.R.). After getting an overview, we will discuss the hurdles in making a real-time H.A.R. system and its application domains. Then we will have a quick look on the project objective and motivation.

2.1 Familiarization with H.A.R.

H.A.R. is an interdisciplinary challenging field having grand applications with social, commercial, and educational benefits[3] which includes visual surveillance system, robotics, human-computer interaction, etc. The main challenge lies in how to design an effective human action representation that is sufficiently descriptive while computationally efficient. Because of complexity and uncertainty in Human Action Recognition, consistently accurate action identification has also remained a challenging goal.

2.2 Applications of H.A.R.

Human Action Recognition System is based on systematic approach of understanding and analysing the motion of humans in captured content. It comprises of the fields which includes Video Processing, Machine Vision, Artificial Intelligence, etc. Some of the premier applications of the Human Action Recognition System are discussed below.

Surveillance: Today different highly secured areas of any organization requires 24x7 human monitoring system. On a general case, it is observed that only after a mishap has occurred, a concerned person regrets for not analyzing the situations that had led to the mishap. In this cases, the Human Action Recognition based Surveillance System can analyse the event online and give a appropriate result using

predefined human motion and behavioural analysis which can have the real time processing capability. This has a high potential application in military, security, and commercial domain.

Robotics: In recent time, robotics is one of the major area of concern. Researchers are in the verge of developing advanced robotic system that can mimic almost every human activity as well as human intelligence. In this emerging field, modelling of human action or activity plays an important role. This modelling can be critical in some cases where it is necessary for the robotic system to conclude some decision based on the result of this model. So, in this case, the accuracy of the model plays a vital role.

Sports Analysis: It is widely used by the sports persons and the team committee members to understand and analyse the pattern of different players across the world. Most of the sports analysis are being done manually till date which sometimes lead to some unavoidable miscalculations. So some research committees are trying hard to automate this analysis system. In order to do so, human action recognition technique is highly important.

There exists varied applications having various representation of human motion along with different recognition model. Human Action Recognition is a general term and it is the application that decides duration of the movement of body along with the parts involved. Human to computer interaction generally involves only hand movement whereas complicated application requires whole body movement such as in sports. Human motion is divided into gestures, activity, action, interaction and group activity depending on the complexity. Approach of 2D representation are in stick figure or kinematic where as 3D representation is defined by shape model, image model and kinematics. Normal recognition requires single layered approach whereas complex recognition requires multi layered approach.

2.3 Real-Time Human Action Recognition

Human Action Recognition involves analysis over a range of image frames in order the arrive at a particular conclusion. It is unable to conclude a particular action by processing a single frame. Due to this analysis over multiple image frames, a short latency is inherent in every such models which is proportional to the constant number of continuous image frames that is taken into consideration to detect a particular action in a given model.

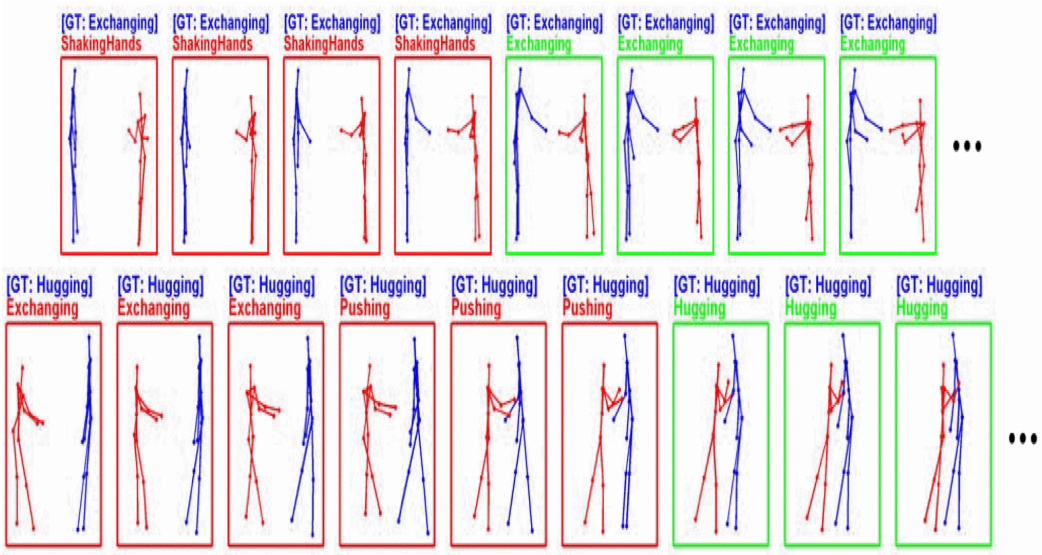


Figure 2.1: Real-Time Human Action

Present day research on real-time human action recognition aims to reduce the recognition latency to such a extent that it becomes almost negligible for a human eye to detect (without compromising the considerable accuracy). Recent research result in this domain validates that the researchers are successful in accomplishing the desired goal. Recent systems with built-in real-time human action recognition module recognizes the action within a fraction of second.

2.4 Project Objective and Motivation

We have became familiar with the fact that human action recognition involves video analysis rather than image analysis due to which a latency (may be negligible with respect to human eye) is inherent. But it requires a huge computational processing power to process multiple image frames at a single block. Moreover, a system where H.A.R. is a sub-module, this latency proves to be a huge disadvantage.

This project aims to define a model that will perform the required video analysis for H.A.R. at the image level. This will reduce the computational cost of the present day systems. The model will process a single image frame at a time along with the previous classified output. In this way, the output of the model at a particular instance will be indirectly dependent on the previous outcomes.

This project is inspired from the natural human vision. Normally, a human cannot detect an action from an image. But there exists many instances when a human

can perform the same from a single image, especially while watching a video. One of these instance can be the example of a human watching a movie in computer. While watching the movie, if the human suddenly press the pause button of the video player, then only a single image frame is displayed on the screen. And by watching this image frame, the human can determine the present action of each and every person present in the image.

Chapter 3

Background 1: Feature Extraction

A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. Throughout this chapter, we will have a quick look on the features that can be utilized for the application of Human Action Recognition.

3.1 Local Binary Pattern

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.

The original LBP operator was defined to only deal with the spatial information. Later, it was extended to a spatiotemporal representation for dynamic texture analysis. For this purpose, the so-called Volume Local Binary Pattern (VLBP) operator was proposed by Zhao and Pietikinen in 2007. The idea behind VLBP consists of looking at dynamic texture as a set of volumes in the (X,Y,T) space where X and Y denote the spatial coordinates and T denotes the frame index (time). The neighborhood of each pixel is thus defined in three dimensional space. Then, similarly to LBP in spatial domain, volume textons can be defined and extracted into histograms. Therefore, VLBP combines motion and appearance together to describe

dynamic textures. This VLBP feature can be used to classify Human Action.

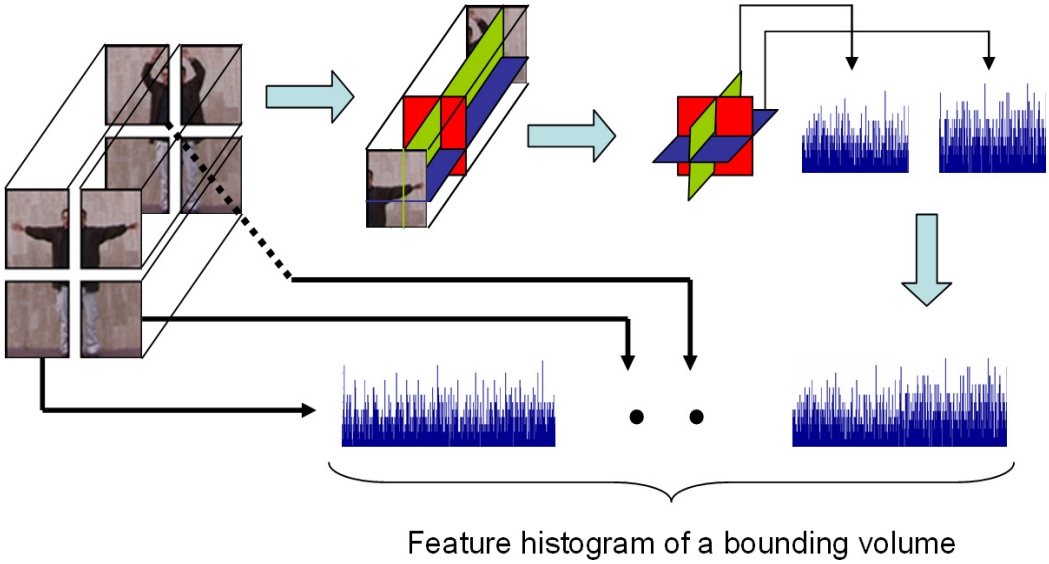


Figure 3.1: Spatiotemporal Volume Local Binary Pattern

3.2 Histogram of Oriented Gradient

The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images.

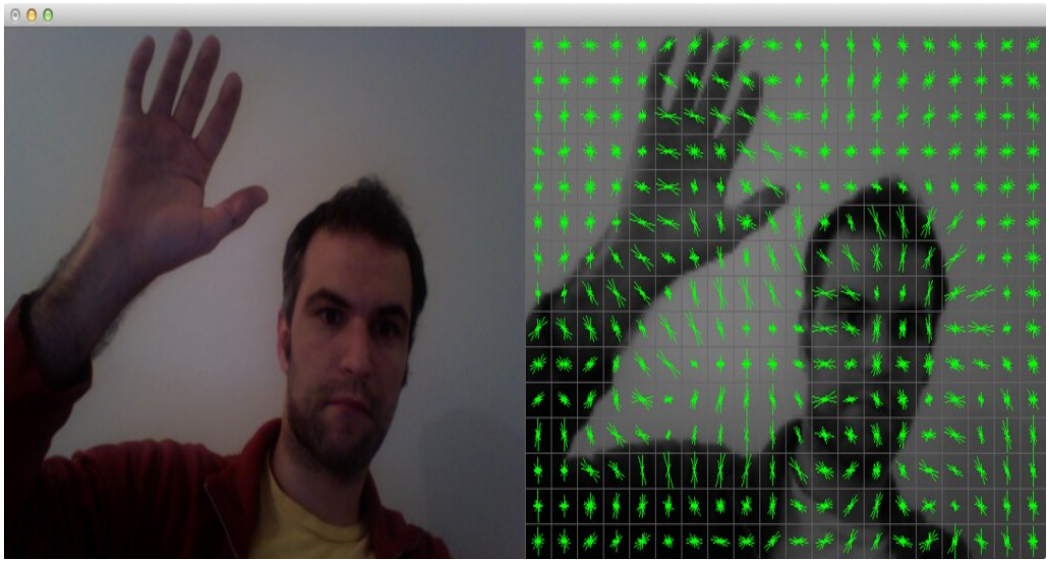


Figure 3.2: Extraction of Oriented Gradients

3.3 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:

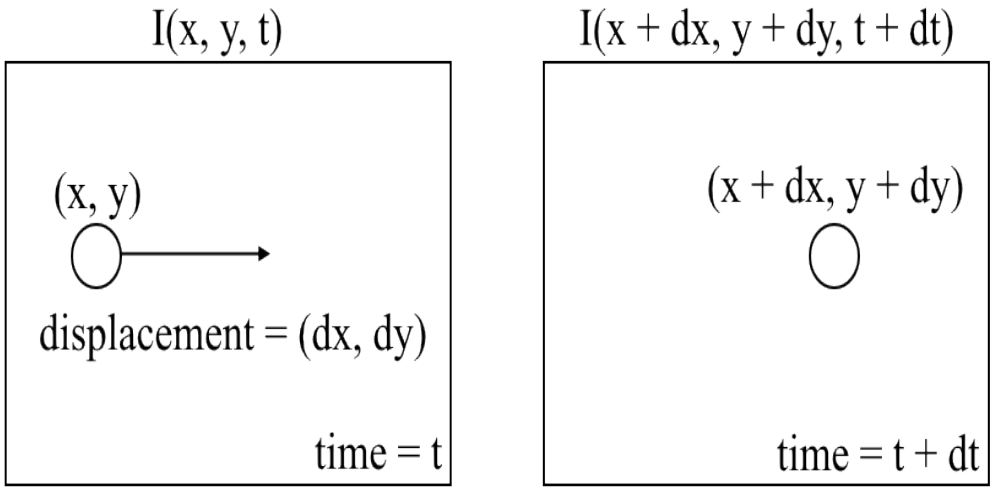


Figure 3.3: Optical Flow Problem

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.

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t) \tag{3.1}$$

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

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \ldots \tag{3.2}$$

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \tag{3.3}$$

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

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{3.4}$$

where $u = \partial x/\partial t$ and $v = \partial y/\partial t$.

$\partial I/\partial x, \partial I/\partial y$, and $\partial I/\partial t$ 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 $u(\partial x/\partial t)$ and $v(\partial y/\partial t)$ to determine movement over time.

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.

3.4 Motion Boundary Histogram

Motion Boundary Histogram (MBH) records the change in Histogram of Oriented Gradients (HOG) over consecutive frames containing the boundaries present the input frame sequence. In order to compute this feature, the input video frame is first pre-processed in order to generate the boundary image and then the HOG feature extractor is applied and the change over this feature over consecutive frames are recorded.

Chapter 4

Background 2: Classification

Classification is the process of categorizing a stimuli into a finite set of classes or labels. The process normally involves recognition of the dominant content in a scene. The dominant content gets the strongest confidence score irrespective of the transformation of that content such as scaling, location or rotation. In this chapter, we will go through a brief introduction of different learning models and classification techniques that can be adopted to recognize human actions once the features are extracted.

4.1 Learning Models

Learning is one of the fundamental building blocks of artificial intelligence (AI) solutions. From a conceptual standpoint, learning is a process that improves the knowledge of an AI program by making observations about its environment. From a technical/mathematical standpoint, AI learning processes focused on processing a collection of input-output pairs for a specific function and predicts the outputs for new inputs. Most of the artificial intelligence(AI) basic literature identifies two main groups of learning models: supervised and unsupervised. However, that classification is an oversimplification of real world AI learning models and techniques.

To understand the different types of AI learning models, we can use two of the main elements of human learning processes: knowledge and feedback. From the knowledge perspective, learning models can be classified based on the representation of input and output data points. In terms of the feedback, AI learning models can be classified based on the interactions with the outside environment, users and other external factors.

4.1.1 Knowledge Based Learning

Factoring its representation of knowledge, AI learning models can be classified in two main types: inductive and deductive.

- **Inductive Learning**: This type of AI learning model is based on inferring a general rule from datasets of input-output pairs. Algorithms such as knowledge based inductive learning(KBIL) are a great example of this type of AI learning technique. KBIL focused on finding inductive hypotheses on a dataset with the help of background information.
- **Deductive Learning**: This type of AI learning technique starts with te series of rules nad infers new rules that are more efficient in the context of a specific AI algorithm. Explanation-Based Learning(EBL) and Relevance-0 Based Learning(RBL) are examples examples o f deductive techniques. EBL extracts general rules from examples by "generalizing" the explanation. RBL focuses on identifying attributes and deductive generalizations from simple example.

4.1.2 Feedback Based Learning

Based on the feedback characteristics, AI learning models can be classified as supervised, unsupervised, semi-supervised or reinforced.

- **Unsupervised Learning**: Unsupervised models focus on learning a pattern in the input data without any external feedback. Clustering is a classic example of unsupervised learning models.
- **Supervised Learning**: Supervised learning models use external feedback to learning functions that map inputs to output observations. In those models the external environment acts as a teacher of the AI algorithms.
- **Semi-Supervised Learning**:Semi-Supervised learning uses a set of curated, labeled data and tries to infer new labels/attributes on new data data sets. Semi-Supervised learning models are a solid middle ground between supervised and unsupervised models.
- **Reinforcement Learning**: Reinforcement learning models use opposite dynamics such as rewards and punishment to reinforce different types of knowledge. This type of learning technique is becoming really popular in modern AI solutions.

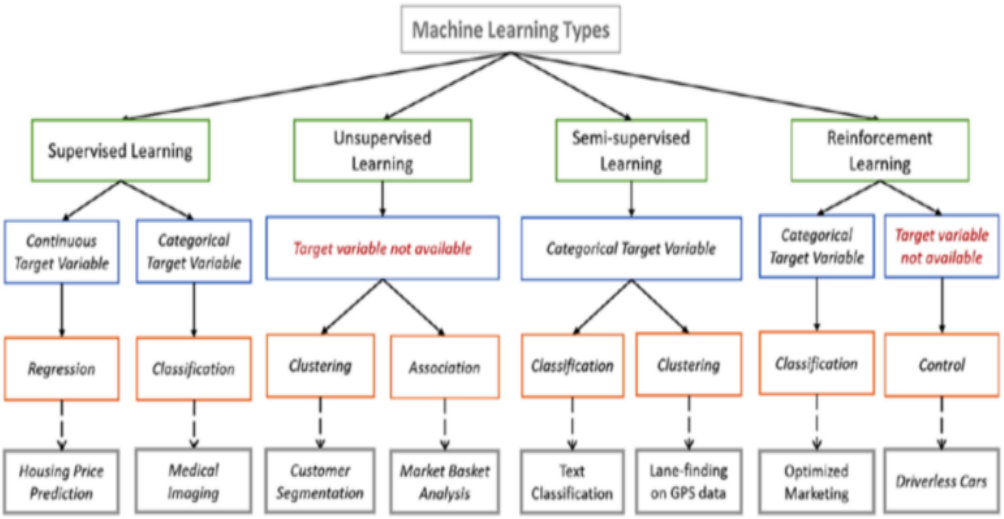


Figure 4.1: Feedback based Learning Model Classification

4.2 Single Layer Perceptron

Rosenblatts single layer perceptron is one of the earliest models for learning. Our goal is to find a linear decision function parametrized by the weigh vector W and bias parameter b. Learning occurs by making small adjustments to these parameters every time the predicted label $\hat{y}$ mismatches the true label y_i of an input data point x_i. In particular, the perceptron implements the following function to solve the classification problem:

$$\hat{y} = f(\sum_{i=1}^{d} w_i \cdot x_i + b) \tag{4.1}$$

where f is a non-linear function and d is the number of features or the dimensionality of the problem. As we can see, changing the values of w_i and b gives us different functions, and thus, it defines a collection of functions. Target of learning is to find out the W and b that best capture the input-output relation.

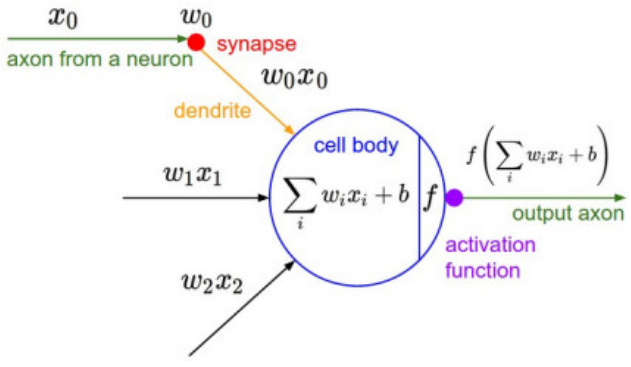


Figure 4.2: Structure of Single Layer Perceptron

4.3 Multi Layer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. A MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

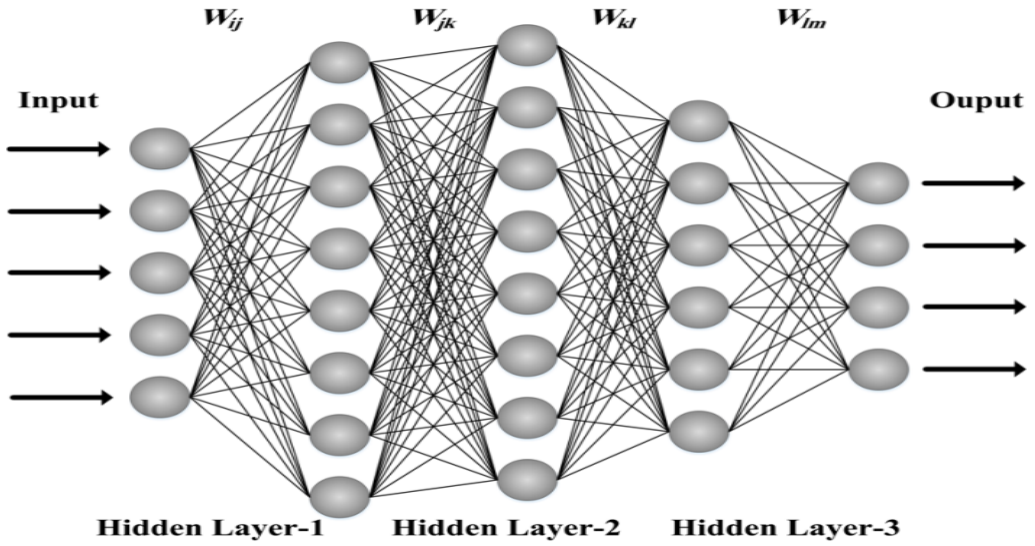


Figure 4.3: Sample model of Multilayer Perceptron

4.3.1 Learning in Multilayer Perceptron

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation, a generalization of the least mean squares algorithm in the linear perceptron.

We represent the error in output node j in the n^{th} data point (training example) by $e_j(n) = d_j(n) - y_j(n)$, where d is the target value and y is the value produced by the perceptron. The node weights are adjusted based on corrections that minimize the

error in the entire output, given by

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{4.2}$$

Using gradient descent, the change in each weight is

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \tag{4.3}$$

where y_i is the output of the previous neuron and η is the learning rate, which is selected to ensure that the weights quickly converge to a response, without oscillations.

The derivative to be calculated depends on the induced local field v_j, which itself varies. It is easy to prove that for an output node this derivative can be simplified to

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n)) \tag{4.4}$$

where ϕ' is the derivative of the activation function described above, which itself does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = \phi'(v_j(n))\sum_k -\frac{\partial \varepsilon(n)}{\partial v_k(n)} w_{kj}(n) \tag{4.5}$$

This depends on the change in weights of the k^{th} nodes, which represent the output layer. So to change the hidden layer weights, the output layer weights change according to the derivative of the activation function, and so this algorithm represents a backpropagation of the activation function.

4.4 Support Vector Machine

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. Equation of the decision boundary is given by

$$\underline{W}^T \underline{X} + b = 0 \tag{4.6}$$

where $\underline{W}$ is the weight vector, $\underline{X}$ is the feature vectors and b is constant term independent of feature components. In two-dimensional space the Decision Boundary

will be a Line. And in three-dimensional space the Decision Boundary will be a Plane. Whereas, in n-dimensional space (n¿3) the Decision Boundary will be a Hyper Plane.

Margin of Separation: Margin of a linear classifier is defined as the width that the boundary that could be increased by before hitting a data point (Feature Vector). SVM aims to determine the line corresponding to maximum margin of separation.

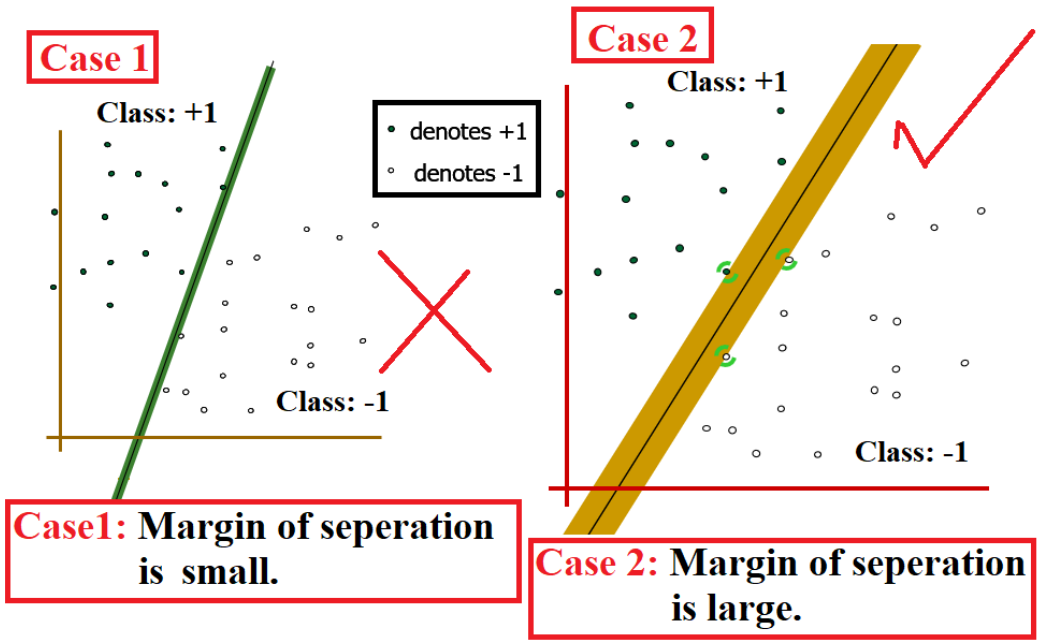


Figure 4.4: Separator Line in SVM

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors (data points or feature vector closest to the margin of separation) are important; other training examples are ignorable.
- Empirically it works very well.

For all training samples $\underline{X}^i$ corresponding to class label $y_i = 1$:

$$\underline{W}^T \underline{X}^i + b \geq 1 \tag{4.7}$$

For all training samples $\underline{X}^i$ corresponding to class label $y_i = -1$:

$$\underline{W}^T \underline{X}^i + b \leq -1 \tag{4.8}$$

Combining the above two equations, we have

$$y_i[\underline{W}^T \underline{X}^i + b] \geq 1 \quad \forall i = 1,2,3,\ldots,N \tag{4.9}$$

where N is the total number of training samples. And the equation of the separator hyperplane is:

$$\underline{W}^T \underline{X} + b = 0 \tag{4.10}$$

Unit Vector normal to the hyperplane:

$$\hat{n} = \frac{\nabla_{\underline{W}}[\underline{W}^T \underline{X} + b]}{\| \nabla_{\underline{W}}[\underline{W}^T \underline{X} + b] \|} = \frac{\underline{W}}{\| \underline{W} \|} = \hat{\underline{W}} \tag{4.11}$$

where

$$\Delta_{\underline{W}} \equiv \begin{bmatrix} \frac{\partial}{\partial w_1} \\ \frac{\partial}{\partial w_2} \\ \frac{\partial}{\partial w_3} \\ \vdots \\ \frac{\partial}{\partial w_n} \end{bmatrix} \tag{4.12}$$

As per the mathematical definition of support vectors we have:

$$\begin{aligned} \underline{W}^T \underline{X}^+ + b &= +1 \quad \text{if } y_i = +1 \\ \underline{W}^T \underline{X}^- + b &= -1 \quad \text{if } y_i = -1 \end{aligned} \tag{4.13}$$

Subtracting the above equations, we get

$$\underline{W}^T (\underline{X}^+ - \underline{X}^-) = 2 \tag{4.14}$$

Projection of the vector $\underline{X}^+ - \underline{X}^-$ on the normal $\hat{\underline{W}}$ to the required hyper-plane is expressed as:

$$d_m = \hat{n}^T (\underline{X}^+ - \underline{X}^-) = \frac{\underline{W}^T (\underline{X}^+ - \underline{X}^-)}{\| \underline{W} \|} = \frac{2}{\| \underline{W} \|} \tag{4.15}$$

where d_m is the width of margin of separation which need to be maximized. Maximization of $\frac{2}{\|\underline{W}\|} \equiv$ Minimization of $\| \underline{W} \| \equiv$ Minimization of $\frac{1}{2} \| \underline{W} \|^2 \equiv$ Minimization of $\frac{1}{2} \underline{W}^T \underline{W}$

Now the problem is a simple constraint based mathematical problem defined as Minimization of $\frac{1}{2} \underline{W}^T \underline{W}$ subject to constraint $y_i[\underline{W}^T \underline{X}^i + b] \geq 1 \quad \forall i = 1, 2, 3, \ldots, N$. Applying Lagrange Method, we get a lagrangian function

$$L(\underline{W}, b, \underline{\Lambda}) = \frac{1}{2} \underline{W}^T \underline{W} - \sum_{i=1}^{N} \lambda_i [y_i (\underline{W}^T \underline{X}^i + b) - 1] \tag{4.16}$$

where $\underline{\Lambda} = [\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_N]^T \in i^N$ and λ_is are lagrangian multipliers.
Now the problem becomes a Primal Un-constrained Min-Max Problem defined as:

$$Min_{(\underline{W},b)}[Max_{\underline{\Lambda}} L(\underline{W}, b, \underline{\Lambda})] \tag{4.17}$$

Min-Max operation can be interchanged in case of Strong Duality. Hence the Dual Problem will be the following:

$$Max_{\underline{\Lambda}}[Min_{(\underline{W},b)} L(\underline{W}, b, \underline{\Lambda})] \tag{4.18}$$

Solving this dual problem, we get the optimal value of weight vector denoted as $\underline{W}^*$ and expressed as:

$$\underline{W}^* = \sum_{i=1}^{N} \lambda_i^* y_i \underline{X}^{(i)} \tag{4.19}$$

And the optimal b, denoted as b^*, is derived from the equation:

$$\underline{W}^{*T} \underline{X}^{(k)} + b^* = y_k \tag{4.20}$$

where $k = ArgMax_i[\lambda_i^*]$.

Now the trained SVM predicts the class label by the expression:

$$y = sign(\underline{W}^{*T} \underline{X} + b^*) \tag{4.21}$$

4.5 Random Forest

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of decision trees. In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

Decision tree concept is more to the rule based system. Given the training dataset with targets and features, the decision tree algorithm will come up with some set of rules. The same set rules can be used to perform the prediction on the test dataset. In decision tree algorithm calculating these nodes and forming the rules will happen using the information gain and gini index calculations. In random forest algorithm, Instead of using information gain or gini index for calculating the root node, the

process of finding the root node and splitting the feature nodes will happen randomly. Will look about in detail in the coming section.

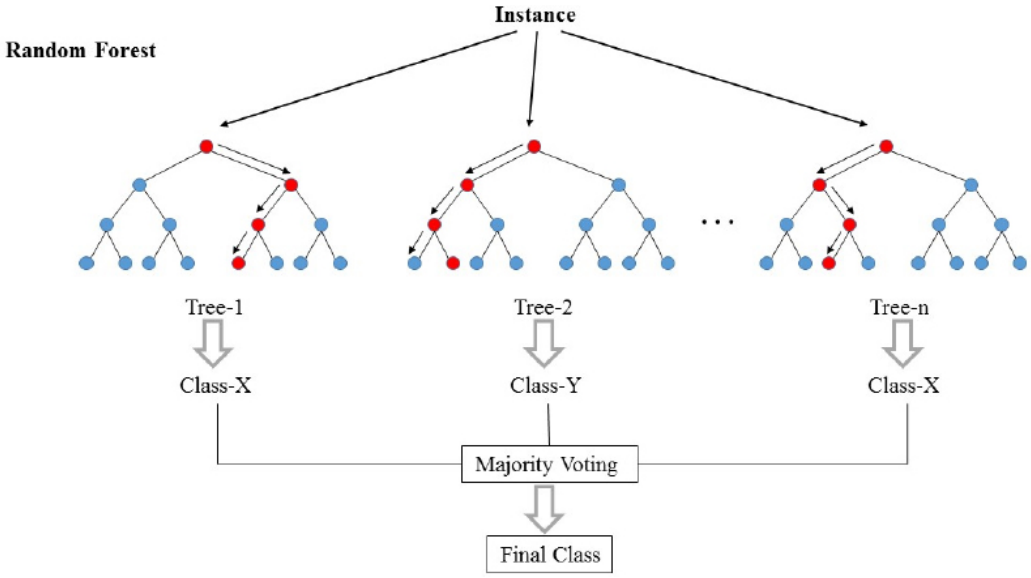


Figure 4.5: Sample Random Forest Classifier

4.5.1 Training and Prediction

The beginning of random forest algorithm starts with randomly selecting "k" features out of total "m" features. In this stage, features and observations are randomly taken. In the next stage, "k" features are randomly selected to find the root node by using the best split approach. The third stage includes calculation of the daughter nodes using the same best split approach and repeat these three steps to form the tree with a root node and having the target as the leaf node. This procedure is continuously repeated to create "n" randomly created trees. This randomly created trees forms the random forest.

To perform prediction using the trained random forest, the following steps are considered:

- Takes the test features and use the rules of each randomly created decision tree to predict the oucome and stores the predicted outcome (target)
- Calculate the votes for each predicted target.
- Consider the high voted predicted target as the final prediction from the random forest algorithm.

4.5.2 Advantages

- The same random forest algorithm or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will handle the missing values.
- When we have more trees in the forest, random forest classifier wont overfit the model.
- Random forest classifier can be modelled for categorical values also.

4.6 Multiple Instance Learning

In machine learning, multiple-instance learning (MIL) is a type of supervised learning. Instead of receiving a set of instances which are individually labeled, the learner receives a set of labeled bags, each containing many instances. In the simple case of multiple-instance binary classification, a bag may be labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one instance in it which is positive. From a collection of labeled bags, the learner tries to either (i) induce a concept that will label individual instances correctly or (ii) learn how to label bags without inducing the concept.

Depending on the type and variation in training data, machine learning can be roughly categorized into three frameworks: supervised learning, unsupervised learning, and reinforcement learning. Multiple instance learning (MIL) falls under the supervised learning framework, where every training instance has a label, either discrete or real valued. MIL deals with problems with incomplete knowledge of labels in training sets. More precisely, in multiple-instance learning, the training set consists of labeled bags, each of which is a collection of unlabeled instances. A bag is positively labeled if at least one instance in it is positive, and is negatively labeled if all instances in it are negative. The goal of the MIL is to predict the labels of new, unseen bags.

The mathematical description of the problem is that if the space of instances is $\mathcal{X}$, then the set of bags is the set of functions $N^{\mathcal{X}} = \{B : \mathcal{X} \to N\}$, which is isomorphic to the set of multi-subsets of $\mathcal{X}$. For each bag $B \in N^{\mathcal{X}}$ and each instance $x \in \mathcal{X}$, $B(x)$ is viewed as the number of times x occurs in B. Let $\mathcal{Y}$ be the space of labels, then a "multiple instance concept" is a map $c : N^{\mathcal{X}} \to \mathcal{Y}$. The goal of MIL is to learn such

a concept. The remainder of the article will focus on binary classification, where $\mathcal{Y} = \{0, 1\}$.

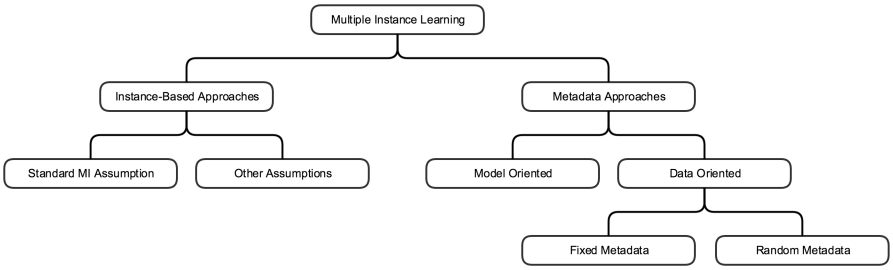


Figure 4.6: Appraoches of Multiple Instance Learning

4.7 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization includes adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. These are further discussed in this section.

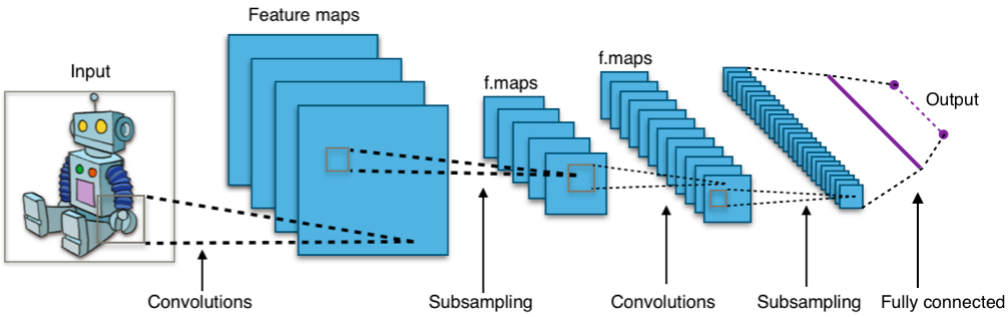


Figure 4.7: Typical CNN Architecture

- **Convolutional layer**: The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.
  Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

- **Local connectivity**: When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.
  The extent of this connectivity is a hyper-parameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.

- **Spatial arrangement**: The spatial size of the output volume can be computed as a function of the input volume size W, the kernel field size of the convolutional layer neurons K, the stride with which they are applied S, and the amount of zero padding P used on the border. The formula for calculating how many neurons "fit" in a given volume is given by $\frac{W-K+2P}{S}+1$. If this number is not an integer, then the strides are incorrect and the neurons cannot be tiled to fit across the input volume in a symmetric way. In general, setting zero padding to be $P = (K+1)/2$ when the stride is $S = 1$ ensures that the input volume and output volume will have the same size spatially. However, it's not always completely necessary to use all of the neurons of the previous layer. A neural network designer may decide to use just a portion of padding.

- **Parameter sharing**: A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on one reasonable assumption: if a patch feature is useful to compute at some spatial position, then it should also be useful to compute at other positions. In other words, denoting a single 2-dimensional slice of depth as a depth slice, we constrain the neurons in each depth slice to use the same weights and bias.
  Since all neurons in a single depth slice share the same parameters, the forward pass in each depth slice of the convolutional layer can be computed as a convolution of the neuron's weights with the input volume. Therefore, it is common to refer to the sets of weights as a filter (or a kernel), which is convolved with the input. The result of this convolution is an activation map, and the set of activation maps for each different filter are stacked together along the depth dimension to produce the output volume. Parameter sharing contributes to the translation invariance of the CNN architecture.
  Sometimes, the parameter sharing assumption may not make sense. This is especially the case when the input images to a CNN have some specific centered structure; for which we expect completely different features to be learned on different spatial locations. One practical example is when the inputs are faces that have been centered in the image: we might expect different eye-specific or hair-specific features to be learned in different parts of the image. In that case it is common to relax the parameter sharing scheme, and instead simply call the layer a "locally connected layer".

- **Pooling layer**: Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the

input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum.
Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.
The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 22 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.
In addition to max pooling, pooling units can use other functions, such as average pooling or l2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to max pooling, which performs better in practice.

- **ReLU layer**: ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function $f(x) = max(0,x)$. It effectively removes negative values from an activation map by setting them to zero.[56] It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.
  Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = \tanh(x)$, $f(x) = \| \tanh(x) \|$, and the sigmoid function $\sigma(x) = (1+e^{-x})^{-1}$. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

- **Fully connected layer**: Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of

a learned or fixed bias term).

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

4.8 Autoencoder

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Recently, the autoencoder concept has become more widely used for learning generative models of data. Some of the most powerful AIs in the 2010s involved sparse autoencoders stacked inside of deep neural networks.

An autoencoder learns to compress data from the input layer into a short code, and then uncompress that code into something that closely matches the original data. This forces the autoencoder to engage in dimensionality reduction, for example by learning how to ignore noise. Some architectures use stacked sparse autoencoder layers for image recognition. The first encoding layer might learn to encode basic features such as corners, the second to analyze the first layer's output and then encode less local features like the tip of a nose, the third might encode a whole nose, etc., until the final encoding layer encodes the whole image into a code that matches. The decoding layers learn to decode the representation back into its original form as closely as possible. An alternative use is as a generative model.

4.8.1 Structure

The simplest form of an autoencoder is a feedforward, non-recurrent neural network similar to single layer perceptrons that participate in multilayer perceptrons

(MLP) having an input layer, an output layer and one or more hidden layers connecting them where the output layer has the same number of nodes (neurons) as the input layer, and with the purpose of reconstructing its inputs (minimizing the difference between the input and the output) instead of predicting the target value Y given inputs X. Therefore, autoencoders are unsupervised learning models (do not require labeled inputs to enable learning).

An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and ψ, such that:

$$\phi : \mathcal{X} \rightarrow \mathcal{F} \tag{4.22}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X} \tag{4.23}$$

$$\phi, \psi = argmin_{\phi,\psi} \| X - (\phi \odot \psi) X \|^2 \tag{4.24}$$

4.8.2 Training

The training algorithm for an autoencoder can be summarized as:

- For each input x,
  - Do a feed-forward pass to compute activations at all hidden layers, then at the output layer to obtain an output $x^{'}$,
  - Measure the deviation of $x^{'}$ from the input x,
  - Backpropagate the error through the net and perform weight updates.

An autoencoder is often trained using one of the many variants of backpropagation (such as conjugate gradient method, steepest descent, etc.). Though these are often reasonably effective, there are fundamental problems with the use of backpropagation to train networks with many hidden layers. Once errors are backpropagated to the first few layers, they become minuscule and insignificant. This means that the network will almost always learn to reconstruct the average of all the training data. Though more advanced backpropagation methods (such as the conjugate gradient method) can solve this problem to a certain extent, they still result in a very slow learning process and poor solutions. This problem can be remedied by using initial weights that approximate the final solution. The process of finding these initial weights is often referred to as pretraining.

Chapter 5

Literature Survey

In this chapter, we will briefly discuss different research approaches for recognizing human action. Then we will explore the recent approaches of real-time human action recognition. In this way, this chapter will familiarize us with the recent research trends that the researchers are exploring in this area.

5.1 Survey on Human Action Recognition

In this section, a brief survey of some of the concepts proposed by different research team working in the field of human action recognition is discussed. All the methods discussed in this section has been taken from [2] and verified accordingly. This will provide a clear idea about the techniques that has been explored in the field of Human Action Recognition which are not real-time in nature.

5.1.1 Method 1

In the approach proposed in [1], Multiple Instance Learning is utilized for subsequent action classification. Features such as trajectory, HOG, HOF, and MBH is calculated and bag of feature encoding technique is used followed by SVM classifier to accomplish the desired goal.
It was implemented on the dataset entitled Hollywood2 and was successful to provide an accuracy of 57.42% and 59.80% for full sequence and combined sequence classifier respectively. The limitation of this technique is that it fails when full body motion is not clearly visible.

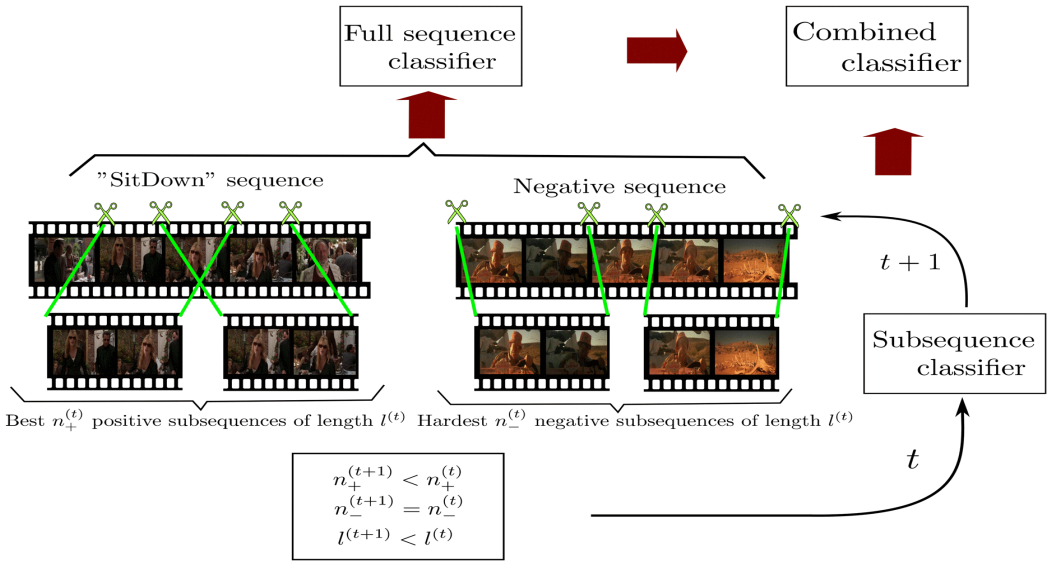


Figure 5.1: Method 1 Sketch Analysis

5.1.2 Method 2

The framework proposed in [8] uses three layer Convolutional Neural Network along with Independent Subspace Analysis and Principal Component Analysis for recognizing human interaction from segmented and unsegmented videos.
It was implemented on the dataset named UT-Interaction and proved to be one of the best technique by giving an accuracy of 90% - 93% when used over segmented videos and 85% - 90% when used over unsegmented videos. But spatial and temporal localization of the recognized activities are not possible using this approach.

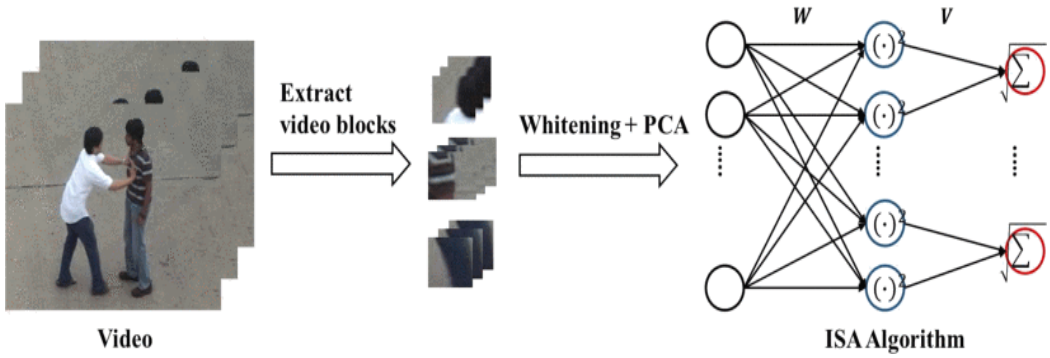


Figure 5.2: Method 2 Sketch Analysis

5.1.3 Method 3

The method described in [4] uses two main models which are termed as attribute model and interaction model. Besides this model, an extra layer of Data Driven

Phrase (DDP) technique is applied on the training data.

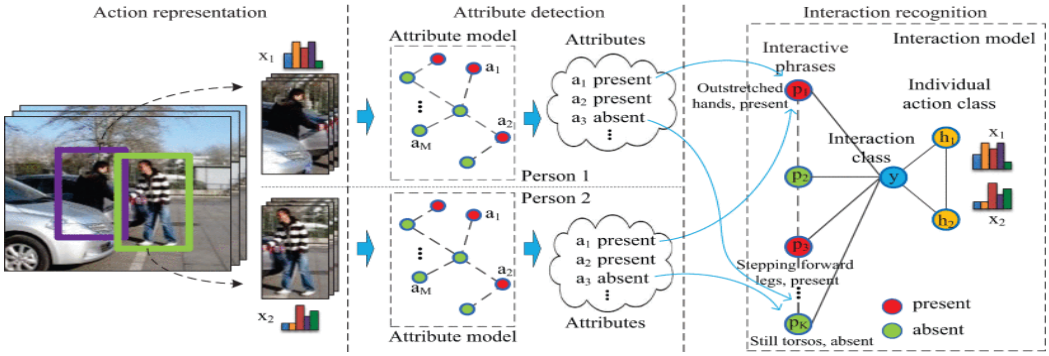


Figure 5.3: Method 3 Sketch Analysis

This method was tested on BIT-Interaction, UT-Interaction, and Collective Activity datasets and provided the accuracy of 90.63%, 91.67%, and 82.54% on the respective datasets. In this method, there is a need to manually label attributes in each video and specify connectivity between attributes and phrases. Moreover it does not consider temporal dependencies in phrases and attributes, and due to this, the system may get confused between different types of interaction.

5.1.4 Method 4

The approach described in [13] demonstrate a flow that detects the optical flow field of motion salient pixels from the region of interest followed by the computation of self-similarity matrix along with the usage of Fischer encoding technique. Then, using the computed motion interchange pattern, a support vector machine is trained to achieve the desired goal.

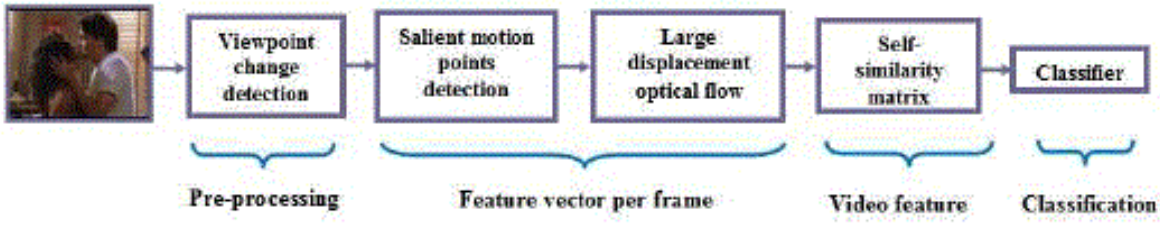


Figure 5.4: Method 4 Sketch Analysis

But this process gave a very low accuracy of 50.1% when tested on TV Human Interaction dataset.

5.1.5 Method 5

The technique used in [10] for human action recognition utilizes dense trajectories with the computation of features which included optical flow, trajectory, HOG, HOF, and MBH followed by the generation of codebook with the help of bag of features concept. Then a SVM is trained which is used for further classification.
It had resulted providing a comparatively low accuracy of 58.3% when implemented on a dataset Hollywood2.

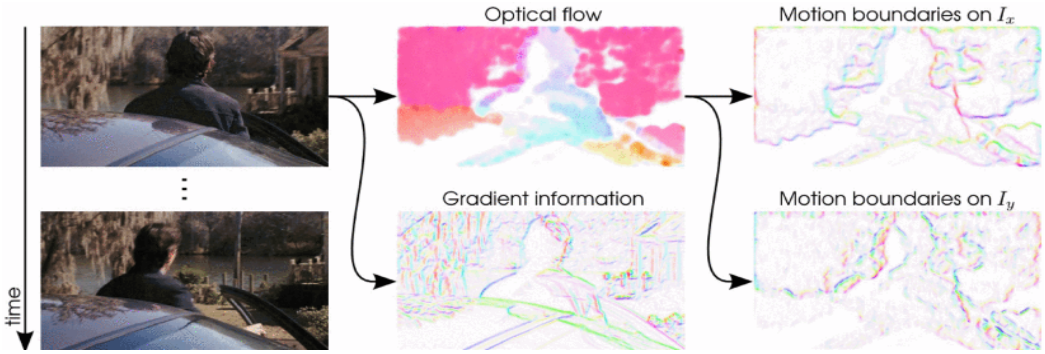


Figure 5.5: Feature Descriptor used in Method 5

5.1.6 Method 6

The method that is proposed in [6] firstly detects the region of interest that consists the upper body of the actor. Then the optical flow between interest regions of successive frame is calculated. After this calculation is done, the pyramid of accumulated histogram of optical flow descriptor is computed. Then for classification purpose, the concept of bag-SVM is utilized.

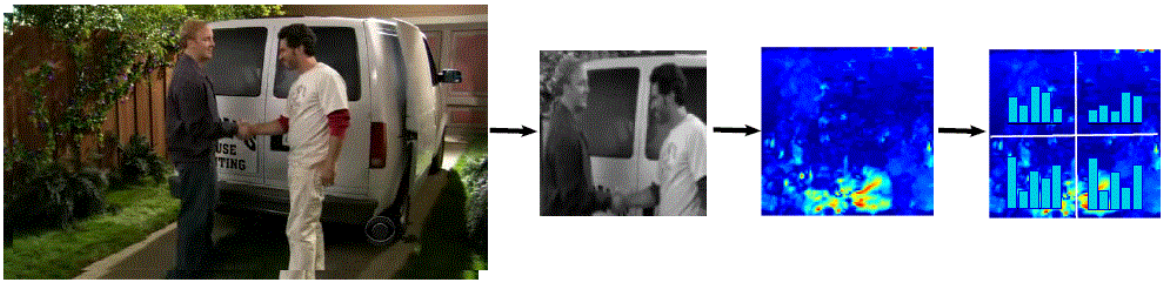


Figure 5.6: Method 6 Sketch Analysis

This approach is not completely validated but it provided a low accuracy of 46.3% during its testing phase on TV Human Interaction dataset.

5.1.7 Method 7

In the framework described in [12], classification of human interaction is carried out using body pose features. For this, joint features of skeleton are extracted using the skeleton features extraction method and the concept of MIL is also used.

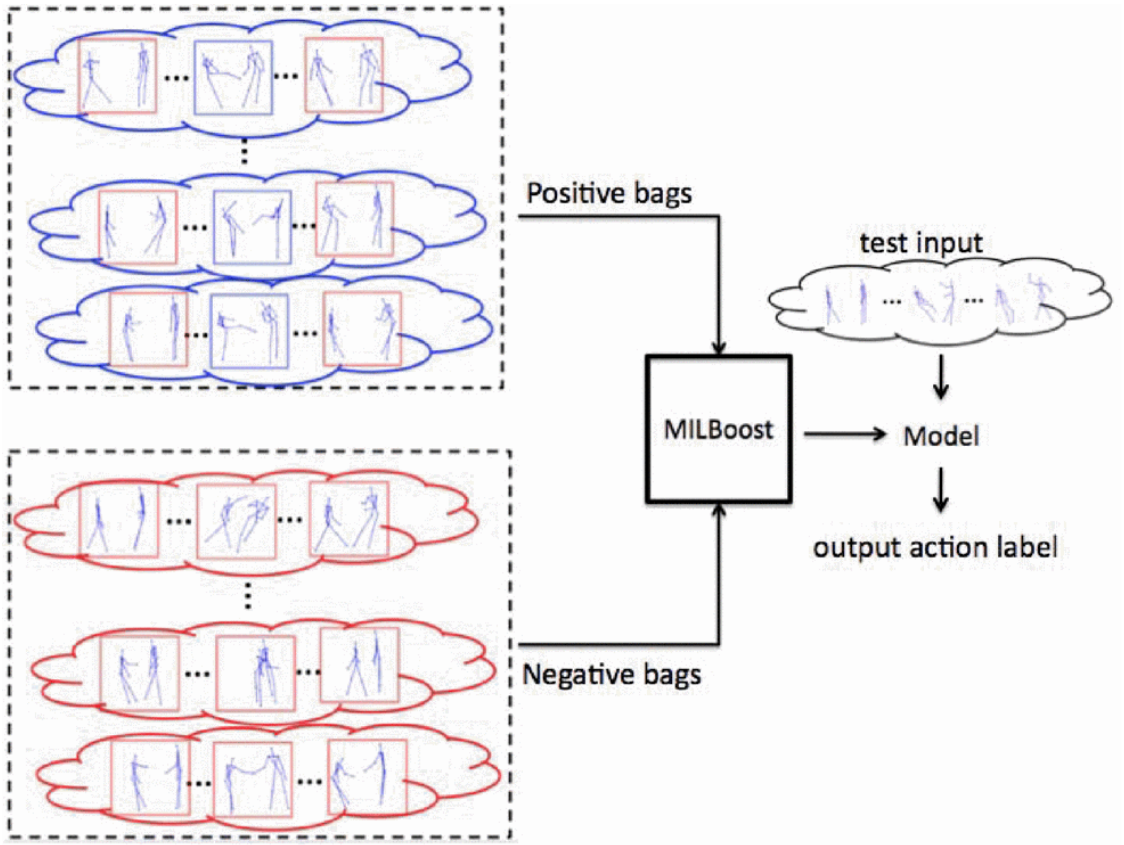


Figure 5.7: Method 7 Sketch Analysis

The framework is tested for the videos from specific viewpoint and gave the accuracy of 87.3% when applied on RGBD videos captured using Microsoft Kinect Sensors.

5.2 Survey on Real-Time Human Action Recognition

In this section, we will be discussing about the most recent technology that had provided successful results in the more precise domain of real-time human action recognition. To make the model behave like a real-time model, these methodologies will process a very small number of continuous image frames in order to classify a particular action performed by humans. The number of frames considered are so less that it becomes impossible for a human to detect the delay.

5.2.1 Method 8

The methodology described in [11] uses a deep neural network for its benefit. During the training phase, the foreground objects are extracted from the labelled dataset and the outlines are calculated which is then combined together to build a set of overlay images. Based on these set of overlay images, an autoencoder along with a pattern recognition neural network are trained sequentially. Now, these two components are combined to construct a deep neural network which is then utilized for classification.

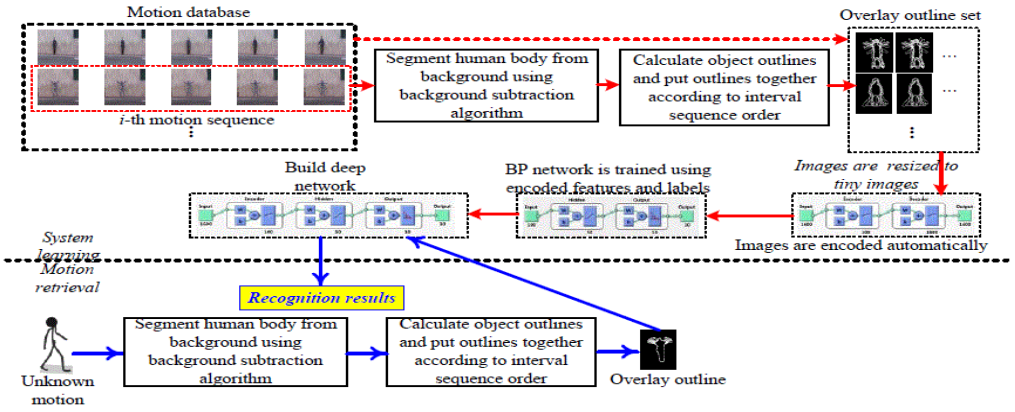


Figure 5.8: Method 8 Sketch Analysis

This process provided an average accuracy of 96% and has a small requirement for training time and network size.

5.2.2 Method 9

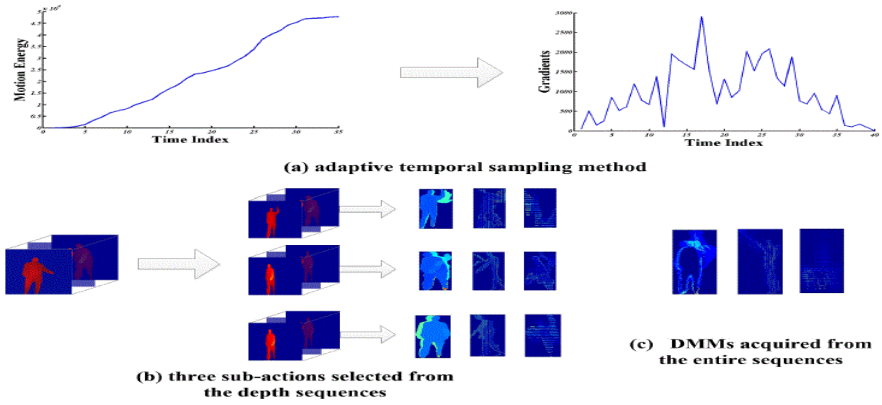


Figure 5.9: Stages Utilized in Method 9

The model structure stated in [5] consisted of several phases. The first phase is Adaptive Temporal Sampling in which the three largest gradients are selected whose corresponding frame indices are used to partition the depth sequence. The second phase is Depth Motion Mapping which is calculated by projecting depth map sequences onto three orthogonal cartesian plane and the absolute difference of the sequential projected sequences are computed. The third phase is Dimensionality Reduction by using the concept of Principal Component Analysis. The last phase is Classification where the l2-CRC classifier is trained and utilized for the purpose. This model proved to be efficient as it provided an accuracy of over 97% when tested over MSR-Action3D dataset.

5.2.3 Method 10

The approach proposed in [9] is fast, view-invarient, and good for early action recognition that also consists of several stages. First the skeleton extraction stage is encountered by the target object. Then the output of this stage is preprocessed by Savitzky-Golay filter. Now the skeletal image is broken into five parts and termed as B_1, B_2, B_3, B_4, and B_5 for right arm, left arm, right leg, left leg, and spine respectively. Then the Body Directional Velocity (BDV) feature is extracted for each body parts and are concatenated. Now an Hidden Markov Model (HMM) with the state-output distribution of Gaussian Mixture Model (GMM) is used for classification purpose.

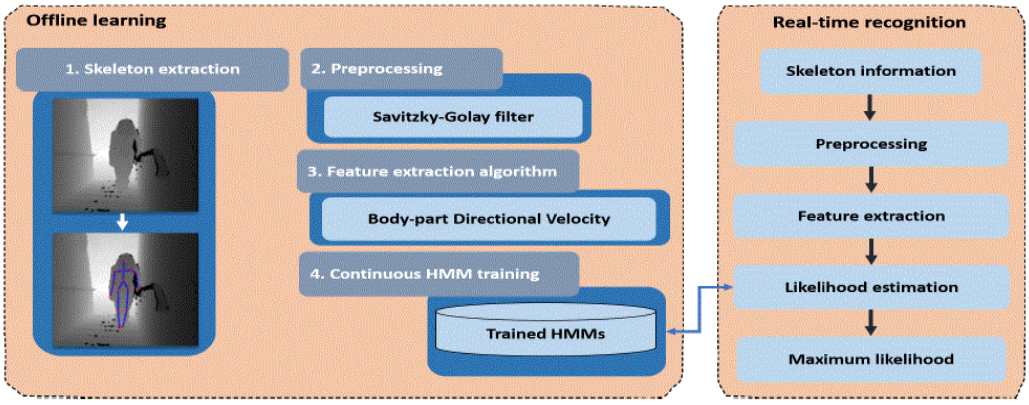


Figure 5.10: Method 10 Sketch Analysis

This approach provided an overall accuracy of 92.90%, 90.32%, and 91.10% when implemented on MSRAction3D, Florence3D, and UTKinect datasets respectively.

Table 5.1: Comparative Study

Method Sl. No.	Methodology	Experimented Dataset	Accuracy Obtained
Method 1[1]	Subsequent action classification by MIL (Machine Instance Learning), features included trajectory, HOG, HOF, and MBH, bag of features encoding technique is also applied, and SVM classifier is utilized.	Hollywood2	Full Sequence - 57.42%, and Combined Sequence - 59.8%
Method 2[8]	Three layer CNN (Convolutional Neural Network) followed by ISA (Independent Subspace Analysis) and PCA (Principal Component Analysis) for classification.	UT-Interaction	Segmented Video Sequence - 90% - 93%, and Unsegmented Video Sequence - 85% - 90%
Method 3[4]	Utilization of two different models - attribute model and interaction model followed by a layer of DDP (Data Driven Phrase) technique utilized for training data.	BIT-Interaction, UT-Interaction, and Collective Activity	90.63%, 91.67%, and 82.54% with respective datasets
Method 4[13]	Optical flow is used to compute self-similarity matrix along with the help of Fischer encoding technique which is provided to a SVM classifier for recognition.	TV Human Interaction Dataset	50.1%
Method 5[10]	Utilization of dense trajectories, computed with the help of features such as optical flow, HOG, HOF, and MBH, for generation of codebook by bag of features concept followed by a SVM classifier.	Hollywood2	58.3%
Method 6[6]	Pyramid of histogram of optical flow is calculated for the interest region followed by bag SVM for classification.	TV Human Interaction Dataset	46.3% (Comparatively Low)

Method 7[12]	Use of MIL along with body pose feature estimation which is computed from joint features of skeleton extracted using skeleton feature extraction methods.	RGBD Videos captured by Microsoft Kinect Sensor	87.3%
Method 8[11]	Extraction of foreground objects followed by outlines calculation that are combined to obtain a set of overlay images which is utilized to train two different components that are autoencoder and PRNN (Pattern Recognition Neural Network) and a deep neural network is constructed for classification.	Weizmann Motion Dataset	96% (on an average)
Method 9[5]	Four sequential phase technique is utilized which includes Adaptive Temporal Sampling, Depth Motion Mapping, Dimensionality Reduction and Classification which included l2-CRC classifier.	MSRAction3D	97%
Method 10[9]	Sequential stage model is constructed which includes skeleton extraction and preprocessing with the help of Savitzky-Golay filter followed by a segmentation process segmenting the human body into five different parts and the BDV (Body Directional Velocity) feature is calculated that helps the HMM (Hidden Markov Model) for classification.	MSRAction3D, Florence3D, and UTKinect	92.90%, 90.32%, and 91.10% with respective datasets

Chapter 6

Overview of the Approaches Applied

Throughout this part of the report, we will discuss different approaches tested by us during the course of this project which includes a traditional approach which is an extension of [7] in which one of our team-mate was a part. After going through the performance analysis of these approaches, we have proposed a novel CNN-derivative architecture that is capable to recognize human action in complete real-time by reducing the temporal latency to its optimized level. We will also describe the major problem that has prevented us from implementing the proposed system and the steps we have taken to overcome.

6.1 Traditional Approach

In this section, we first describe the dataset used in the work and different related challenges. We, then, briefly present the features that are utilized. Then, we will move on to the classification part. This section also contains a detailed figure demonstrating the flow of our proposed model.

6.1.1 Dataset

One of the major challenge during this work was the unavailability of standard dataset. We had to make the dataset, to be worked with, on our own. We had collected some data from different soccer matches. We had also standardized the data with the frame size of 150 x 150 and 25 frames per video data. We had also manually labeled the collected data so that the labeling can be used as ground truth during the entire paper. As the data was collected and labeled manually, so biasing is inherently present.

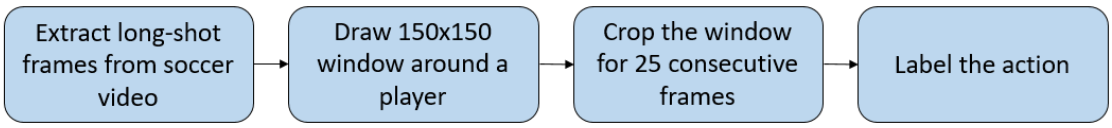


Figure 6.1: Pipeline for Collecting Dataset.

Our dataset consisted of 4 class labels with a total of 446 video data which were almost equally distributed in different classes. The class labels included in our dataset were Dribbling, Kicking, Running, and Walking. All the collected data consisted of one complete action that is labeled. As the dataset was collected from different soccer video, it was impossible to overcome the concept of camera motion. But all the data had been collected from long-shot parts of the entire soccer sports, thus decreasing the effect of camera motion to some extent.

Figure 6.2: Sample Frames from the Dataset.

6.1.2 Feature Extraction

Our method is based on extracting motion features from video sequences using optical flow. The distinct advantage of such approach is that the burden of correctly estimating motion in variable lighting conditions and clutter is entirely confined to optical flow calculation.This algorithm does not make any assumptions about the source of optical flow data, therefore, it could be applied in variety of ways. The implicit assumption is that the sequences have same frame rate and flow field dimensions. Additionally, the algorithm assumes that each sequence contains a single temporal reference, which can be used for temporal alignment, and that there exists predefined partitioning of the image into sub-regions. In a real world implementation, the descriptors can be extracted from the flow sequences immediately after the flow is obtained, therefore reducing the need for storage of original video sequences or optical flow field sequences. There is another method, which is dictionary-based representation of motion can be extremely compact, and is therefore ideally suited for embedded devices. Here, in this work, one of the feature used is histogram of such optical flow with 18 bins which gave us 18 different levels of features.

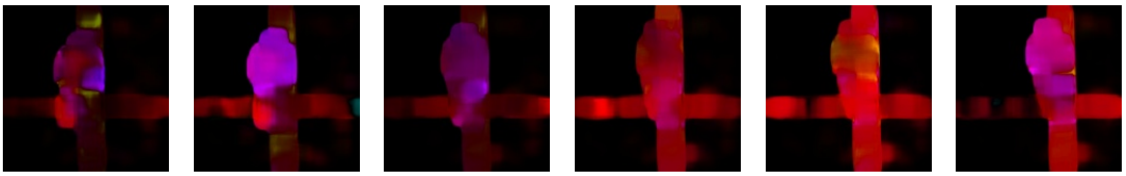


Figure 6.3: Change in Optical Flow during an action.

The next feature extractor which we had used is Local Binary pattern. It is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings. The original LBP operator forms labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these $2^{(8/2+1)} = 32$ different labels can then be used as a texture descriptor.

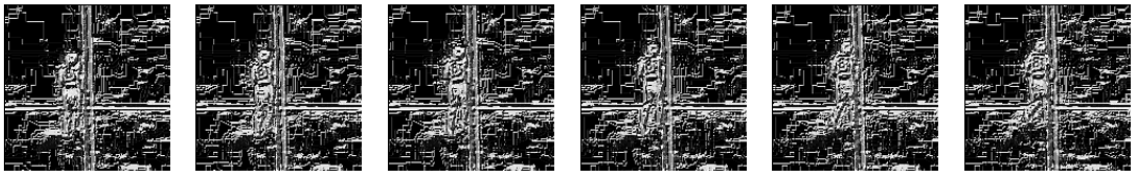


Figure 6.4: Change in Local Binary Pattern during an action.

We have used the above two mentioned features, which are stacked frame-wise and concatenated with each other, forming the feature vector for our classifier. This process gave us a feature vector of 1232 dimension when applied on the standard video clips containing 25 frames in the self created dataset. The first feature, histogram of optical flow, gave us the motion estimation of the player. Whereas, the second feature, Local Binary Pattern, gave us the textural changes within different frames of the same video clip. These two features together provided an approximately good estimation of the action performed by the player in the given video clip.

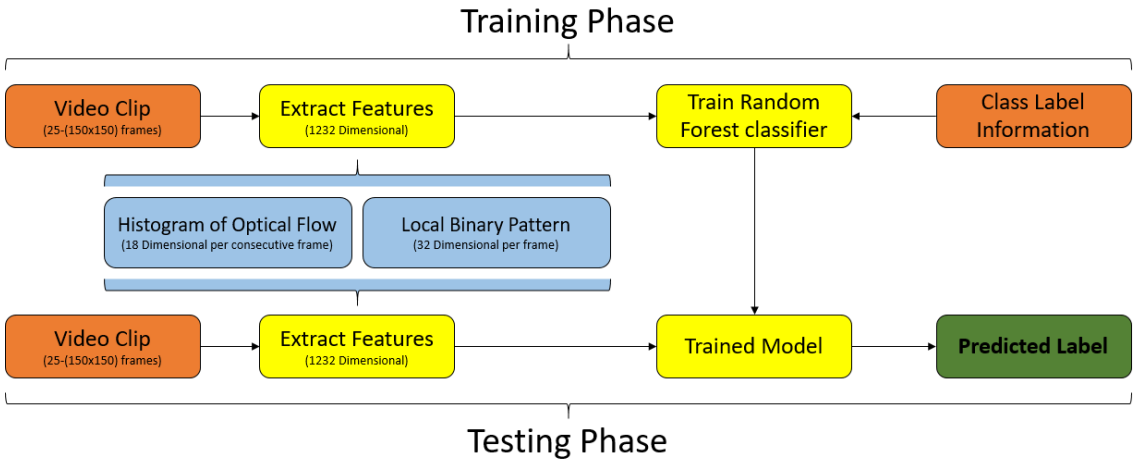


Figure 6.5: Traditional Model Pipeline.

6.1.3 Classification

We have preferred to use Random Forest as a classifier. Random forest is an ensemble method that combines several individual classification trees in the following way: From the original sample several bootstrap samples are drawn, and an unpruned classification tree is fit to each bootstrap sample. The variable selection for each split in the classification tree is conducted only from a small random subset of predictor variables, so that the ”small n large p” problem is avoided. From the com-

plete forest the status of the response variable is predicted as an average or majority vote of the predictions of all trees. Random forests can highly increase the prediction accuracy as compared to individual classification trees, because the ensemble adjusts for the instability of the individual trees induced by small changes in the learning sample, that impairs the prediction accuracy in test samples. However, the interoperability of a random forest is not as straightforward as that of an individual classification tree, where the influence of a predictor variable directly corresponds to its position in the tree.

The proposed system takes videos which can be called image sequences as inputs and outputs the labels of the actions in these image sequences.Local motion and temporal information in the image sequences are extracted by adopting above approaches. Then, this local information is combined into a novel global representation with statistical methods. Finally, the compact representations which is concatenation of frame-wise stacked feature vector is fed into the Random Forest to obtain the action types in the given video sequence. We need classifiers to recognize the human actions by the given compact representations. We prefer to employ the Random Forest classifiers for this task. The Random Forest can handle thousands of input variables and large dataset efficiently. Moreover, the Random Forest exhibits excellent performance and outperforms many other machine learning algorithms. Each decision tree in this forest behave like weak classifiers and when comes together, it forms a strong classifier. During training stages, nodes in the trees are split by randomized selection of features. This selection decreases the error rate in forest by decreasing the correlation among trees in the forest. Finally, each random tree in the forest grows and predicts the input test datas class label. The importance of variables are estimated at the end of training stage.

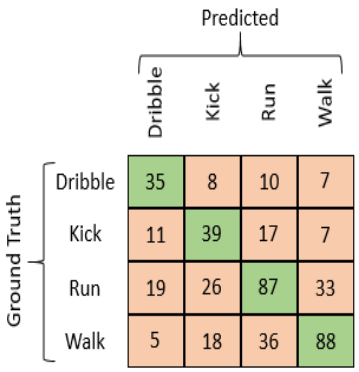


Figure 6.6: Confusion Matrix during Classification.

6.1.4 Observation

We have separated our dataset in such a way, so that we are able to perform 10-fold cross-validation over it. And in order to illustrate the importance of different features used in our model, we have tried to validate using different combination of our model which are illustrated in the table below. The results shows that the frame-wise stacked globally applied HOF and LBP feature along with random forest as classifier gives the best result of 70.4545 percentage.

Feature Used	**Classifier Used**	**Accuracy**
Locally Applied HOF	SVM	40.0000 %
Locally Applied HOF	Random Forest	34.9359 %
Globally Applied HOF	SVM	39.6861 %
Globally Applied HOF	Random Forest	46.6165 %
Locally Applied LBP	SVM	42.2222 %
Locally Applied LBP	Random Forest	46.3687 %
Globally Applied LBP	SVM	44.3820 %
Globally Applied LBP	Random Forest	55.6818 %
Locally Applied HOF + LBP	SVM	37.0787 %
Locally Applied HOF + LBP	Random Forest	50.2262 %
Globally Applied HOF + LBP	SVM	53.3333 %
Globally Applied HOF + LBP	Random Forest	**70.4545** %

Table 6.1: Table illustrating accuracies using different approaches.

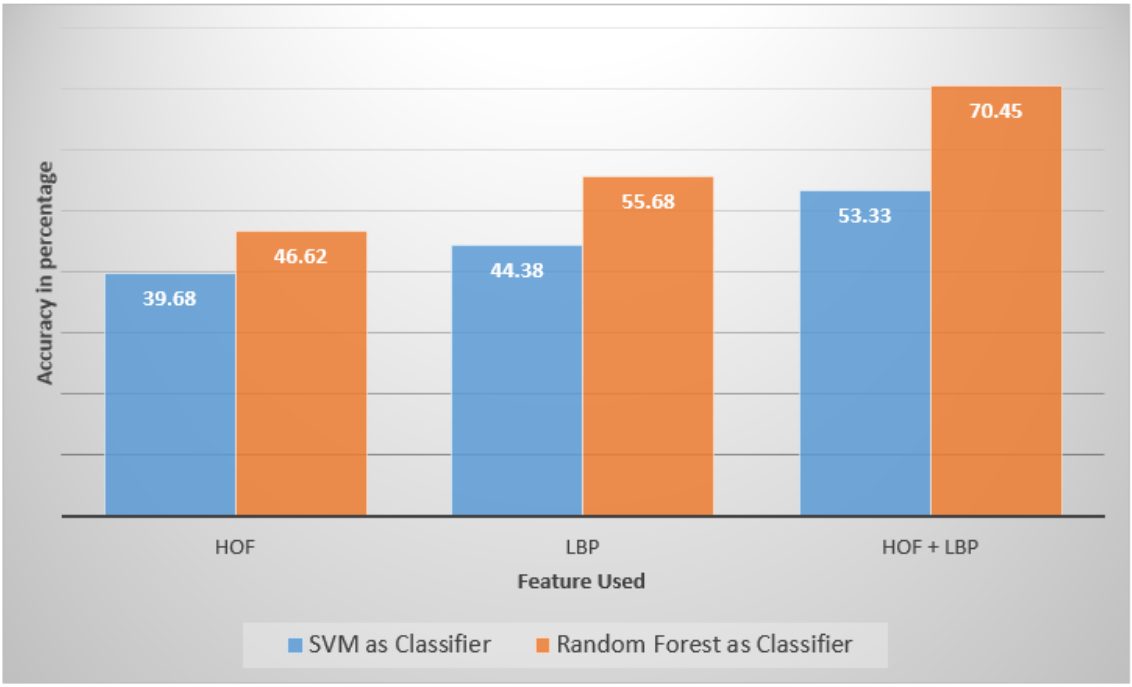


Figure 6.7: Chart illustrating accuracies using different approaches.

6.1.5 Simulation

Another most challenging part of our work is to demonstrate the working model in a long-shot soccer video as soccer video may consists of different background noise with abrupt camera motion in different direction and presence of group actions in a single frame. One more thing that makes long-shot soccer video as challenging to apply our model is that we have to learn the low level features in the training set whereas in the evaluation of the soccer video includes learning different high level feature that are performed with random noise in it. Another thing is that, the dataset is collected by three persons which includes personal bias in it which means one may classify as walking other may classify as running for the same sample. It is one of the reason why our model is biased toward dribbling scenes. In addition, the action statistics have different characteristics in different group actions. Therefore, these statistics provides discriminative features for global motion classification and our motion descriptor is fine modeled for both group action recognition and single action recognition.

Figure 6.8: Sample Frames from Soccer Video after Applying the Proposed Model.

In order to apply our model in long-shot soccer video, we have first manually marked the player in each frame. Then we have created a bounding box of size 150x150 pixels surrounding each player in the frame and moved it with the player till 25 frames and extracted the feature. Then the extracted feature is fed to the classifier to predict the label. The same procedure is followed for each player present in each frame. Thus generating the video consisting of the action label of each and every player.

6.2 Proposed Architecture

As observed from the present state-of-the-art research approaches in this field of Human Action Recognition, it is seen that all the approaches considered a minimum predefined number of image frames, that is to be analysed for recognizing an action. These approaches of considering more than one frame at a time has resulted to a short latency which is directly proportional to the number of frames considered to recognize an action.

Mathematically, the latency period of the system is given by the equation

$$L = t_i + n(f) * t_f + C \tag{6.1}$$

where L denotes Latency time, t_i denotes the initial time required to capture the predefined number of frames, $n(f)$ denotes the number of frames considered, t_f denotes the time taken to process single frame, and C denotes the time taken for constant time tasks that needs to be performed. The main goal of the proposed architecture is to minimize the latency time which can be achieved by considering only a single frame at a time that will reduce the equation of latency to $L = t_f + C$.

In order to achieve the desired goal, we can use a new convolutional neural network model, as shown in fig. 6.9, which is a derivative of the traditional convolutional neural network model and overcomes the disadvantage which prevents it from being used as an action recognition model. In this model, the immediate previous predicted class label is concatenated with the flattened feature vector of its immediate successive feature component obtained after the feature extraction section of the convolutional neural network which is then fed to the classification section. This feedback system of the output to the intermediate section of the model makes it recursive in nature. And as the prediction of the class label at a particular instance is dependent upon the previously classified class label over time, so the model is considered to have memory for storing the previous outputs. This makes the suggested model to be termed as "Memory-Based Recursive Convolutional Neural Network". This model satisfies all the pre-requisites to be used as an action recognition model and as the recognition is only dependent on the instance image frame and the previously classified class label which gets stored in the model, the latency of the model becomes almost negligible which makes the model to behave as a real-time action recognition model.

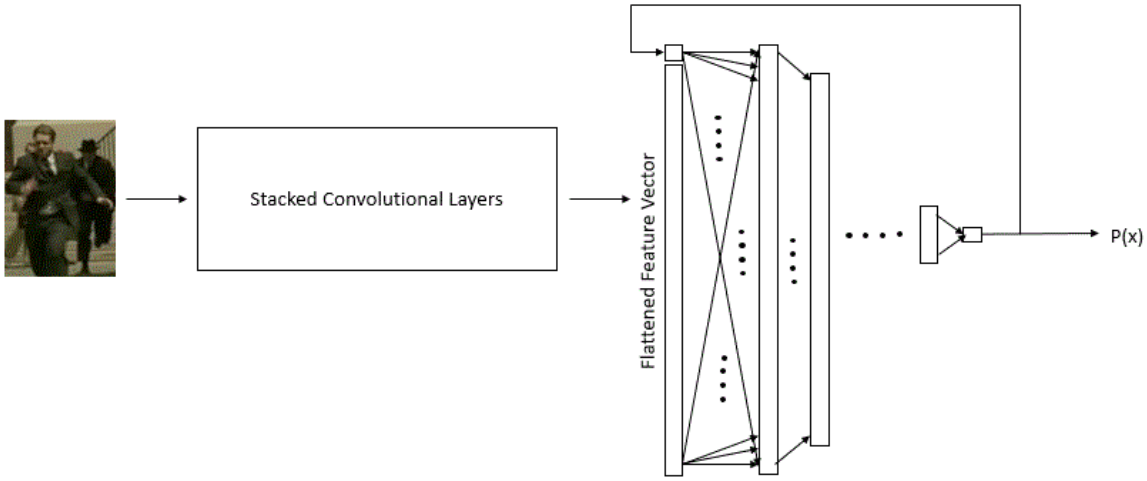


Figure 6.9: System Framework for the Proposed Model

6.3 Major Problem

Any changes made to the traditional CNN model needs a mathematical derivation for its implementation. Additional properties would give rise to more number of weights that need to be optimized during training. Similarly, the proposed model has led to this situation where a mathematical technique need to be developed in order to deal with the updation of the additional weights during training phase. Till now, this mathematical technique has not been figured out.

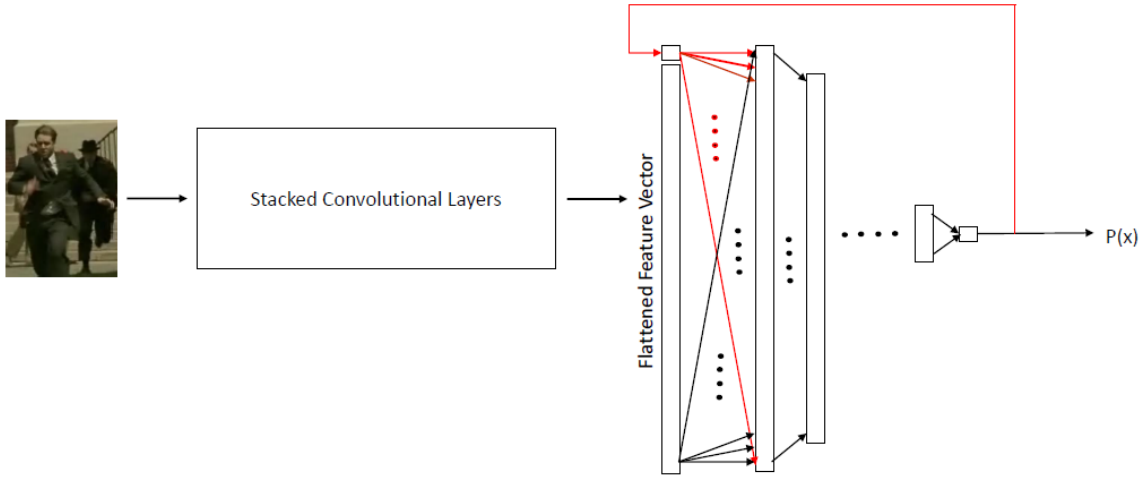


Figure 6.10: Weight Correction Problem of the Proposed Model

In order to overcome this problem, many approaches had been tried. One of those include use of Transfer Learning with autoencoder as classifier utilizing the algorithm:

1. Train a CNN with a predefined data set.

2. Use the concept of Transfer Learning to trim the classification module of the CNN.
3. Now the flattened feature layer becomes the last layer of the CNN.
4. Connect this layer with some auto-encoders like LSTM.
5. Train the LSTM part only by feeding the same data set to the CNN.
6. Now test the model using some test data and record the accuracy.

But unlike other tried approach, this algorithm also failed to give the desired result. This showed that there is no other alternative but to come up with mathematical solution that would update the additional weights of the proposed model which has not been derived yet. We have consulted with some external researchers regarding the problem so that we can come up with the mathematical model as soon as possible.

Chapter 7

Conclusion

In this work, we have briefly discussed the concept of Human Action Recognition along with the basic background required to start with. We have also gone through the objective of this project in order to add something new in the recent technology and also the motivation behind the concept of this project. Next we have also discussed the concepts of some of the latest technology used for the purpose of Human Action Recognition, some of which also included the concept of real-time. Then we have stated the detailed description of the traditional method implemented by us in the given field and seen the comparative study of the outcome. Then we have proposed a new architectural model and its need in present technology along with the problems it inherits which when solved can be utilized to recognize human action in real time with the help of frame level features.

References

[1] B. Antic, T. Milbich, and B. Ommer. Less is more: Video trimming for action recognition. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 515–521, Dec 2013.

[2] C. J. Dhamsania and T. V. Ratanpara. A survey on human action recognition from videos. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, pages 1–5, Nov 2016.

[3] Geetanjali Vinayak Kale and Varsha Hemant Patil. A study of vision based human motion recognition and analysis. *CoRR*, abs/1608.06761, 2016.

[4] Y. Kong, Y. Jia, and Y. Fu. Interactive phrases: Semantic descriptionsfor human interaction recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(9):1775–1788, Sep. 2014.

[5] Yang Li, Qin Lu, and Wusheng Luo. Adaptive temporal sampling for real-time human action recognition. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 1306–1310, Oct 2016.

[6] Manuel J. Marín-Jiménez and Nicolás Pérez de la Blanca. Human interaction recognition by motion decoupling. In João M. Sanches, Luisa Micó, and Jaime S. Cardoso, editors, *Pattern Recognition and Image Analysis*, pages 374–381, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[7] D. P. Mukherjee, S. Sarkar, A. Saha, S. Barnwal, and S. S. Mahapatra. Player action recognition on broadcast soccer videos. *Fifth Summer School on Computer Vision, Graphics and Image Processing, Indian Statistical Institute, Kolkata*, May-July 2018.

[8] N. Nguyen and A. Yoshitaka. Human interaction recognition using independent subspace analysis algorithm. In *2014 IEEE International Symposium on Multimedia*, pages 40–46, Dec 2014.

[9] S. A. W. Talha, M. Hammouche, E. Ghorbel, A. Fleury, and S. Ambellouis. Features and classification schemes for view-invariant and real-time human action recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 10(4):894–902, Dec 2018.

[10] H. Wang, A. Klser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *CVPR 2011*, pages 3169–3176, June 2011.

[11] Q. Xiao and Y. Si. Human action recognition using autoencoder. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1672–1675, Dec 2017.

[12] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–35, June 2012.

[13] Bo Zhang, Yan Yan, Nicola Conci, and Nicu Sebe. You talkin' to me?: Recognizing complex human interactions in unconstrained videos. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 821–824, New York, NY, USA, 2014. ACM.

Annexure I: Publication Info

1. Title : "A Survey on Current Trends in Human Action Recognition"
2. Journal : "Image and Vision Computing"
3. Manuscript Number : IMAVIS-D-19-00169
4. Review Comment : "The paper summarizes a set of methodologies for action recognition. However, the paper does not provide a critical discussion on the surveyed methods. Moreover, the surveyed methods are not compared empirically. Hence, the contribution of the paper is judged to be too low to be considered for publication in a top computer vision journal like Image and Vision Computing Journal (IVCJ) and the manuscript cannot be recommended to be considered for publication."