High Performance Video Stream Analytics System for Object Detection and Classification

Muhammad Usman Yaseen

College of Engineering and Technology
University of Derby
UK

This dissertation is submitted in fulfillment of the requirements
for the degree of
Doctor of Philosophy

September 2018

Abstract

Due to the recent advances in cameras, cell phones and camcorders, particularly the resolution at which they can record an image/video, large amounts of data are generated daily. This video data is often so large that manually inspecting it for object detection and classification can be time consuming and error prone, thereby it requires automated analysis to extract useful information and meta-data. The automated analysis from video streams also comes with numerous challenges such as blur content and variation in illumination conditions and poses. We investigate an automated video analytics system in this thesis which takes into account the characteristics from both shallow and deep learning domains. We propose fusion of features from spatial frequency domain to perform highly accurate blur and illumination invariant object classification using deep learning networks. We also propose the tuning of hyper-parameters associated with the deep learning network through a mathematical model. The mathematical model used to support hyper-parameter tuning improved the performance of the proposed system during training. The outcomes of various hyper-parameters on system's performance are compared. The parameters that contribute towards the most optimal performance are selected for the video object classification. The proposed video analytics system has been demonstrated to process a large number of video streams and the underlying infrastructure is able to scale based on the number and size of the video stream(s) being processed. The extensive experimentations on publicly available image and video datasets reveal that the proposed system is significantly more accurate and scalable and can be used as a general purpose video analytics system.

Acknowledgements

At first, all praise is due to ALLAH(s.w.t) for granting me the ability and determination to complete my PhD. I am extremely grateful to my supervisor, Professor Ashiq Anjum for offering me the PhD studentship at University of Derby and also for supporting and encouraging me throughout the PhD journey. I would like to avail this opportunity to appreciate his kind advices and his suggestions that helped me to tackle the challenges successfully. I feel so fortunate to have a great mentor and advisor.

I am profoundly obliged to Professor Omer Rana and Mohsen Farid for their continuous support and precious time to discuss the ideas that are now a part of this thesis. I would also like to thank them for enhancing my research and writing skills that ended up publishing the parts of this research work to well-known international conferences and well reputed journals. My gratitude also extends to The Head of School of Computing and Mathematical Sciences, Professor Lu Liu for his support through some of the issues during PhD.

I would like to express my gratitude to all of my colleagues with whom I had the opportunity of working during this period. I would also like to acknowledge the research community and open-source community for their ideas that aided the work.

Finally, and most importantly, I would like to thank my parents and my wife for their continuous and unconditional love, support and prayers. I would also like to thank my dear brothers and sisters, who have always motivated and comforted me with their sweet gossip and laughter.

List of Publications

J = Journal Paper; **C** = Conference Proceeding; **P** = Poster Presentation

J1: **Muhammad Usman Yaseen**, Ashiq Anjum, Omer Rana, Nick Antonopoulos "*Illumination and expression invariant face recognition using convolutional neural network*", in IEEE Transactions on Systems, Man and Cybernatics, 2018 **(Under Review)**

J2: **Muhammad Usman Yaseen**, Ashiq Anjum, Omer Rana, Nick Antonopoulos "*Deep Learning Hyper-parameter Optimization for Video Analytics in Clouds*", in IEEE Transactions on Systems, Man and Cybernatics, vol. 99, pp. 1-12, 2018. DOI: 10.1109/TSMC.2018.2840341 **(IF:5.131)**

J3: **Muhammad Usman Yaseen**, Ashiq Anjum, Nick Antonopoulos "*Cloud-based Video Analytics using Convolutional Neural Networks*", in Software: Practice and Experience, pp. 1-19, 2018. DOI: 10.1002/spe.2636 **(IF:1.338)**

J4: **Muhammad Usman Yaseen**, Ashiq Anjum, Omer Rana, Richard Hill "*Cloud-based scalable object detection and classification in video streams*", in Future Generation Computer Systems, vol. 80, pp. 286-298 , 2017. DOI: 10.1016/j.future.2017.02.003 **(IF: 4.639)**

C1: Muhammad Ali, Ashiq Anjum, **Muhammad Usman Yaseen**, Ali Reza Zamani, Balouek Thomert, Omer Rana, Manish Parashar, *"Edge Enhanced Deep Learning System for Large-scale Video Stream Analytics"*, in 2nd IEEE International Conference on Fog and Edge Computing, May 1-3 , 2018, Washington DC USA

C2: **Muhammad Usman Yaseen**, Ashiq Anjum, Nick Antonopoulos, *"Modeling and Analysis of a Deep Learning Pipeline for Cloud based Video Analytics"*, in 4th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, December 5-8 , 2017, Texas, USA

C3: **Muhammad Usman Yaseen**, Ashiq Anjum, Nick Antonopoulos, *"Spatial Frequency based Video Stream Analysis for Object Classification and Recognition in Clouds"*, in 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, December 6-9 , 2016, Shanghai, China

C4: **Muhammad Usman Yaseen**, Muhammad Sarim Zafar, Ashiq Anjum, Richard, Hill, *"High performance video processing in cloud data centres"*, in 10th IEEE International Symposium on Service-Oriented System Engineering, 29 March-1 April, 2016, Oxford, UK

P1: **Muhammad Usman Yaseen**, Ashiq Anjum, *"Video Object Classification from Video Streams in Cloud Data Centres "*, in Big Data In Transit, February 16th, 2016, Derby, UK

P2: **Muhammad Usman Yaseen**, Ashiq Anjum, *"Automated Detection and Classification of Objects from Video Streams in a Cloud Data Centre"*, in 21st British Machine Vision Summer School, July 4-8, 2015, Swansea, UK

List of Abbreviations

2DEMD	Two-dimensional Empirical Mode Decomposition
Acc	Accuracy
Avg	Average
Amp	Amplitude
ANN	Artificial Neural Network
BEMD	Bi-dimensional Empirical Mode Decomposition
CNN	Convolutional Neural Network
CONV	Convolution
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CWD	CUDA Work Distribution
D-LDA	Direct Linear Discriminant Analysis
FPR	False Positive Rate
FIFO	First In First Out
FPS	Frame Per Second
FS	File System
F-LDA	Fractional Linear Discriminant Analysis
FP	False Positive
FN	False Negative
FT	Fourier Transform
GUI	Graphical User Interface
GD	Gradient Descent

GB	Gigabyte
GHz	Gigahertz
GPU	Graphical Processing Unit
GSF	Gradient Surface Features
HoG	Histogram of Oriented Gradient
HDFS	Hadoop Distributed File System
HIB	Hipi Image Bundle
HGPP	Histogram of Gabor Phase Pattern
HMM	Hidden Markov Model
IMFs	Intrinsic Mode Functions
ICA	Independent Component Analysis
KNN	K-Nearest Neighbours
LBP	Local Binary Pattern
LTP	Local Ternary Pattern
LFW	Labelled Faces in the Wild
LR	Learning Rate
LF	Loss Function
LDA	Linear Discriminant Analysis
LBPH	Local Binary Pattern Histogram
LPQ	Local Phase Quantization
LQP	Local Quantized Patterns
LRC	Local Regression Classifier
MR	MapReduce
MLBP	Multi-scale Local Binary Pattern
MLPQ	Multi-scale Local Phase Quantization
MLTP	Multi-scale Local Ternary Pattern
MDS	Multi-dimensional Scaling
MLP	Multilayer Perceptron
MBC	Monogenic Binary Coding

MBP	Monogenic Binary Pattern
NMF	Non-negative Matrix Factorization
Norm	Normalization
Ori	Orientation
Pre	Precision
PCA	Principle Component Analysis
PNG	Portable Network Graphics
PSNR	Peak Signal-to-Noise Ratio
Pha	Phase
ReLU	Rectified Linear Unit
RAM	Random Access Memory
RDD	Resilient Disk Drive
Rec	Recall
R-CNN	Recurrent Convolutional Neural Network
RBF	Radial Basis Function
RoI	Region of Interest
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
SUB	Sub-sampling
SVM	Support Vector Machine
SIMD	Single Instruction Multiple Data
SM	Streaming Microprocessor
STD	Standard Deviation
TPR	True Positive Rate
TP	True Positive
TN	True Negative
TSVM	Transductive Support Vector Machine
VOC	Visual Object Challenge
WNs	Worker Nodes

Nomenclature

$\circledast$	Convolution Operator
$\times$	Multiplication
$\otimes$	Element-wise Multiplication
$\downarrow$	Down-sample
$\uparrow$	Up-sample
Δ	Rate of Change

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

This chapter provides an introduction to the problem description, context and details the aim and objectives of this research study. This include introducing the context of video analytics systems and the challenges involved in achieving a high performance and accurate video analytics system. It further presents the major contributions of this research study. The organisation of the thesis is presented at the end of the chapter.

1.2 Research Context

The increasing availability and deployment of video cameras, sensors and other devices has resulted in the generation of large amounts of data. A number of cameras installed at various locations generate large volumes of image and video data. According to a study, there are more than 6 million video cameras in the UK alone [1]. Video camera based monitoring of events has increased from just over 300,000 in 1996 to over 2 million in 2004 [2]. This data needs to be processed to generate useful information such as detection and classification of a marked person from large amount of video data. Various other types of information can also be extracted from

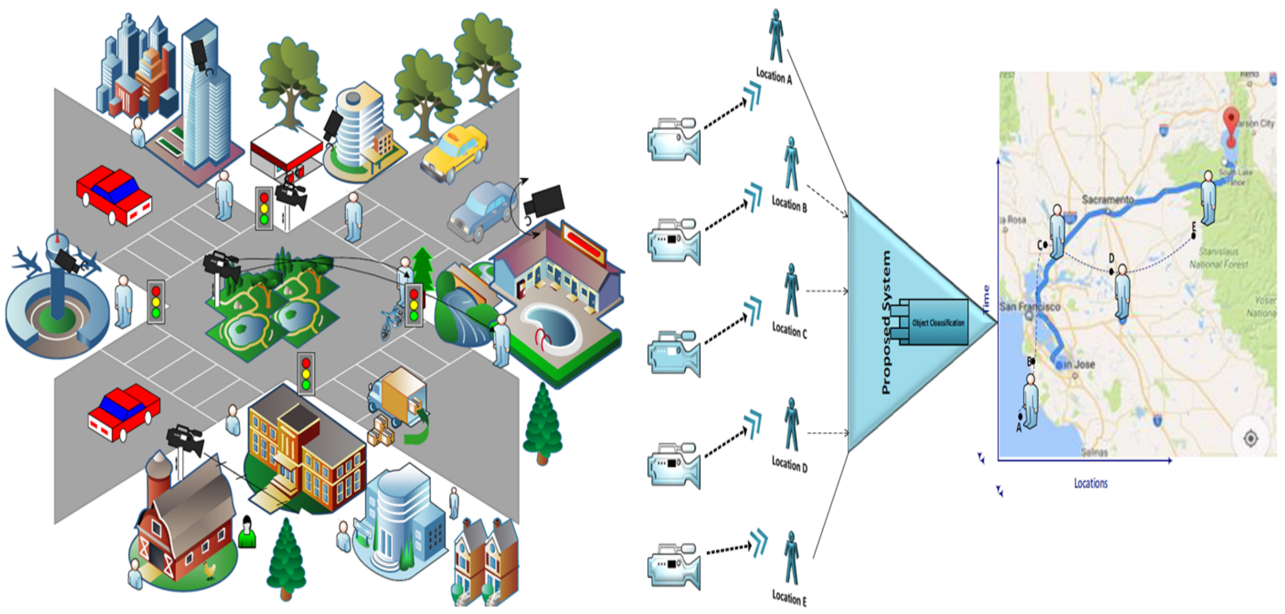


Figure 1.1: Capturing and Mapping of Marked Person

these video streams, such as recognition and identification of moving objects corresponding to a specific area of interest.

The term video analytics refers to the optimized processing of these video streams by using intelligent approaches, so that clusters of information can be automatically extracted from them. Video analytics systems mainly perform object detection and classification. Often a video may contain a number of objects. These objects can reside at any location within a frame, requiring the detection process to investigate different parts of a frame to locate the object of interest. Object classification, on the other hand, refers to the identification of detected objects. A video stream and some known labels are provided to the system. It then assigns the correct labels to the detected objects in a video stream.

Video analytics plays a vital role in analysing videos to detect and track different temporal and spatial events. Figure 1.1 depicts the phenomenon of processing the video data captured from different camera sources, in order to locate a person of interest from large amount of video streams. The mapping of the person is then made with the locations he visited. The large amount of video data makes it nearly impossible for operators to manually process it and presents itself as a severe challenge for the object classification process.

1.3 Problem Description

Most of the existing deep learning based video analytics systems are manual, require human intervention and are time consuming. The recent increase in the availability of video cameras has resulted in the generation of large amount of video streams. A conventional way is to process these video streams manually by human operators. However, humans are prone to errors and they have cognitive limitations [3]. Therefore, the probability of errors and incorrect information become higher in the manual systems. Also manual processing of large amount of video streams consumes considerable amount of time in finding objects of interest [4] [5].

The video analytics systems have to process video streams that contain a number of challenges such as illumination and blur. The video cameras used by conventional video analytics systems work under uncontrolled conditions. The video streams generated by these cameras, therefore, often contain blur frames due to motion, out of focus, or atmospheric turbulence. They are also prone to various illumination effects [6]. Rotation angle of the objects being monitored is also another challenge posed by these cameras.

In case of person monitoring and identification, the most common challenges include pose and illumination variations, facial expressions and ageing conditions. The expression and illumination challenges when come together can particularly vary the appearance of face images of individual and mark their impairment. These variations are so intense that it becomes impossible even for humans to recognize them.

The deep learning based video analytics systems are difficult to tune as they involve a number of hyper-parameters. They proved to be successful on a number of object detection and classification tasks on large scale data. They also have the generalization capability and can be trained on large scale input dataset belonging to different classes. However, these systems also struggle to perform well on the challenging datasets and their accuracy severely drops down especially for the case of expression and illumination variant datasets.

The deep learning based video analytics systems involve a number of hyper-parameters, including learning rate, momentum, activation function, optimization algorithm and weight parameter initialization. The tuning of these hyper-parameters is still a very challenging task [7] [8]. A trial-and-error approach is mostly followed in selecting these parameters, which makes it time consuming and at times may provide inaccurate results.

The video analytics systems based on both shallow and deep networks are compute intensive and require higher processing time. The systems based on shallow networks extract features from small local patches of subsequent video frames and then aggregated to produce global features for appearance and motion information. This phenomenon tends to produce high dimensional feature vectors [9], requiring large number of computing resources and processing time which becomes quite expensive and time consuming for large sets of video data.

The systems based on deep networks require substantial amount of time for training and tuning the hyper-parameters of the system. Deep networks are compute intensive in nature [10] and perform slowly and in-efficiently on single machine. Especially, if they are operating on large datasets, these systems need to be scaled and configured on a cloud based distributed infrastructure for rapid processing of video streams at a reasonable computational cost.

Automatic video object classification case-studies, required for the validation of video analytics systems are scarce in existing literature. No automated case-studies exist today which could address all the existing problems at one platform. The automated object classification case-study should be capable of validating the video analytics systems based on both shallow and deep networks.

1.4 Research Aims and Objectives

This research work aims to investigate an intelligent video analytics system for automatic detection and classification of objects from large number of video streams in a cloud based distributed system. This research aim leads to the following research objectives.

- To propose a video analytics system for the automated detection and classification of objects from large number of video streams.

- To improve the accuracy of the system by investigating feature fusion approach in spatial-frequency domain under uncontrolled conditions.
- To propose a mathematical model for the hyper-parameter tuning of the deep learning to improve the performance of training and inference for video analytics.
- To scale the proposed system based on the number and size of the video streams by investigating a distributed video analytics system.
- To validate the proposed system using a video object classification case-study on the data that has been collected from real life scenarios.

1.5 Major Contributions of this Research Study

1. *An intelligent video analytics system is proposed to automate and speed-up the processing of large number of video streams in a cloud based distributed system.* The video streams are automatically fetched from the cloud storage and then decoded to extract individual frames. Each individual frame is processed independently for detection and classification. We have introduced the concept of adaptive frame sampling in which the frames which do not contain any object in them are discarded. This reduces the amount of video frames to be processed and only those frames are retained which contain objects in them. (Published in IEEE SOSE, Oxford, UK, 2015)

2. *We have demonstrated that cascaded detection and classification is an efficient way to automate the video analysis process and to minimize human intervention.* The object in each video frame is first detected to provide a reference for the location of the object which can be tracked in the subsequent frames. It is cropped and saved as a separate image, so that the recognition steps have to process a smaller sized image. The object is then passed on to the subsequent object classification phase. We have proposed an object matching algorithm in which the target object is compared with the candidate objects for classification.(Published in Future Generation Computer Systems, ELSEVIER, 2017)

3. *We have demonstrated that higher intrinsic mode functions(IMFs) are adequate to perform classification with high accuracy rate under challenging conditions.* We pioneer the use of Bi-dimensional empirical mode decomposition(BEMD) to decompose a video frame into Intrinsic Mode Functions (IMFs) in the spatial frequency domain. BEMD provides several advantages over spatial domain analysis. Features can be extracted easily according to the distribution of local phase or energy. Each IMF is analysed independently using Reisz transform to extract local properties (amplitude, orientation, and phase) of a video frame. These local properties are further examined to perform object classification. (Published in IEEE/ACM BDCAT,Shanghai, China, 2016)

4. *We have proposed a feature fusion strategy based on the orientation components of the video frames to achieve high accuracy, recall and precision for blur and illumination invariant object classification.* We studied the orientation, phase and amplitude properties derived from each intrinsic mode function and showed their performance in terms of accuracy. It has been observed that the orientation component of the object leads to a higher accuracy rate. We have shown that feature fusion strategy based on the orientation components can significantly improve the accuracy of the deep learning pipeline. (Under Review in Transactions on Systems, Man and Cybernatics, IEEE, 2018)

5. *We have proposed a deep learning pipeline for cloud-based video analytics and a mathematical model is formulated to observe the effects of hyper-parameter tuning on the system's performance.* We identified the parameters that have a major contribution in improving the performance of the system. We parallelized the training of the deep learning pipeline by dividing the dataset into small subsets and then passing over these subsets of data to separate neural network models. The models are trained in parallel and the resultant parameters for each model are then iteratively averaged and collected at the main driving node of the compute cluster. This approach helped in speeding up the network training even on larger datasets.(Published in Transactions on Systems, Man and Cybernatics, IEEE, 2018)

6. *We have shown that the cloud-based parallel distributed training is an efficient way to speed*

up the training process. Multiple nodes of the compute cluster in the cloud have been used to train partial models on each node. The parameters of the compute cluster including number of cores and executors, serializers, rate of parameter averaging and number of mini-batches are finely tuned to achieve maximum utilization of available resources. The execution time of the system is also improved by changing the block size, replication factor and varying the number of cloud nodes. This enabled the system to perform rapid computation and helped to process large amounts of data. (Published in IEEE/ACM BDCAT, Austin, Texas, USA, 2017)

7. *We have proposed a video object classification case-study to validate the proposed system with the data that has been collected from real life scenarios.* The target object which is to be classified is passed through the trained classifier. The classifier generates a set of probabilities against the target object. The matching scores greater than an empirically determined threshold reveal the classification of the target object.(Published in Software, Practice and Experience, WILEY, 2018)

1.6 Thesis Organisation

This section describes the organization of the thesis as depicted in Figure 1.2. In Chapter 1, the research problem is introduced and the issues involved in it are studied. A research methodology and its objectives are formalized.

Chapter 2 details the most recent state-of-the-art approaches used for object detection and classification with the motivation to identify potential research gaps and limitations in the existing works. It is divided into three main sections: Local features based methods, global features based methods and neural network based methods. We have also reviewed various classification algorithms and highlighted the weaknesses of current algorithms. An explanation on how we are addressing these weaknesses is then presented.

Chapter 3 provides the details of the video analytics workflow for object detection and classification. All the details of the proposed workflow are explained and the connection between

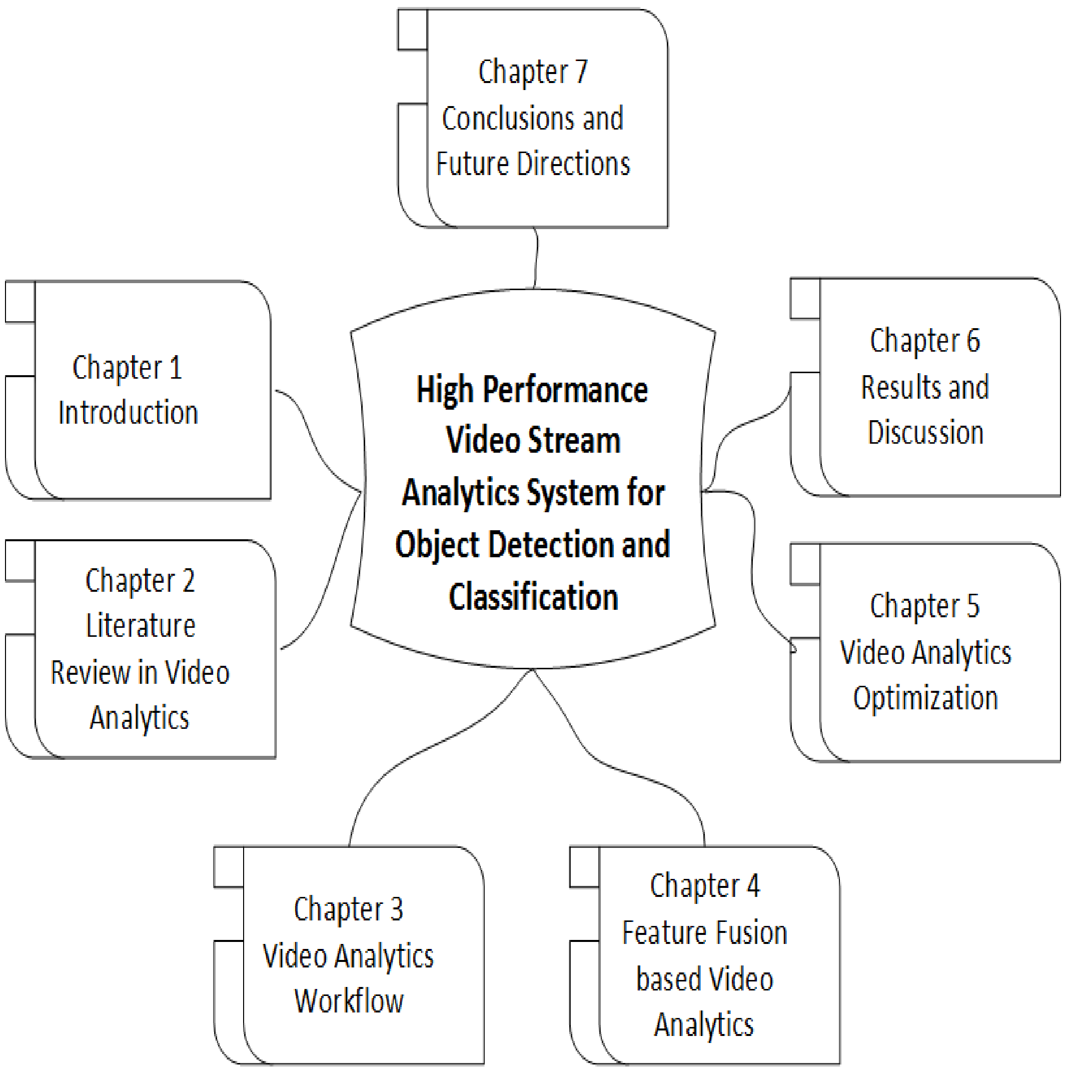

Chapter 7
Conclusions and Future Directions
Chapter 1
Introduction
High Performance Video Stream Analytics System for Object Detection and Classification
Chapter 6
Results and Discussion
Chapter 2
Literature Review in Video Analytics
Chapter 5
Video Analytics Optimization
Chapter 3
Video Analytics Workflow
Chapter 4
Feature Fusion based Video Analytics

Figure 1.2: Thesis Organization

each component is described in this chapter. It is shown that the proposed workflow helps to minimize human intervention and provides automated object classification from large number of video streams.

A novel blur and illumination invariant video analytics approach is presented in chapter 4. The proposed approach aids in classification of objects from video data containing challenges such as illumination and blurred objects. The use of bi-dimensional empirical mode decomposition (BEMD) for feature extraction is explained. The details of feature fusion strategy are also provided. We also discussed the details of the deep learning model which is used for classification from the fused features.

In chapter 5, the optimization of the deep learning pipeline for cloud based video analytics system *is presented.* The system is represented by a mathematical model and hyper-parameters of deep convolutional neural network are finely tuned on the basis of proposed mathematical model. A number of results in the results section are presented to assess the performance of the proposed model. The details on the distributed training of the system and experimental setup detailing the deployment of the proposed system on the cloud infrastructure are also provided in this chapter.

Chapter 6 presents the results of our proposed video analytics system. A number of experiments have been performed to validate the proposed system.The results are discussed in great detail both graphically and analytically. It is shown that the proposed video analytics system can perform object classification with high accuracy, precision and scale under uncontrolled environmental conditions.

Chapter 7 concludes the thesis with a summary of the contributions of this research study. Conclusions and remarks are made to highlight the research accomplishments, applicability and limitations of this work. The perspectives on future research directions are also part of this last chapter.

Chapter 2

Background and Literature Review

2.1 Introduction

Chapter 1 described the aims and objectives of an intelligent video analytics system for automatic detection and classification of objects from large number of video streams. Chapter 2 will survey the state of the art approaches related to the research undertaken in this work. It will also identify the gaps and critically describe how the work undertaken links to the existing literature.

A video analytics system consists of two steps. 1) Feature Extraction 2) Classification. Features are extracted from an object. A feature is a piece of information in the form of numerical values extracted from the images or video frames. These extracted features are much lower in dimension than the original image or video frame. The existing feature extraction approaches can be divided into two categories i.e. local features based approaches and global features based approaches. These features are fed to a classifier in order to perform classification. Classification is the process of identifying the class of an unknown object. Both local and global features have been used in the past to perform object classification.

This chapter provides an overview of the most commonly used feature extraction approaches and classification algorithms. It also describes the frameworks that have been used in the past

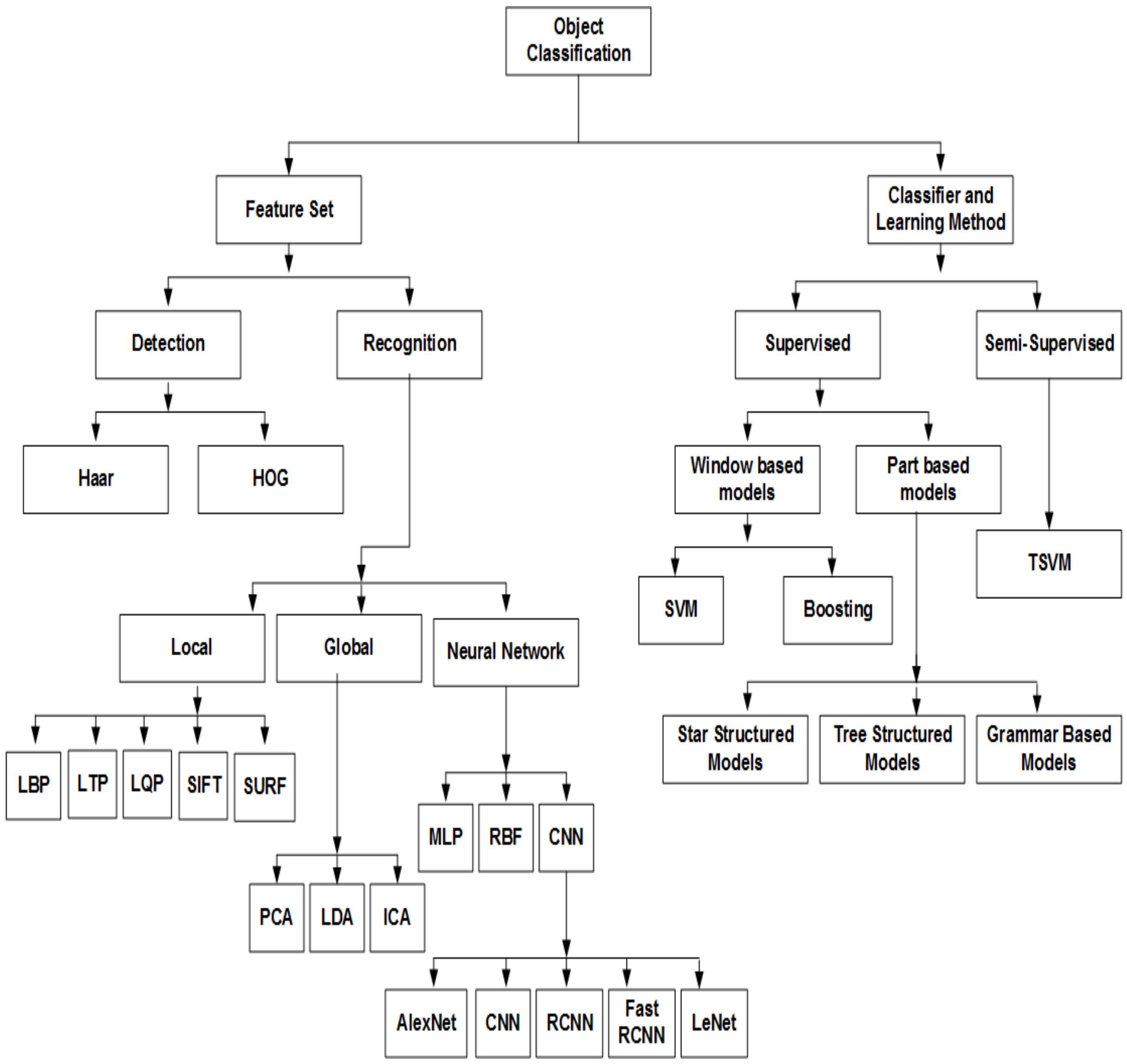


Figure 2.1: Object Classification Taxonomy

for feature extraction and classification. Figure 2.1 depicts the two classification steps needed to perform object classification. Feature extraction approaches are arranged in the left pan while classification approaches are arranged in the right hand side pan of the taxonomy.

2.2 Local Features Based Approaches

Local features based approaches work on smaller image areas. Each dimension of the local feature matrix embodies local areas or regions in the face image. Each dimension of feature

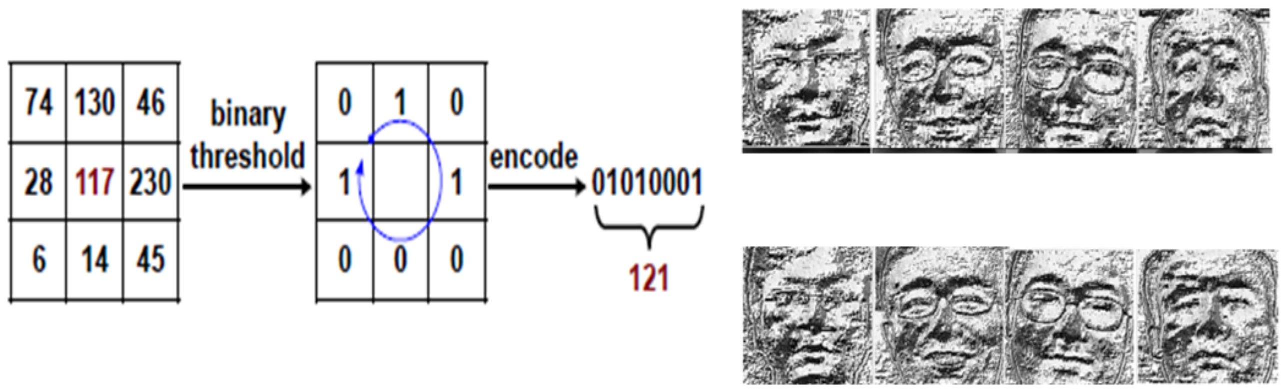


Figure 2.2: Local Binary Patterns

matrix provides more details of a specific local patch within an image. Local features are more robust to noise, illumination changes and occlusion as they work on small image areas. However, it is difficult to determine the importance of features and arbitrary decisions have to be taken to remove unnecessary features. If the feature set is unable to provide discriminative ability, no processing can compensate this deficiency.

2.2.1 Local Binary Patterns (LBP)

Local Binary Pattern features [11] are the most well-known local features based methods and are computed by dividing the examined window into cells. Each cell contains a certain number of pixels. Then each pixel in the cell is compared to its neighbouring pixels. If the value of centre pixel is greater than its neighbour pixel, 1 is written, and if it is the other way round then 0 is written as shown in Figure 2.2. Then the histogram is calculated over the frequency of each number occurrence. The LBP histogram is defined as;

$$H_i = \Sigma I\{f_i(x,y) = i\}, i = 0, ..., n-1$$

Normalized histograms give the feature vector of the window. This feature set can then be used with Support Vector Machine for classification, which is normally used for face recognition. The advantage of LBP features is that they provide acceptable accuracy rate and handle illumination very well. However, they are very sensitive to noise.

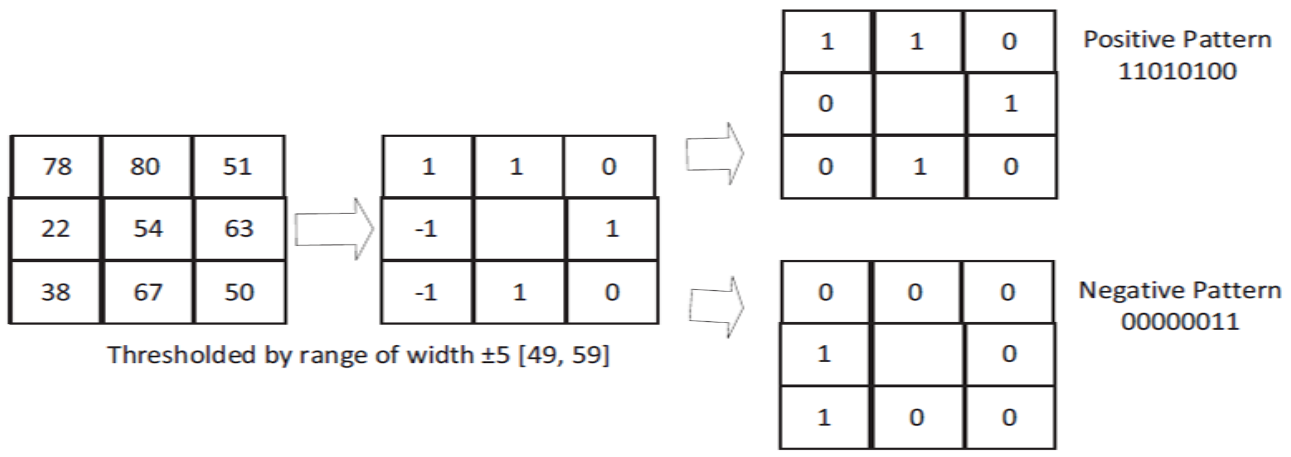


Figure 2.3: Local Ternary Patterns

2.2.2 Local Ternary Patterns (LTP)

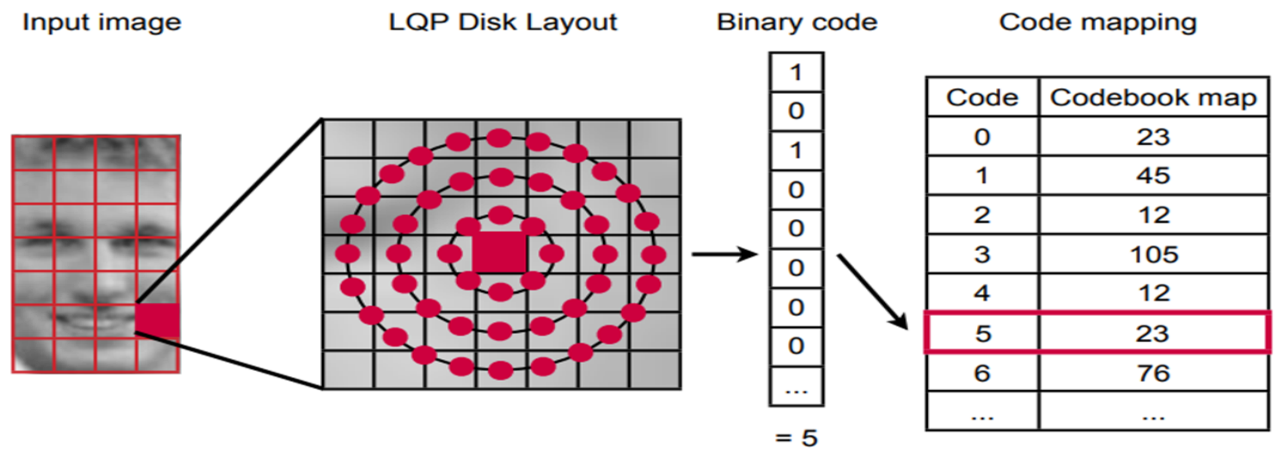
Based on the ternary threshold, Local Ternary Pattern features [12] as an extension of LBP were proposed. In LTP, the pixel difference between central pixel and neighbouring pixels is encoded into a ternary code. This ternary code is further split into positive LBP and negative LBP to reduce the dimensionality, as shown in Figure 2.3. This encoding of pixel difference into a separate state makes it more robust to noise.

$$\left\{ \begin{array}{cc} 1 & \text{if p} \geq \text{c+k} \\ 0 & \text{if p} \geq \text{c-k and p} \leq \text{c+k} \\ -1 & \text{if p} \leq \text{c-k} \end{array} \right\}$$

A pattern histogram, like in LBP, is then created by using the ternary values of neighbouring pixels. LTP features are more robust to noise than LBP but loss of information occurs when splitting into positive and negative LBP. Also, redundant information resides in the histograms of both LBPs, since they are strongly correlated.

2.2.3 Local Quantized Patterns (LQP)

Further, the idea of LBP and LTP was extended and Local Quantized Pattern features [13] were proposed. LQP features aims to group the dissimilar codes by employing clustering techniques.



Code	Codebook map
0	23
1	45
2	12
3	105
4	12
5	23
6	76
...	...

Figure 2.4: Local Quantized Patterns (LQP)

This helps to project the features into a lower dimension and a set of new features is obtained.

Initially, a table is used which contains all the codes. By using a clustering technique such as K-Means clustering [14], all the codes are mapped to the nearest cluster centres, as shown in Figure 2.4. Then, tables are built up consisting of the number of clusters and their associated codes. The division of codes into clusters results in the formation of a dictionary. The advantage of LQP features is that they are capable of holding large patterns as compared to LBP and LTP.

2.2.4 Gabor Wavelets

Gabor wavelets are powerful joint time-frequency based features and are based on Gabor filters [15]. Gabor wavelets have been used for years in numerous feature extraction algorithms. The Gabor features can be extracted from an image by using different frequencies and orientations of Gabor filters. Gabor filters can extract the edges and are mainly used to detect corners and blobs.

The Gabor function is a complex exponential function similar to sine and cosine used to detect frequencies in various directions. To perform edge detection, the convolution is performed in various dilated wavelets. A wavelet, which serves as a second order partial differential operator, is used for this purpose. The Gabor features provide optimal resolution in time and frequency domain but have high computational complexity. In extension to Gabor wavelets, another

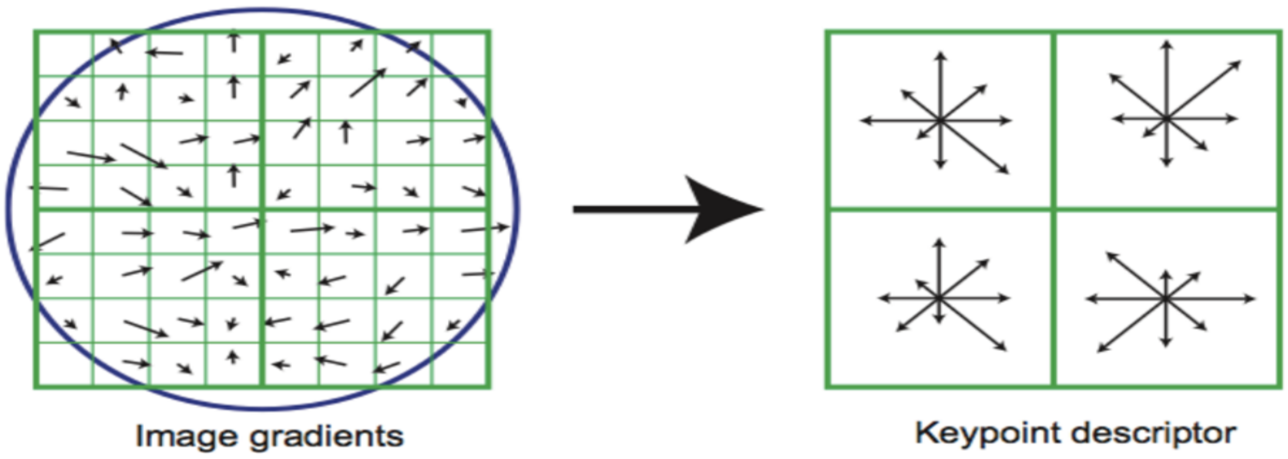


Figure 2.5: Scale-invariant Feature Transform (SIFT)

feature extraction approach, similar to local binary pattern, was introduced known as local XOR pattern and is shown in Figure 2.6. It calculates quadrant bit codes by using the XOR operator. The calculated values of LXP are then applied on the real and imaginary parts of the phase components.

2.2.5 Scale-invariant Feature Transform (SIFT)

SIFT [16] is another most commonly used local feature extraction approach. SIFT key-points are extracted from a set of images present in a database. To recognize an object in a new image, each feature from the new image is compared individually with the database. The candidate matching features are found out based on Euclidean distance of their feature vectors. From the full set of matches, the subsets of key-points that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches, as shown in Figure 2.5.

The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subjected to a further detailed model verification and subsequently, outliers are discarded. Finally, the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can then be identified as correct with

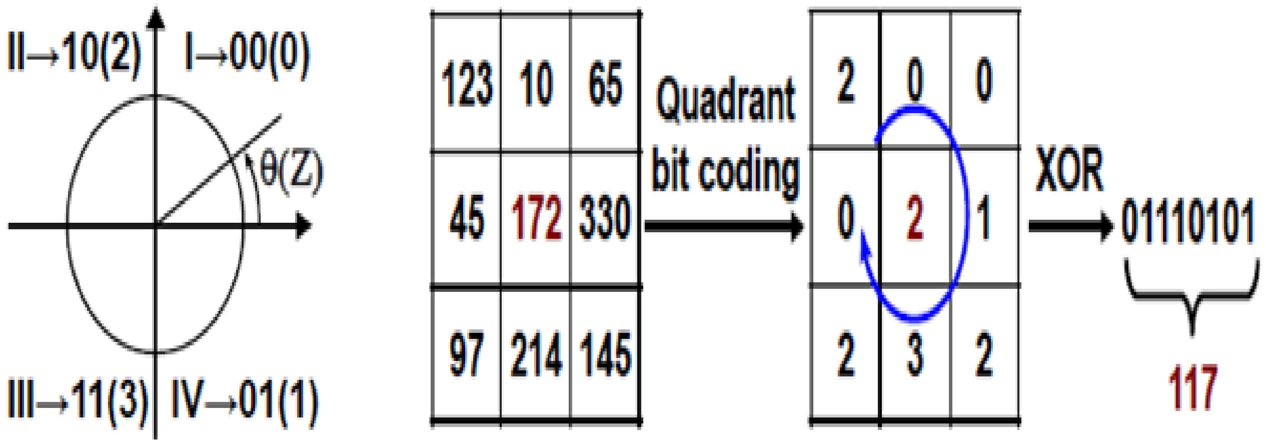


Figure 2.6: LXP

high confidence. SIFT features provide trustworthy recognition but have high computational cost and they are slow to compute.

2.2.6 Speeded Up Robust Features (SURF)

An extension of SIFT features is SURF, [17] which is based on Hessian matrix. Unlike SIFT, it computes LOG with a filter box. This approach of filter box really makes it rapid as compared to SIFT because convolution with filter box can be calculated easily with the help of integral image and can be computed in parallel for different scales. The determinant of Hessian matrix can be used for scale and orientation. Feature descriptors are calculated by using wavelet responses in horizontal and vertical direction. This is also performed with the help of integral images.

$$S(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j)$$

A circular region is drawn around the detected point of interest to assign unique orientation. SURF descriptors are constructed by extracting square regions around point of interest. A neighbourhood of size 20sX20s is taken around the point of interest where s is the size. It is divided into 4x4 sub-regions. For each sub-region, horizontal and vertical wavelet responses are taken and a vector is formed which gives SURF feature descriptor. SURF features are faster

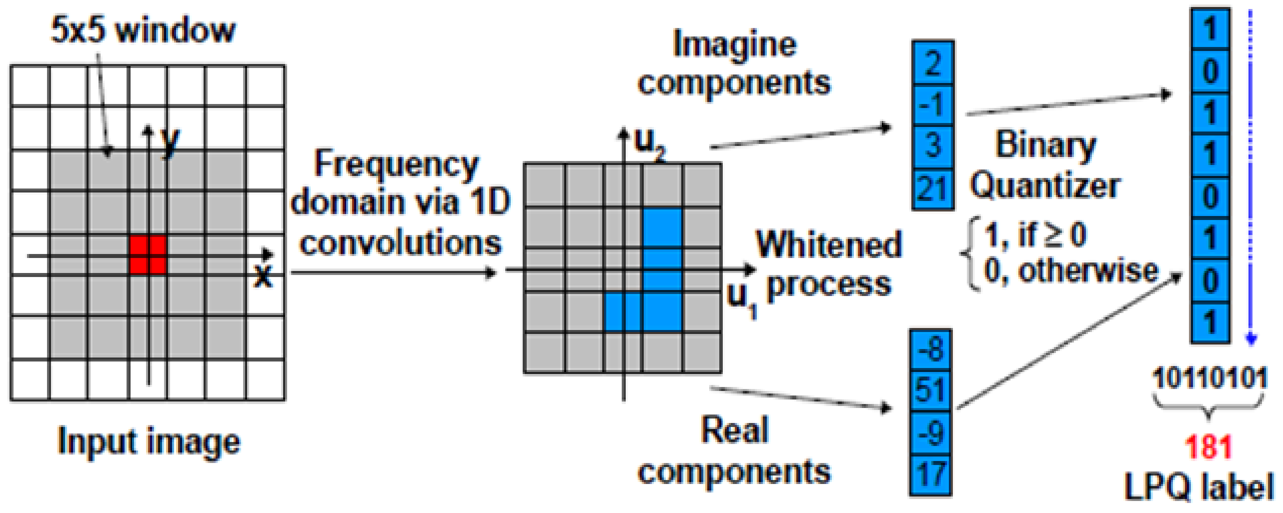


Figure 2.7: LPQ

to compute as compared to SIFT features.

2.2.7 Local Phase Quantization (LPQ)

Local Phase Quantization (LPQ) had been a major focus by researchers in the recent past. It was initially employed as a blur invariant descriptor [18] for texture classification. The methods based on local phase quantization proved to be much more successful as compared to Gabor based features when applied to the problems of texture classification. They worked really well especially in case of blurred textures. Inspired from the success of LPQ in texture classification, it was then later on applied for the face recognition and object classification problems as well. The LPQ features remained successful for blurred face images as well as for blurred textures.

LPQ utilizes the advantages of short-term fourier transform and is applied on a small window of size M x M. The real and imaginary components are then extracted from the image which undergoes to a binary quantization to have the LPQ label, as shown in Figure 2.7. Usually four real and imaginary components are extracted from the image. After applying the LPQ operator on the whole image, LPQ quantized image is achieved which is further used for recognition or classification of objects. LPQ operator was used by [19] on facial images and they also performed the histogram concatenation procedure on sub-regions of images just as it was performed in LBP and LTP. The difference was that the resultant feature vector was four times the length of LBP feature vector.

2.2.8 Haar Features

The Haar Cascade Classifier algorithm [20] is one of the most widely used algorithms for object detection. It is based on Haar like features and uses Adaboost [21] for dimension reduction and classifier training. Each feature is represented by a weak classifier in the cascade. A single weak classifier is represented as;

$$h(x, f, p) = \left\{ \begin{array}{ll} 1 & \text{if pf(x)} \leq \text{p}\theta \\ 0 & \text{otherwise} \end{array} \right\}$$

where 'f' is the feature value, and 'p' represents the polarity indicating the direction of the inequality. A combination of all these weak classifiers is used to make a strong classifier which is further used for detecting objects. The HOG features based person detection [22] works by dividing an image into multiple cells connected with each other. The algorithm then computes histograms of oriented gradients for each cell. These gradients describe the appearance of an object in an image and can be used to train a classifier for detection.

2.2.9 Multi-scale Descriptors

In order to further improve the performance of the above cited feature descriptors, these descriptors are also applied in combination with multiple elementary descriptors. An example of these is the combination of LBP, LTP and LPR. These descriptors are also applied by changing their associated parameters such as neighbourhood size, shape of neighbourhood (circular, elliptical) and varying radii.

Some of the major examples of multi-scale descriptors include multi-scale local phase quantization (MLPQ), multi-scale local binary pattern (MLPB) and multi-scale local ternary pattern (MLTP). [23] made the use of kernel discriminant analysis along with multi-scale local phase quantization to improve the performance of the descriptor. Multi-scale local phase quantization

(MLPQ) has also been used with linear discriminant analysis to improve the performance. [24] used multi-scale local phase quantization (MLPQ) with linear regression classifier (LRC) for recognition purposes.

[25] proposed a variant of Local Phase Quantization called LFD in which they utilized the magnitude information as well as the phase information of the images. The phase and magnitude information was generated from the short-term Fourier transform. The LBP encoding scheme was applied on the magnitude information of the image, while binary qualification was applied on the phase information of the image. The two encoded images in combination created the LFD feature vector by concatenating the histograms of these images. The proposed method proved to be a good approach for low resolution images.

All these variations in the elementary descriptors could improve the overall performance to some extent but there is always a trade-off between performance and computational cost. Sometimes it may occur that the performance gain might not be worth-while. On the other hand it may increase the memory and computational requirement to a high extent. These descriptors are normally referred to as multi-scale descriptors and have been proposed by a number of authors in the recent past.

Table 2.1 shows the performance measures of local features based approaches. As discussed above, the local pattern features are obtained from small local patches. In case of video data, these features are extracted from the subsequent video frames and are then aggregated to produce combined feature vectors which are then used for appearance and motion information. This phenomenon tends to produce high dimensional feature vectors which makes it difficult to use them for large scale video processing. Also, these systems are not very successful with the video streams captured under uncontrolled environmental conditions and have resulted in a drop of accuracy and precision.

The local features based approaches which are reviewed in this section work well for small dataset sizes. These approaches work on smaller image areas and, therefore, provide good accuracy rate. However, these approaches consider each pixel of the object, which makes them time consuming. So, there is a trade-off between accuracy, performance and computational

Table 2.1: Performance measures of local features based approaches

Approaches	Recognition Rate	Performance		Drawbacks	Advantages
		Image Size	Time		
LBP	92.6 %	110 x 150	0.03s	Sensitive to noise	High accuracy rate
LTP	98.7 %	128 x 128	50ms	Contains redundant information	Robust to noise
LQP	98.6 %	90 x 150	0.04s	Requires large lookup tables	Labelled dataset is not required
Gabor	92.2 %	110 x 150	30s	High computational complexity	Optimal in time and frequency
SIFT	95.9 %	256 x 384	0.91ms	High computational complexity	High recognition rate
SURF	96.0 %	256 x 384	0.59ms	Prone to illumination variance	Faster to compute
ORB	72.8 %	640 x 480	15.3ms	Scale invariance is not tested	Resistant to noise

cost for these approaches. We have made the use of these approaches in our proposed workflow for object extraction which will be explained in next chapter of this thesis.

2.3 Global Features Based Approaches

Global features based approaches extract features from the entire image and usually represents the holistic facial information like the contour of a face. The global features matrix contains all parts of the image and includes each and every pixel of it. This is why no information in the images is destroyed by global features. Global features are good in representing the coarse representation of an image. However, since all the information of image is retained in these features, they are high in dimension. They work on the assumption that all the features are equally important which is not true in most of the cases. This makes these techniques computationally intensive.

2.3.1 Principal Component Analysis (PCA)

The first and foremost global features based method called Eigenfaces, [26] shown in Figure 2.8 is based on principal component analysis [27] which is a statistical technique and comes under

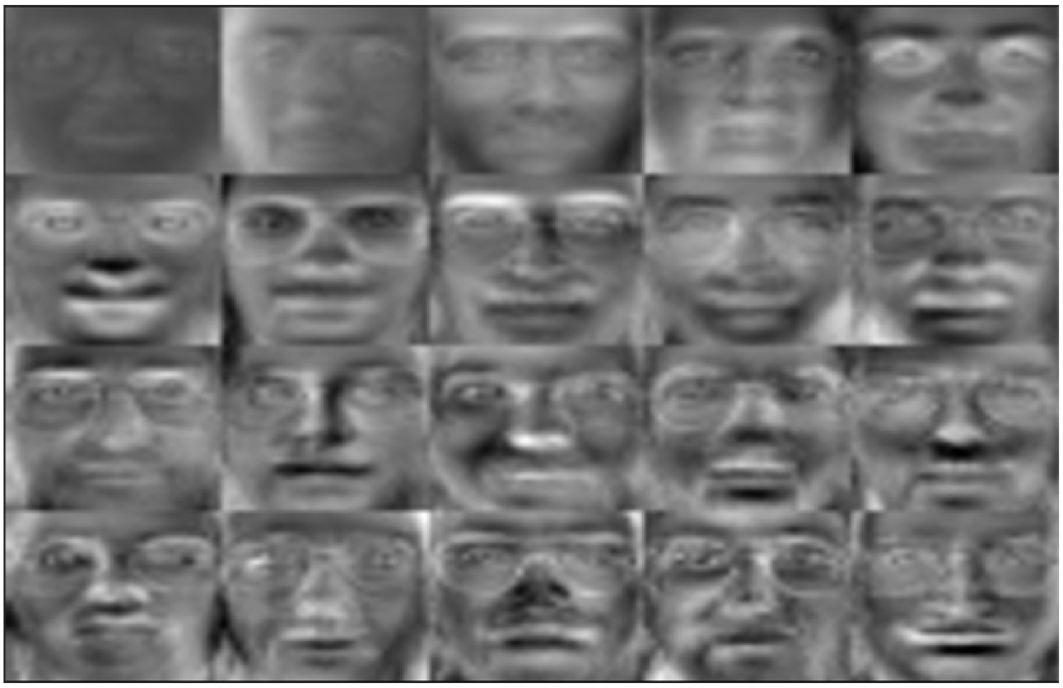


Figure 2.8: EigenFaces Approach

the category of holistic approaches. This technique was initially used to reduce or simplify the data. It projects the data from a higher dimensional space to a lower dimensional space in such a way that data with higher variance lies on the first axis. This is usually said to be the first principal component of data as it contains data having higher variance.

The 'K' principle components can be given by;

$$X_k = X - \sum_{s=1}^{k-1} X w_s w^T$$

where 'W' represents the weight vectors and 'X' represents the vector of principal component scores. The second principal component similarly contains the data which have the second greatest variance. All the remaining components contain the data in order of their variance. These components can also be treated as features when it comes to the classification task. Principal components of a face image are usually known as Eigenfaces and can be used for classification. Facial recognition performed by PCA is insensitive to facial expressions. However, the performance degrades in extreme lighting conditions.

2.3.2 Linear Discriminant Analysis (LDA)

Motivated from Eigenfaces, Fisherfaces is another holistic approach and is based on Linear discriminant analysis [28]. It proved to outperform PCA in face recognition tasks under complex conditions. The objective of LDA is to clearly distinguish between the samples of the same class and the samples of a different class. It tends to group the images belonging to same class and creates a separation from images of different classes.

Linear discriminant analysis ends up with a discriminant function that linearly separates samples of different classes. This linear transformation function provides a well-defined class separation by increasing the ratio between class-variance and within-class variance. LDA provides a way to overcome the shortcomings of PCA approach but it can face the small sample size problem.

2.3.3 Independent Component Analysis (ICA)

Independent component analysis [29] is a generalization of principal component analysis. It is a widely used method for projection of high dimensional data to a low dimensional sub-space. The objective of ICA is same as PCA but it generates spatially localized features. In contrast to PCA, the components generated by ICA are statistically independent. No information in images is destroyed by using this technique. But one also has to compromise on redundant information present in the images. This also makes this technique computationally expensive.

2.3.4 Multidimensional Scaling (MDS)

Multidimensional scaling (MDS) [30] which is very similar to the principal component analysis is often used to reduce the linear dimension. Unlike principal component analysis which retains the data variance while projection, MDS retains the distance between two examples. Another technique similar to principal component analysis is the non-negative matrix factorization (NMF) [31] which represents the facial information as a vector combination. This vector combination is linear and does not include the negative vectors. The absence of negative vectors from the projection enables it to reflect the facial representation better than principal component analysis.

A number of improvements have been proposed in all the three holistic approaches namely PCA, ICA and LDA. These improvements have been made in order to improve the accuracy under illumination and pose variations. This was achieved by utilizing probabilistic subspace framework. Also, direct linear discriminant analysis (D-LDA) and fractional linear discriminant analysis (F-LDA) are used together by [32] to reduce the number of miss-classifications that could occur by the closeness of category products.

Principal component analysis and linear discriminant analysis have also been used with shallow neural networks to perform robust and accurate facial recognition [26]. These two techniques are used in combination in order to extract features from the face. To perform face alignment, normalization and face enhancement, Gabor wavelets were utilized. Principal component analysis and linear discriminant analysis proved to be good in extracting discriminant features from face which were later classified through neural network using back propagation algorithm for learning.

The use of principal component analysis has also been made with discrete cosine transform (DCT) to perform recognition in HMM [33]. Principal component analysis has been used to perform dimension reduction. The recognition is performed by dividing the object into blocks and then discrete cosine transform is applied on each block to perform recognition. The use of hidden markov model (HMM) has also been made with local binary patterns in [34]. They used the same approach as [33] and divided the face image into blocks for recognition.

The global features based approaches were successful in some of the areas but remained unstable as compared to local features based approaches. The global features were unable to describe the underlying subtleties of features and their geometric varieties that were present in the original image. They were also unable to handle the non-linearity in object classification and remained limited within their scope. The performance of the global features based methods can get adverse if their concavities are fulfilled and deformations are smoothed. Table 2.2 shows the

Table 2.2: Performance measures of global features based approaches

Approaches	Recognition Rate	Performance		Drawbacks	Advantages
		Image Size	Time		
PCA	70 %	85 x 60	0.3s	Performance degrades in extreme conditions	Insensitive to facial expressions
LDA	88 %	85 x 60	0.4s	Contains redundant information	Resistant to illumination conditions
ICA	89 %	60 x 50	0.2s	High computation cost	Resistant to illumination conditions

performance measures of global features based approaches.

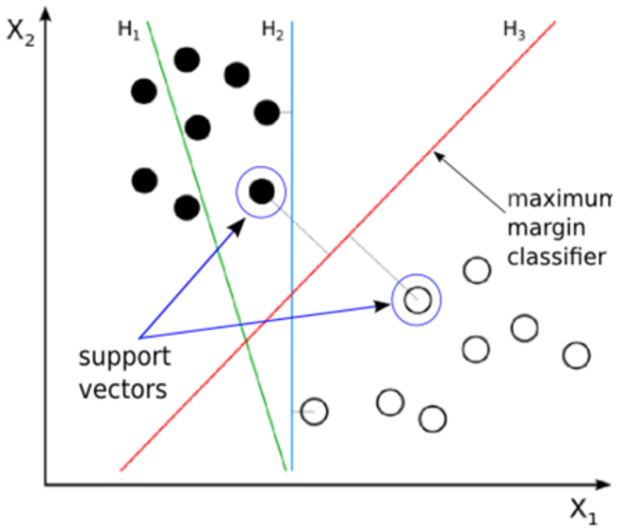
We have made the use of principal component analysis for dimension reduction in this research work. The use of spatial-frequency domain results in the generation of large number of features. So we have made the use of principal component analysis in order to reduce the dimension of features. Feature selection mechanism has also been adopted to get rid of irrelevant features. This helped to reduce the computation cost of overall system.

2.4 Classification Algorithms

The feature extraction step ends up in the generation of large number of features. These features are then fed to the classification algorithm to perform classification of objects. The most commonly used classification algorithms used so far are described below:

2.4.1 Decision Trees

Decision Trees [35] are used to perform instance classification based on trees. The instances are first sorted depending upon their feature value. These features present in an instance which are to be classified are then represented as nodes in the decision trees. The value of the feature or the node is represented by a branch in the decision trees. The classification of instances is



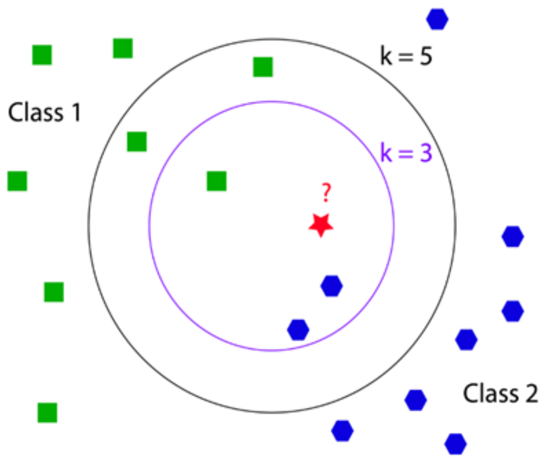


Figure 2.9: Support Vector Machine and K-Nearest Neighbour

performed by sorting the values starting from the root node.

The root node has the features which are proved to be the best in dividing the training dataset. Information gain [36] and gini index [37] are the processes which are normally used to identify the features which can best divide the training dataset. This procedure is repeated on the remaining datasets and a number of sub-trees are created. One of the major advantages of decision tress is that it is easy to track the classification of an instance made by the decision tree.

2.4.2 Support Vector Machine

Support Vector Machine [38] is another well-known classification algorithm that works under the supervised learning domain. It needs to be trained on a set of given training examples already marked with predefined classes. SVM can then categorize any incoming new example into one of the predefined classes. Although it works for linear classification, it can also be used for non-linear classification by using the kernel trick.

In order to perform the classification, a hyperplane is created by the SVM, as shown in Figure 2.9. A hyperplane having the greatest distance from the training data set points gives the most accurate classification results. After obtaining features from the feature extraction phase, SVM is the most widely used classifier for object detection and recognition.

An extension of SVM is Transductive Support Vector Machine (TSVM) [39]. The objective of TSVM is to classify data that is partially labelled, such as in the domain of semi-supervised learning. TSVM uses the technique of regularization to obtain separation between labelled and unlabelled data. To achieve good results tuning of TSVM is required which also helps to avoid unstable performance.

2.4.3 Boosting

Boosting [40] has also been used for classification purposes. It is a technique to develop a strong learner with the help of many weak learners. A weak learner can be said as a classifier which performs at least better than the random guessing and is a little bit correlated to true classification. Boosting works in an iterative process. In each iteration, learning of weak classifier is performed and is added to the ensemble to make a strong classifier. Each weak learner is weighted according to its accuracy. When all the weak classifiers are added to the ensemble, the data is again re-weighted and those examples which have been misclassified are assigned with higher weights and those which are correctly classified are assigned with lower weights. This helps the future weak learners to concentrate more on misclassified examples.

Instance based learning is another technique which comes under the category of statistical learning methods. They are also known as the lazy learning methods because the generalization or induction process is delayed in them till the classification process takes place. The advantage of instance based learning algorithms is that they require less time for training as compared to other methods such as neural networks or decision trees. But on the other hand, they also require more time while performing classification.

2.4.4 K-nearest Neighbour

K-nearest Neighbour, [41] is one of the examples of instance based learning and is a well-known algorithm used both for classification and regression. As the name suggests, it performs classification on the basis of voting by the neighbours of object to be classified. The object is assigned to the class from which the majority of votes are received as shown in Figure 2.9. Weights can also be assigned to the votes of the neighbours, so the nearest neighbours will have higher contribution in the weights than the ones which are far away.

To train the classifier, training examples in the form of vectors, with pre-assigned classes, are used. Training only consists of the piling up of feature vectors with their class labels. Then in classification, the point (k) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. Usually, Euclidean distance is used to calculate the distance between test and neighbouring points.

2.4.5 Bayesian Networks

Bayesian Networks come under the category of statistical learning algorithms which represents the relationships between different variables with the help of graphical model. The most common graphical model is the directed acyclic graph in which the nodes of the graph have a one to one relationship with the features. The edges of the model represent the influences between different features. The learning in the Bayesian Network is achieved through two sub-processes. In first sub-process the structure of the directed acyclic graph is learned and then its parameters are determined in the second sub-process. A table is generated consisting of the probabilistic parameters in the encoded format. Each variable holds one table in the network and the multiplication of these tables can result in the joint distribution of the network.

Naive Bayes [42] which is one of the most common Bayesian Network, is another technique for classification in which we have vectors of feature values which assign labels to unknown instances. It is mainly used for problems in which the dimensionality of input is high. Naive Bayes Classifier is based on Bayesian theorem and works on the principle that the value of any feature is not dependent on the value of any other feature. It uses the concept of prior probability and likelihood. Prior probabilities are those probabilities which are achieved on the grounds of some previous experience. Once prior probability is assigned, then the likelihood is calculated by drawing a circle around the unknown object and to see how many instances of both the classes are in the circle. In this way, one class will have smaller likelihood than the other. By combining this likelihood with the prior probability, final classification can be achieved.

2.5 Learning Algorithms for Intelligent Analysis

Learning algorithms such as Neural Networks are proven to be good in various image recognition tasks. They have also been recently tested on video classification problem using convolutional neural network. They have been used in face analysis such as recognition and expression recognition. However, the training time of Convolutional Neural Network is quite large. A large number of training examples are required to tune the parameters of the model.

2.5.1 Artificial Neural Networks (ANNs)

The basic building blocks of Artificial Neural Networks [43] are neurons which are interconnected and send messages to each other, as shown in Figure 2.10. The connections in ANN are weighted and these weights are updated as the system learns with experience. These changing of weights enable the system to adapt and learn with the passage of time. A simple Neural Network normally has three layers; input layer, hidden layer and output layer. The function of input layer is to take input data and send it to the following layer, i.e. hidden layer, as synapses. More complex systems can have more than one hidden layer. These synapses have weights which keep modifying themselves and manipulate the data calculations. The neurons weighted input is converted to output via an activation function. Back Propagation is the most commonly used algorithm in ANN for learning.

2.5.2 Radial Basis Function Networks (RBFs)

Radial Basis Function Network [44] is a type of Artificial Neural Network. Radial basis function is used as an activation function in RBFs. This radial basis function is used in combination

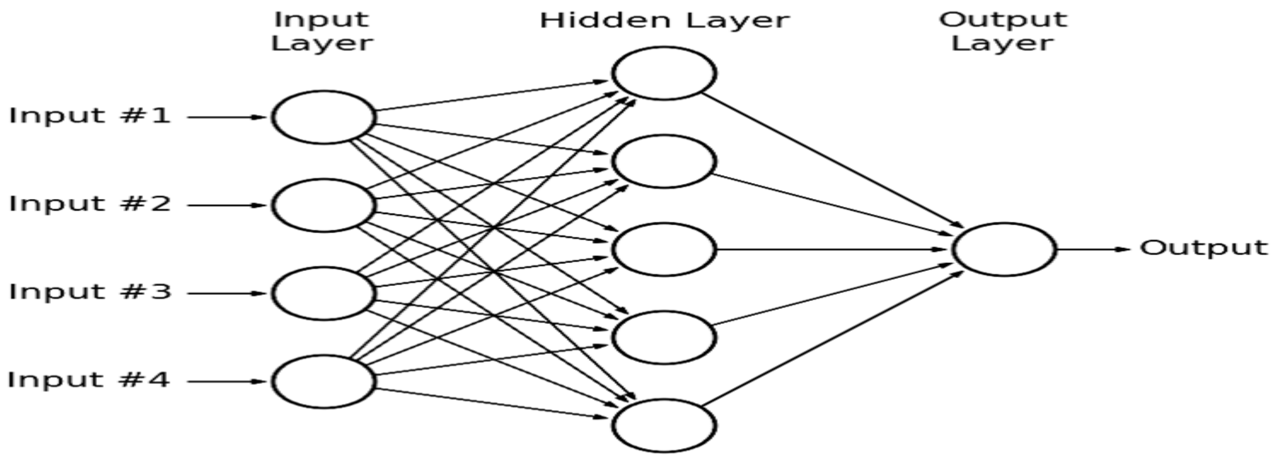


Figure 2.10: Artificial Neural Network

with the neuron parameters to generate the network output. These networks normally consist of three layers. The first layer is the typical input layer, the second layer is the hidden layer which contains RBF activation function. The third layer is the output layer which produces the output. These networks have fast learning speed and are appropriate for lesser number of classes.

Artificial Neural Networks have emerged as influential tools for detection and tracking of objects from video streams. However, these algorithms were proven not to be applicable for the analysis of large scale data as large number of parameters were needed to be trained requiring abundant computing resources. With the advent of modern hardware resources including edge and cloud platform, the limitation of computing resources is solved. This led to the development of more advance form of learning algorithms known as Deep Learning Algorithms. These algorithms have the capability to solve practical problems such as pattern classification and recognition on large scale data but require more computation resources for training.

2.5.3 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network [45] as shown in Figure 2.11 is a kind of multilayer Neural Network but it has some additional layers including convolutional and pooling or sub-sampling layers. These additional layers are then followed to fully connected layers as in a standard multilayer perceptron. The convolutional layers contain a set of convolution kernels. Each

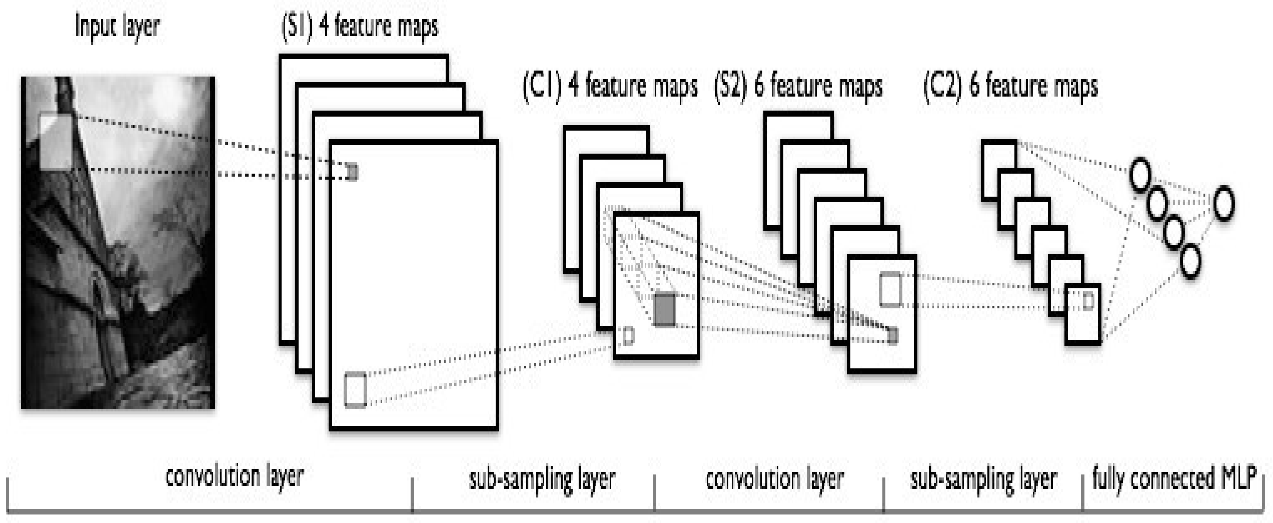


Figure 2.11: Convolutional Neural Network

kernel is a filter or mask which is convolved with the image to extract features.

There can be many convolution kernels in each layer. This results in the formation of feature maps. The pooling layer is next to convolution layers. The purpose of this layer is to subsample a small rectangular box or feature map taken from convolution layer in order to reduce variance. A single output from each rectangular block is generated. A number of convolution and pooling layers lead to the fully connected multi-layer perceptron (MLP) layer. An MLP is a standard Multilayer Neural Network. This layer takes all the neurons from previous layer and connects it to every single layer within itself.

CNNs are most suitable for images as their architecture is designed in a way as to take full advantage of 2D structure. In this case, the input to the convolution layer is a 2D image and the kernels are applied to it. The convolution with the image produces feature maps. Next, mapping is performed on each feature map. This further leads to fully connected layers i.e. MLP layers.

2.5.4 Regional Convolutional Neural Networks (R-CNNs)

Szegedy et al. [46] proposed to modify the state-of-the art deep CNNs by replacing the last layer of the network with a regression layer. This modification resulted in the average precision

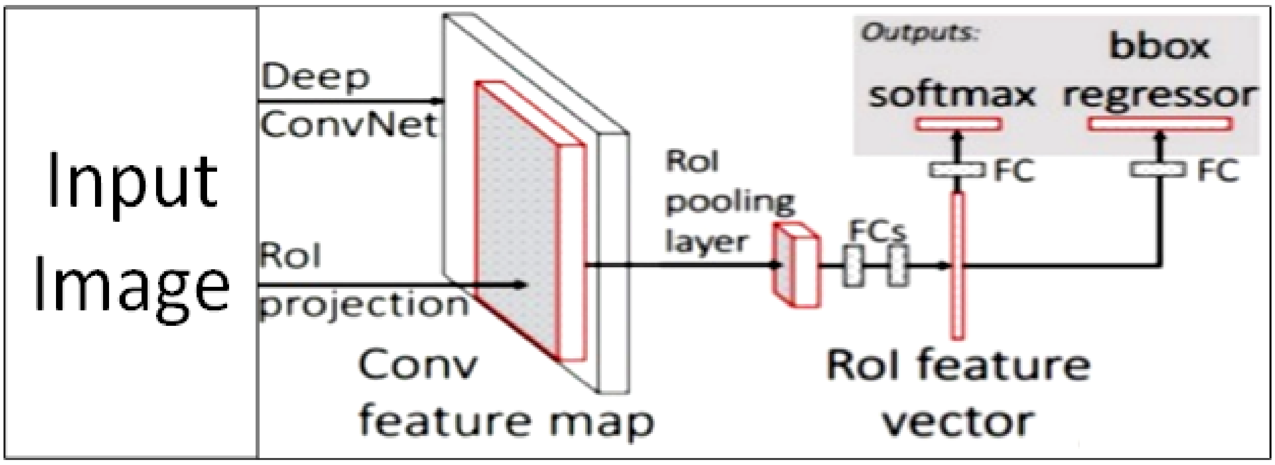


Figure 2.12: Regional Convolutional Neural Network

of 0:305 over 20 classes on Pascal Visual Object Challenge (VOC). As opposed to Szegedys proposed model, Girshick et al. [47] adopted a bottom-up region based deep model called R-CNN as shown in Figure 2.12. The proposed model generated two thousand region proposals and a CNN was used for feature extraction from each region which were then classified by SVMs. An improvement of 30 percent in accuracy was observed but it was slow as training was a multi-stage pipeline.

2.5.5 Fast and Faster R-CNNs

Ross Girshick [48] further improved their method and proposed a fast region-based convolutional network method called Fast R-CNN to detect objects rapidly. This method reported higher detection accuracy and performed training in a single stage using a multi-task loss. Disk storage was also not required as in the case of R-CNN.

Shaoqing Ren et al. [49] further improved Girshick's work and proposed Faster R-CNN and reduced the computation of region proposal by sharing full image features with the detection network. They also combined Fast R-CNN and RPN by sharing their convolutional features into a single network. This method outperformed both R-CNN and Fast R-CNN on publicly available image datasets.

Table 2.3: Performance measures of neural network based approaches

Approaches	Recognition Rate	Drawbacks	Advantages
MLP	79 %	Prone to over-fitting	Multiple training algorithms are available
RBF	98 %	Suitable for small classes	Fast learning speed
CNN	97 %	High computation cost	Easy to migrate to parallel hardware

2.5.6 You Only Look Once (YOLO)

Joseph Redmon et al. [50] presented You Only Look Once (YOLO), which detects objects in one evaluation of CNN. It resizes the images to 448 x 448 and executes a single pass of CNN on the image to detect the objects and it outperformed R-CNN. However, all these works have been proposed to perform detection and classification tasks on still images. It is more likely that leveraging these methods for videos, oversight some positive illustrations as the objects being classified might not be in their best position in each video frame. Few recent works have also investigated video classification for multimedia data using deep networks.

2.5.7 Stacked Auto-encoders

A number of studies in the recent past have employed deep learning networks for object detection and classification purposes. Wang and Yeung [51] suggested to learn deep features for object tracking by using stacked denoising autoencoder [52]. Stacked auto-encoders were used to reconstruct the input data at the output by using the hidden nodes. The input data is projected in to a low dimensional space and the hidden layers were used to learn the representation from the training examples. It was reported that using the learned deep features outperformed other 7 state-of-the-art trackers. Table 2.3 shows the performance measures of some of the neural network based approaches.

2.5.8 CNNs for Detection and Classification

2.5.8.1 CNNs for Image Data

Several approaches [53] [54] employed CNNs to learn features from raw pixels primarily for still images. Most recently Alex Krizhevsky et al. [55] proposed deep CNNs for ImageNet [56] dataset and achieved high accuracy. Similarly Gil Levi et al. [57] used CNN to perform age and gender classification.

Dan Ciresan et al. [58] proposed the use of multi-column deep neural networks for image classification and performed their experimentation on MNIST [59], CIFAR [60] and NORB [61] datasets. Yaniv Taigman et al. [62] proposed DeepFace to perform face verification from facial images and reported an accuracy of 97.35 percent on the Labeled Faces in the Wild (LFW) dataset [63].

Li et al. [52] proposed to learn feature representations for object tracking by using CNNs. They suggested to use a pool of multiple CNNs to maintain different kernels regarding all possible low-level cues. These low-level cues helped to distinguish object patches from their surroundings. In a specific frame, the most prospective convolutional neural network was utilized to predict the new location of the target object.

Wang et al. [64] suggested to learn deep hierarchical features that should be robust to target appearance changes. First, the generic features were learned off-line from video data; and then pre-learned features were modified on-line according to the particular target object class. Consequently, the modified features were able to capture appearance changes of target objects. It was reported that using the proposed algorithm can significantly improve tracking performance. The robust detection, tracking and identification of targets among multiple cameras facilitate scene understanding thus leading to the development of appropriate situation awareness mechanisms.

Giang et al. [65] used CNNs to differentiate between pedestrians and non-pedestrians in an urban environment. They input images are scanned at different scales, and at each scale all windows of fixed size are processed by a CNN classifier to determine whether an input window is pedestrian or not. Feature extraction and classification phases are integrated in one single fully-adaptive structure. All three layers of CNN i.e. convolution layers, sub-sampling layers, and output layers are used to perform classification. This work showed that it is possible to lower training time while maintaining a threshold classification rate.

However, all of these pipelines are used to perform vision tasks in still images. Leveraging these approaches for videos can oversight some positive illustrations as the objects being classified might not be in their best poses and conditions in each frame of the video.

2.5.8.2 CNNs for Video Data

Some related work in the recent past investigated video classification for multimedia data using deep networks. Kai Kang et al. [66] used CNNs to detect objects from video tubelets. They also proposed a temporal CNN to combine temporal information to regularize the detection outcomes. Zhongwen Xu et al. [67] proposed discriminative CNN video representation to perform event detection from video dataset. Andrej Karpathy et al. [68], Joe Yue-Hei Ng [69] and Shengxin Zha [70] used CNN architectures to perform video classification. They also retrained the top layers of their systems to study the generalization performance of their models.

Karen Simonyan et al. [71] and liu et al. [72] also used CNNs to perform action recognition from video streams. However, these approaches do not shed any light on the behaviour of the system by varying the hyper-parameters of the deep network. The analysis of the system and its behaviour on how it reflects on the changing values of parameters is scarce in state-of-the art. It is even rarer for video analytics systems. We analyse these parameters both mathematically and architecturally and propose the most appropriate tuning parameters for any video analytics system.

Erhan et al. [73] suggested a saliency-inspired deep CNN for the detection of multiple objects of interest belonging to any class. However, it was not very successful and produced a small number of bounding boxes as object candidates. Yuan Yuan et al. [74] proposed a video-based

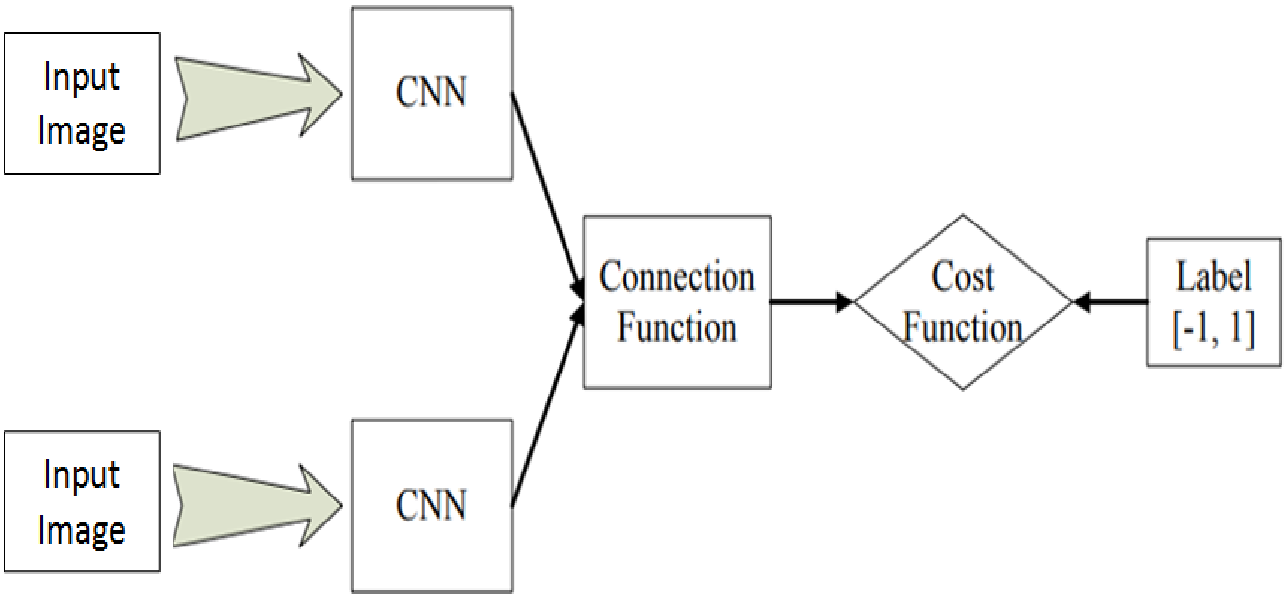


Figure 2.13: Siamese Convolutional Neural Network

vehicle detection and tracking system. They proposed tracking by detection framework in which a sequential detection model is presented to tackle with the problem of occlusions. In order to track the vehicles, they modelled group behavior to handle complex interactions with overlaps and ambiguities among vehicles. The model proved to be effective for real surveillance videos at road junctions.

In another study Qi Wang et al. [75] proposed the use of siamesed fully CNN to perform road detection. They made the use of RGB-channel images, semantic contours, and location to segment the road region. The training speed of the proposed model proved to be 30 percent faster than the original fully connected network because of the guidance of highly structured contours.

Few recent studies have made the use of deep learning based approaches for video analytics. Li et al. [76] proposed a filter pairing neural network. They jointly optimized all the components of the system to handle geometric and photometric transforms and achieved good results on their self-generated dataset. Similarly Yi et al. [77] proposed a siamese CNN as shown in Figure 2.13 to learn a similarity metric from pixels in a unified framework.

In another study ahmed et al. [78], a deep convolutional architecture was proposed to simultaneously learn features and corresponding similarity metric. However, they input a pair of images and tried to find a similarity value to indicate whether the two images belong to same object or not. They performed more of a verification task than classification of objects.

Most of the existing deep learning based video analytics approaches lack automation and require human assistance to perform object classification. These approaches do not shed any light on the behaviour and performance of the system by varying the hyper-parameters of the deep learning model. The investigation on the behaviour of the system and its analysis on how it reflects on the varying values of hyper-parameters is scarce in recent literature. This analysis of the system, specifically for cloud based video analytics is even a less focused area.

Deep learning networks have the capability to perform classification and recognition on large scale data as compared to shallow networks but require more computational resources for training. They also pose a number of other challenges on difficult tasks, such as hyper-parameter tuning of the deep network, increasing training times and scarce availability of labeled data.

Convolutional Neural Network based video analytics systems proved to be successful as compared to shallow networks recently. However, these systems are still in their infancy stage and pose a number of challenges on difficult tasks. Most of the proposed approaches are still struggling in coping with the major challenges such as hyper-parameter tuning of the CNN, increasing training times and scarce availability of labelled data.

2.5.9 Multi-model based Approaches

A number of studies [79] [80] and [81] worked on multi-model features including audio features, lip motion features and face images to perform person identification. They used Hidden Markov Models [82] for audio features, guassian density functions [83] were used for lip motion and different local pattern features and eigen [84] and fisherfaces [85] were used for face images.

Yi et al. [86] used a probabilistic fusion scheme to fuse the features from different modalities and then performed the identification. Li et al. [87] proposed a rank-level fusion approach to fuse the features from different modalities. Then a majority voting rule was applied to perform decision. However, all these approaches need to have a good compromise between computational cost and performance as the use of multi-model features produce high dimensional feature vectors and made them incapable for large scale video processing.

Another study utilized Electrocardiograms (ECG) signals to perform person identification. Bassiouni et al. [88] made the use of auto-correlation and discrete cosine transform to extract features from ECG signals. These features were then fed to Artificial Neural Network to perform classification.

However, the ECG signals are difficult to obtain and cannot be used for general-purpose person re-identification system. Audio and facial features have also been used in the past [89] to perform person identification. Mel-frequency cepstral coefficients [90] were extracted from audio signals and vector quantization was used for classification.

For facial features, eigenfaces [84], fisherfaces [85] and principle component analysis [91] has been used widely. Smowton et al. [92] performed KNN based classification on these features to perform classification. They also mixed these features with wavelet transform features extracted from visual lip reading. HMM [82] based approaches were then used for identification.

Loris et al. [93] proposed an ensemble of different approaches to develop a person re-identification system. They combined different widely used re-identification systems based on color spaces. They extracted both the color and texture features from the images and made a comparison of them on the basis of distance measures. Other studies [94], [95] also exploited the use of color features but all these approaches built upon the color and texture properties fails in the case of strong illumination conditions.

Some approaches [96] made the use of local features and generated a set of key-points to produce the signature of a target person. But the accuracy of these approaches is highly dependent on the performance of key-point detector. Another study [97] made the use of 3D data from depth cameras and skeletal tracker [98] was used for identification. This approach provided good results but its efficiency is dependent on the skeletal tracker. If the tracker fails then the generated signatures are deemed useless.

2.6 Object Detection and Classification in Spatial Frequency Domain

Object classification has been the focus of many studies for the last several decades. However, classifying objects from video streams under uncontrolled conditions poses many challenges and is now acquiring much attention from the research community.

2.6.1 Gabor Wavelets based Methods

Gabor wavelets have been used recently as a monogenic filter for classification purposes. It has been used with local binary patterns known as Monogenic Binary Pattern (MBP) [99] and Monogenic Binary Coding(MBC) [100] . The Gabor wavelet as a monogenic filter can produce multiple frequency components. These components include amplitude or sometimes known as magnitude, orientation and phase.

The Monogenic Binary Pattern (MBP) was used along with quadrant bit coding in [100] on the orientation components. This procedure produced Monogenic Binary Pattern at three scales. In order to perform the classification, weighted intersection metric was used between the test and gallery image as a similarity measure. This procedure performed much better as compared to [101] and [102] in terms of accuracy. [100] applied local binary patterns on the amplitude property, and LXP on the phase property and achieved good results as compared to other Gabor wavelets based methods.

Multi-Scale Methods have also been used with monogenic components. Different variants of Local Binary Patterns including Local Gabor Binary Pattern Histogram (LGBPHS) [101] made the use of monogenic components with local pattern features as shown in Figure 2.14. LGBPHS employs a feature fusion scheme. Features are generated by employing local binary patterns on the amplitude property generated from the Gabor wavelet. Generated feature vector was used with template matching framework to perform classification.

In another approach piecewise FDA [103] was proposed. Piecewise FDA works on smaller image

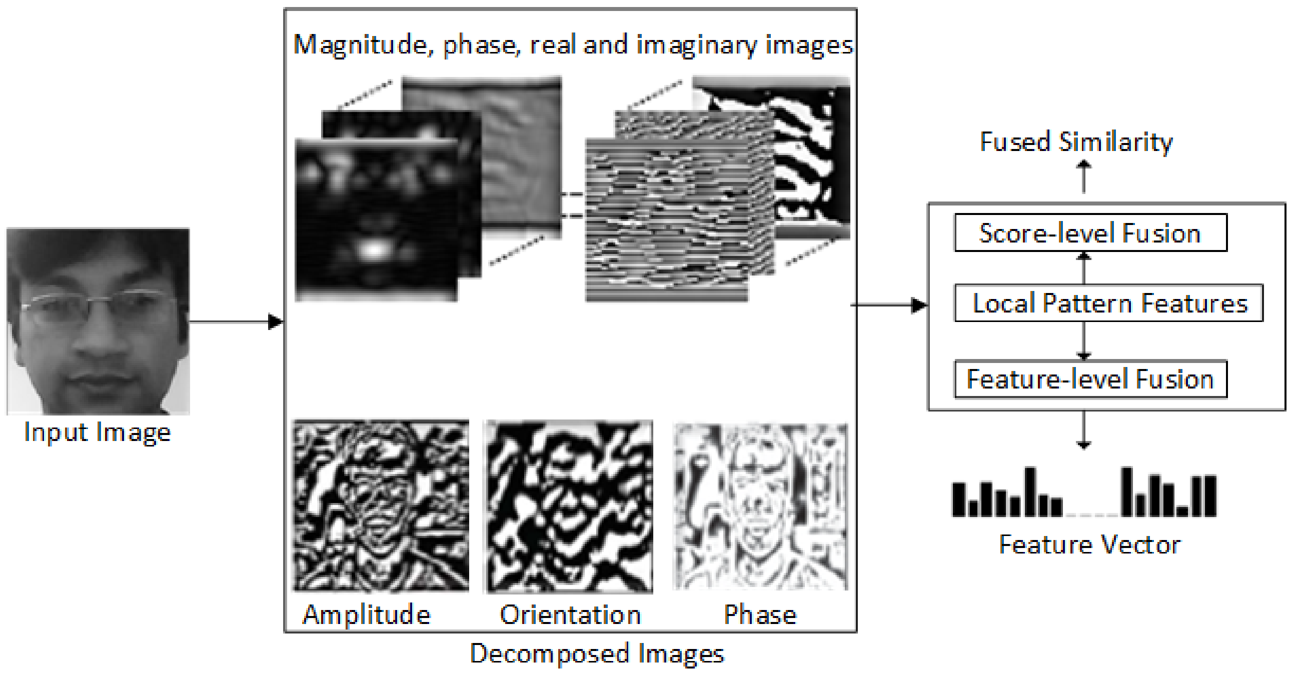


Figure 2.14: Feature Extraction in Spatial-Frequency Domain

patches and each patch is further sub-divided into sub-regions. Gabor wavelets are employed on each sub-region which in turn generates a feature vector for each sub-region. The classification is performed by concatenating the feature vectors from all the sub-regions with the help of a sum rule.

Gabor Phase Pattern based histograms (HGPP) were proposed by Zhang et al. [102] in which quadrant bit coding scheme was adopted to assign quadrant values to each phase value. After assigning all the values, local XOR patterns similar to local binary patterns were used to perform the feature extraction. These fetaures were extracted both the real parts as well as the imaginary parts of the image which resulted in high accuracy rate. A feature fusion strategy based on the local binary patterns and Gabor wavelets was proposed by Tan et al. [104].

2.6.2 Color and Wavelet Transform based Methods

A number of authors have used color information to develop blur and illumination robust descriptors for 2d and 3d object recognition. Two such robust descriptors have been proposed by Joost et al. [105] and Drost et al. [?]. They used ratios of image derivatives to develop a descriptor for color constant ratios that is invariant to blur and illuminant color.

A similar kind of approach based on color information has been proposed by Ballard et al. [106]. They make use of color histograms to perform recognition of objects. Another approach based on color histograms is proposed by Funt et al. [107]. They used color constant derivatives to represent an object for recognition. However, these approaches do not perform well against variations in illuminant color especially with a change in the camera viewpoint or object orientation and are also dependent on lightening geometry.

Lai et al. [108] proposed the use of Fourier transform and Wavelet transform to extract features from face in order to perform facial recognition. They have named their method Spectroface as it extracts features from two domains. They make use of wavelet transform to overcome the effects of facial expressions present in their face database. The wavelet transform eliminates the expression variance but also degrades the image resolution. After wavelet transform, Fourier transform is applied on the low frequency components of the images. Face representations are divided into two categories: The first order spectroface which extracts the features that are expression and translation invariant and second order spectroface, which extracts the features that are rotation and scale invariant and provide good accuracy rate.

The fusion of features from these two approaches produce high dimensional feature vectors. Principal Component Analysis is employed to reduce the dimension of the features before feeding them to the classifier for classification. Gabor surface features [109] also makes use of Local Binary Patterns on the amplitude property of Gabor wavelets. This is employed on the first and second order derivatives of the Gabor wavelets. In order to reduce the dimensionality of Gabor surface features EPFDA is used. The classification is performed by using cosine similarity measure.

2.6.3 Moment Invariants based Methods

The moment invariants have also been used in the past as blur invariant features. Flusser et al. [110] pioneered the use of moment invariants developed on top of geometric moments. They

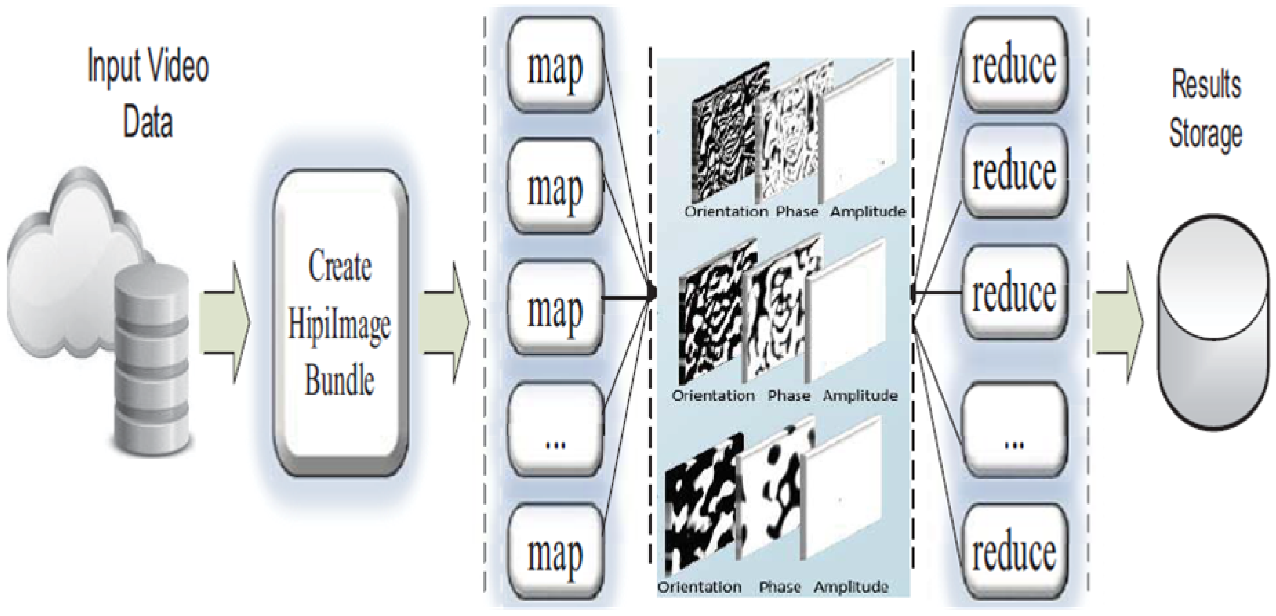


Figure 2.15: Architecture of Distributed Cluster

also used central moments to provide invariance to translation. Moment invariants have also been the part of various applications such as template matching [111], recognition of defocused objects and X-ray imaging.

Complex moments are also proposed by Cogdell et al. [112] for blur, rotation and scale invariance. Despite their wide usage in various applications, moment invariants remained sensitive to noise and background clutter. Invariants based on the phase of frequency spectrum obtained by Fourier transform are investigated by [18]. These invariants are also insensitive to the shift of the image. Ville et al. [113] proposed a centrally symmetric blur invariant descriptor based on phase-only spectrum of an image. The phase-only spectrum was normalized so it became insensitive to linear brightness changes as well.

A similar kind of approach is adopted by Ville et al. [19] in which the phase information was calculated within a local window for every image position. The quantization of the phase of discrete Fourier transform and de-correlation of low frequency components is performed in an eight dimensional subspace. A histogram of the resulting features is used for classification of blurred texture images. However, these invariants are limited to image shifts. Also, invariance to translation has not been considered in phase based frequency spectrum invariants.

Empirical Mode Decomposition (EMD) [114], has been used in the past to perform recognition and classification. Ehsan et al. [115] proposed an image fusion and enhancement technique using EMD to decompose non-stationary signals into IMFs. Linderhed et al. [116] proposed image empirical mode decomposition (IEMD) to locally separate superposed spatial frequencies from the image.

Liu et al. [117] presented 2DEMD to extract the local features of the two-dimensional Intrinsic Mode Function for edge detection. Yaseen et al. [118] pioneered the use of EMD on video streams in a cloud based parallel system as shown in Figure 2.15. They used first three IMFs and proposed a stack based hierarchy of features to perform object classification on a challenging dataset. However, all these works made the use of EMD with shallow networks and did not exploited the use of EMD for deep networks.

2.7 Object Detection and Classification in Clouds

Modern hardware resources and open source platforms for distributed stream and batch data processing have become an active research area to improve performance of video processing algorithms. For example, Apache Kafka [119], is a streaming platform which supports distributed processing of streams of records. It can store and process streams of records as they occur in a fault-tolerant way. The various APIs offered by Kafka can be used to publish and process stream of records effectively and efficiently. Similarly Apache Flink [120], is another open source platform for distributed stream processing. Flink has a streaming dataflow engine supported by various APIs to create fault tolerant streaming applications. These platforms can be executed as a cluster on one or more servers in cloud or edge infrastructures.

Cloud computing [121], offers shared computer processing resources to meet the computational demands of modern compute intensive algorithms. Cloud computing based analytics are widely being used today for large scale data processing. The results from existing studies suggest that such a model will be hugely beneficial for video processing and real time video analytics systems.

2.7.1 Object Detection and Tracking in Clouds

One of the key capabilities that should be implemented in the cloud for video analysis is the autonomous detection and tracking of targets across multiple camera feeds. A number of different approaches have been proposed in the literature for addressing the problem of multi-camera detection and tracking [122]. These techniques range from: feature matching, camera calibration and/or 3D environment modelling [123], to motion-trajectory alignment, etc.

In addition, some of these approaches have also been addressed in different application areas including surveillance [124] and sports etc. The problems of multi-target and multi-camera tracking can be treated as mutually exclusive and are often complicated due to their individually characterized challenges [125]. However, the joint problem is typical to most real-world scenarios such as in video surveillance. In this context, despite real-world constraints, detection and tracking requires dealing with occlusion; illumination variations, background clutter, calibration and data association. Recent research has indicated that most existing modelling techniques, such as motion flow models, appearance models and hybrid models, have all been extended from conventional multi-target tracking into multi-camera multi-target tracking with specific data association constraints.

2.7.2 Object Classification in Clouds

When object classification is performed on large scale data, it requires large computational resources. Leveraging cloud infrastructure using map-reduce framework as shown in Figure 2.16 to fulfil the needs of high computational resources has been a preference in both academic and industrial sectors. Researchers have also made use of in-memory cluster based on Spark to perform classifier training in parallel on multiple nodes, as shown in Figure 2.17.

An approach based on SIFT and Gabor descriptors was proposed by Matsuda et al. [126] to recognize food images. These features were clustered by K-means algorithm [127], to achieve an acceptable recognition rate. To achieve scalability and performance gains, cloud computing paradigm was utilized. It was concluded that for small amount of data, cloud computing

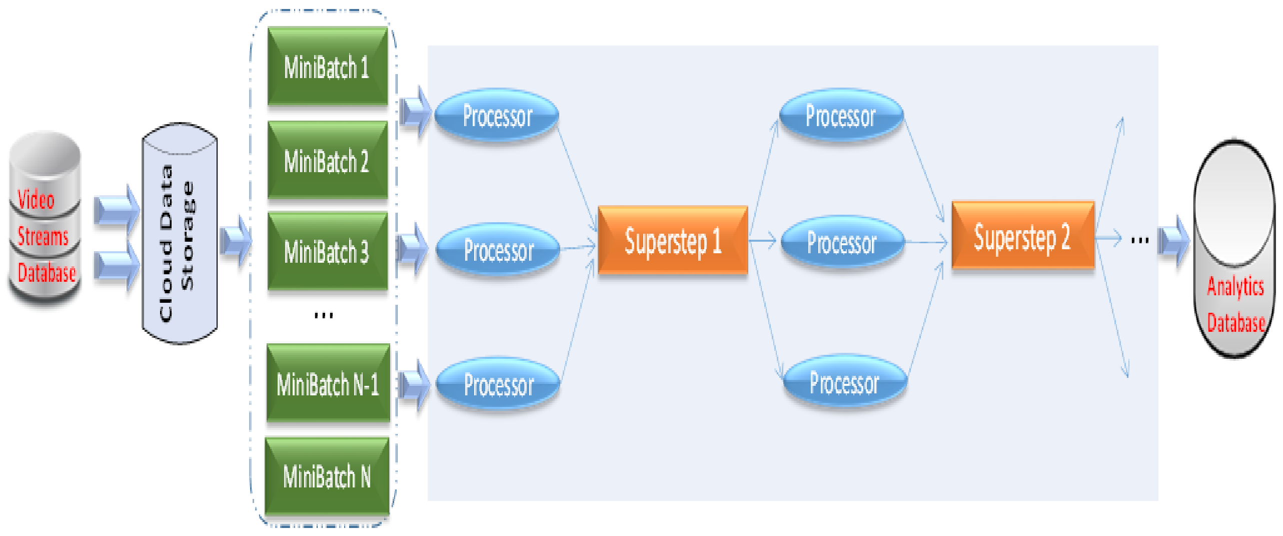


Figure 2.16: Video Analysis on the Cloud

performance was not very promising. This is because of the fact that cloud needs to prepare job runtime environment before a job is served. However, for large datasets cloud processing efficiency is far better because clouds are designed to support these kinds of workloads.

A cloud computing based object recognition system using two dimensional principal component analysis was implemented on a Hadoop based cloud system. However, this approach was less effective for object images with multi-illumination. The massively parallel cloud computing based approach [128] was also used to process astronomical images where a parallelised mapper without applying a parallelised reducer was used. However, there was no improvement in the image processing routines. Another cloud based system for analyzing large scale videos was developed using the MapReduce based clusters [129]. A cluster of six computers was created and video processing algorithms were ported to it. The execution time of algorithms was reduced as compared to a single system. Since the experiments were executed on a small dataset, scalability was not addressed. The improvement in accuracy was also not the focus of this work.

2.7.3 GPUs based Cloud Computing

The use of Graphical Processing Units (GPUs) as a high performance general purpose computing resource was commenced in 2009. The initial study was based on the implementation of

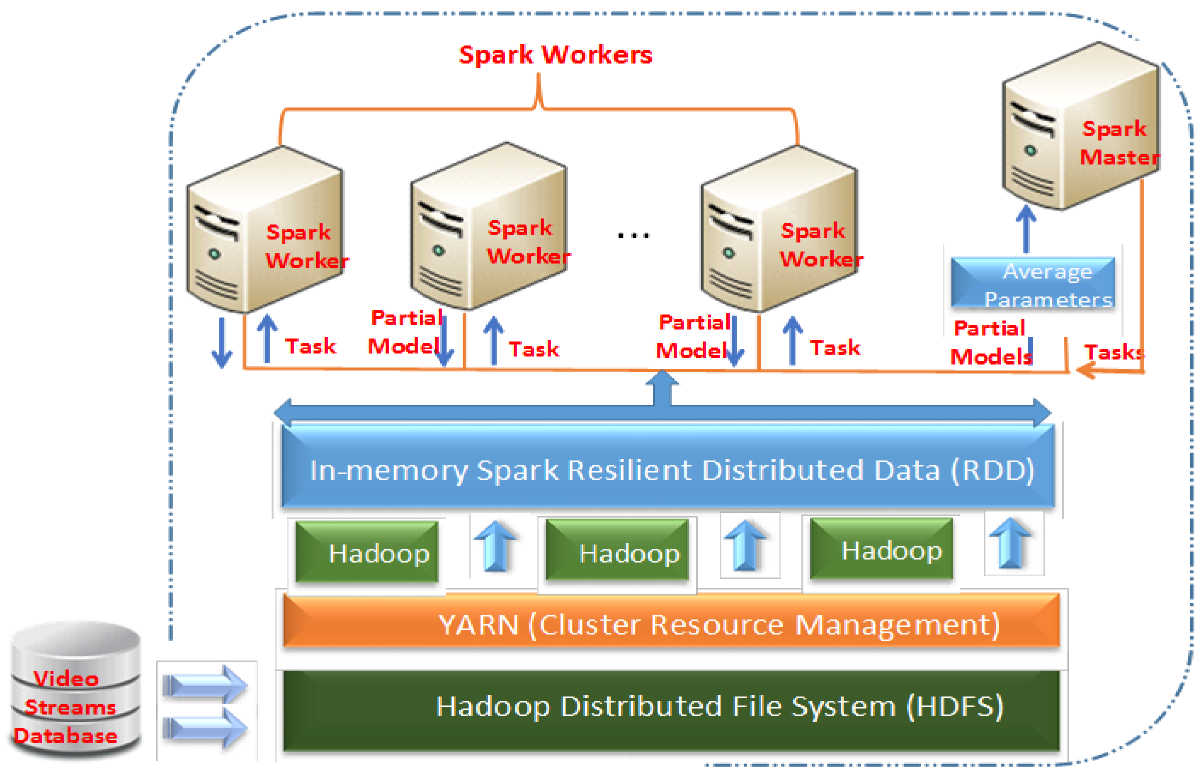


Figure 2.17: Deep Learning Model Training in Cloud

large scale unsupervised machine learning algorithms, Deep Belief Networks [130] and Sparse Coding [131] on GPU resources. These studies unfolded the power of GPUs to the research community by achieving a speedup of 5 to 15 times as compared to the CPU implementations. A parallelised object detection approach achieved a speed-up of 1.91 times on GPUs [132]. A parallelised motion estimation approach using a full search algorithm on GPUs achieved a speedup up of around 50 times than its CPU implementation [133].

Studies such as [134] proposed a GPU implementation of Haar Cascade Classifier algorithm and achieved a speedup of 13.8 times than a CPU implementation. Another Haar based face detection approach on GPUs [135] achieved 2.5 times speedup as compared to its CPU based implementation. These approaches provide a good overview of previously proposed video processing platforms and algorithms. However, most of the above cited approaches lack scalability and agility. These are also computationally expensive as the dimensionality of features is very high.

Furthermore, studies such as [136] and [137] have suggested the use of FPGAs as a hardware prototype to improve the performance of the deep learning based neural networks. FPGAs have proved to maintain less power consumption while improving the performance of deep network. [138] have proposed a scalable accelerator architecture which three pipelined FPGA units to improve the throughput of the deep network.

2.8 Summary

This chapter provided a survey of most of the state-of-the-art approaches that have been used for object detection and classification. The surveyed approaches are divided into three main categories and a taxonomy of these approaches is provided. The approaches related to each category are then explained with their strengths and weaknesses. A brief survey of the existing classifiers is also provided. We also considered the approaches which have been used on the cloud infrastructure in both time and spatial-frequency domains.

Admittedly, the state-of-the-art provides good performance in some situations but on the other hand they have their own limitations. Most of these approaches address challenges that are prone to still images and are expensive in terms of time and money. These approaches work in a serial fashion with a large number of features which are necessary for accurate classification but this slows down the process. Also, the training of the classifier is a time consuming task because they require a large number of training examples which must be collected and labelled manually. These training examples enable the system to capture variations in object appearances but also burden the training process.

Existing feature extraction and classification approaches often provide false detections and lack accuracy in complex situations. These approaches also show less resilience to changing illumination conditions, require huge computing resources and consume more time. Furthermore, these approaches are slow due to limited availability of computing resources and do not scale well. Additionally, in order to gain detailed insights, complex algorithms such as deep learning algorithms need to be employed which further give rise to several challenges in terms of computational power, data storage and hyper-parameter tuning of these networks.

Survey of the recent literature in this chapter helped to identify the potential research gaps and challenges. As compared to existing literature, we propose a video analytics system that addresses most of these challenges and performs object classification under controlled and uncontrolled environment requiring minimum human interaction. We have proposed a workflow which automates the object classification process. We have also made the use of spatial-frequency domain features which helped to improve the accuracy of the classification even under uncontrolled conditions. We have also proposed a mathematical model for the hyper-parameter tuning of the model which further improves the accuracy and performance of the classifier.

The following chapter will explain the proposed video analytics workflow to perform object classification. A set of components orchestrated in a workflow will be proposed to perform object classification. Each component is cascaded, resulting in an integrated system which attempts to minimize human intervention by executing each component of the workflow automatically.

Chapter 3

Video Analytics Workflow

3.1 Outline

Chapter 2 reviewed the state-of-the-art work in the context of video analytics. A number of approaches focusing on object detection and classification for video streams were reviewed in detail. The cloud-based frameworks used for the execution of detection and classification algorithms were also reviewed. Existing approaches relevant to the proposed work were critically analysed and their weaknesses and shortcomings were highlighted. This critical analysis of the existing approaches helped to highlight the research gaps where further research could be carried out.

In chapter 3, we provide an in-depth description of the proposed video analytics workflow. We propose a set of components orchestrated in a workflow to perform object classification. Each component is cascaded resulting in an integrated system which could serve as an object detection and classification pipeline for cloud-based video analytics. The proposed system attempts to minimize human intervention by executing each component of the workflow automatically, reduces computation time and enables the processing of large number of video streams.

3.2 Introduction

The workflow of the proposed video analytics system is designed to automate and speed-up the processing of large number of video streams. The video streams are automatically fetched from the cloud storage and processed intelligently for object classification. All the components of the workflow are executed automatically one after the other. There is no manual labour required for this work. Completion of one task automatically triggers the execution of next task which reduces human intervention. The following are the main components of the proposed workflow of our video analytics system, as illustrated in Figure 3.1.

Video Decoding: The video streams are by default in the encoded format with H.264 format having an fps of 25, data rate and bit rate of 421 kbps and 461 kbps respectively. These are decoded to generate individual video frames which contain side, front and rear views of objects. The total number of decoded video frames is directly proportional to the duration of video stream being analyzed. For a video stream of 120 seconds length, 3000 video frames are generated.

Object Detection: The proposed workflow of our video analytics systems is built upon object detection and classification. The object detection refers to the detection of all instances of an object belonging to a known category such as faces or cars. Usually, a video contains a number of objects. These objects can reside at any location within a frame. Exploring these locations of objects comes under the category of object detection. Object classification, on the other hand, refers to the identification of detected objects. Labels are assigned to the detected objects during this process. A video stream and some known labels are provided to the system. It then assigns the correct labels to the detected objects in a video stream.

Object Extraction: The object detection and extraction procedure is performed on all the frames one by one. The object is first detected in the frame. This provides the exact location of object in the frame. It is cropped and saved as a separate image, so that the classification step will have to process a small size image. The cropped or extracted object is then passed to the subsequent object classification component.

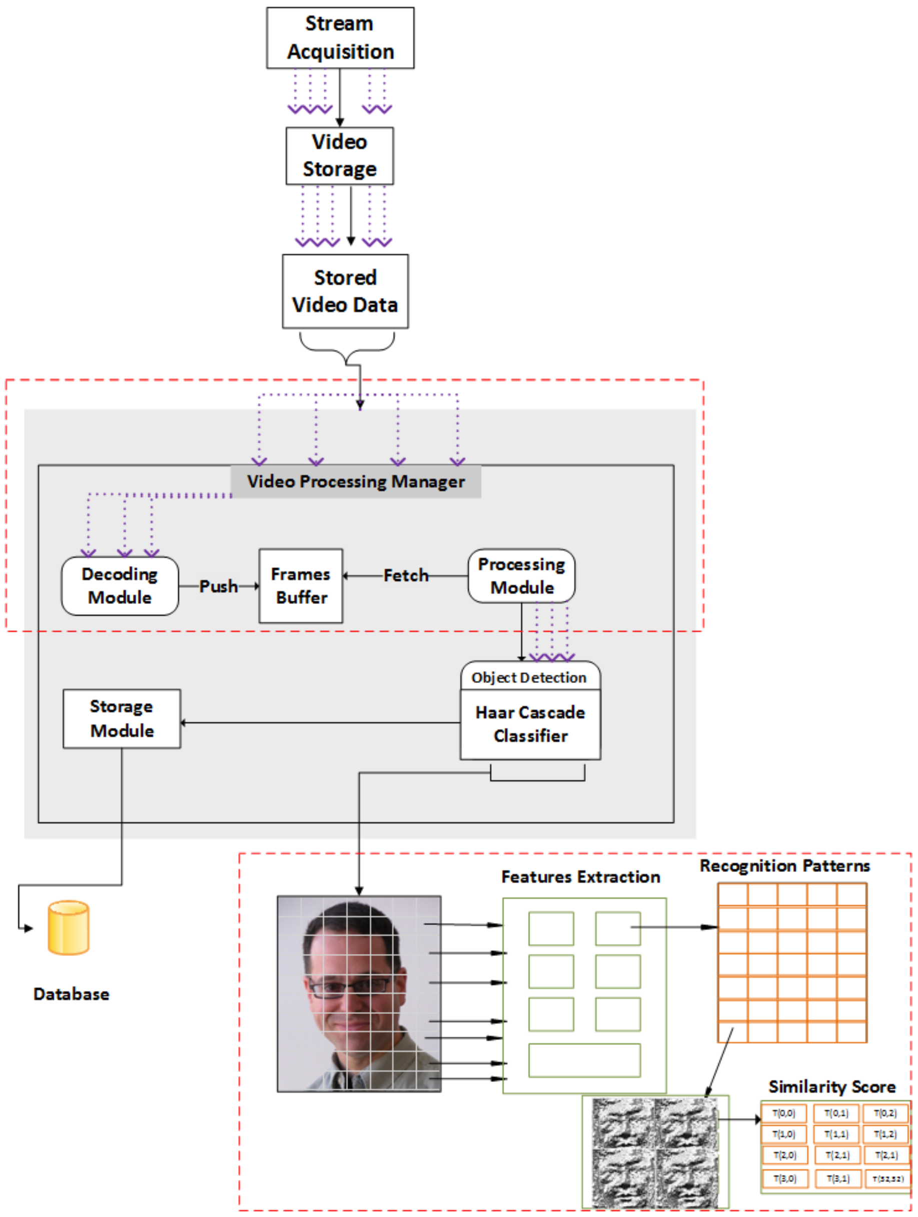


Figure 3.1: Video Analytics Workflow

Adaptive Frame Sampling: We have introduced the concept of adaptive frame sampling in which the frames which do not contain any object in them are discarded. This reduces the amount of video frames to be processed and only those frames are retained which contain objects in them. Also, the consecutive video frames in a video stream are more likely to contain the same object with the same pose and lightning conditions. Due to constraints on motion, no change can occur in such a short interval of time. For consecutive video frames we took 5 frames out of 25 per second as the fps is set to 25. So we have a frame sampling rate of 5. The frame sampling helps in discarding redundant frames and leads to less data processing time.

Features Extraction: The extracted objects from the video streams are then processed via the features extraction component, which generates local pattern features of all the extracted objects. These local patterns serve as features which can be used for the classification of an object. These features represent the extracted objects in such a way that they become highly discriminative to various grey-level changes in the objects. We have proposed the optimization of both shallow and deep learning features at this stage. The details of the deep learning based features will be described in chapter 4 of this thesis.

Object Classification: The classification component first analyses the marked input object which is to be classified. It extracts and stores features from it. This marked object is then compared with all the other objects present in the video frames. If the same object is identified in any other frame, its instance is updated and its corresponding time and location is saved. If the comparison fails, then it means that the marked object is not present in the video stream which is currently being processed.

The proposed video analytics workflow based on cascaded detection and classification is an efficient way to automate the video analysis process and to minimize human intervention. The object detection provides a reference for the location of the object which can be tracked in the subsequent frames.

It is cropped and saved as a separate image, so that the classification step has to process a smaller sized image. The object is then passed on to the subsequent object classification phase. We have proposed an object matching algorithm in which the target object is compared with the candidate objects for classification. Each component of the workflow is described in more detail in the next section of this chapter.

3.3 Workflow Components

This section provides a detailed description of the components of the proposed system. The operations employed to process video streams to support object detection and classification are

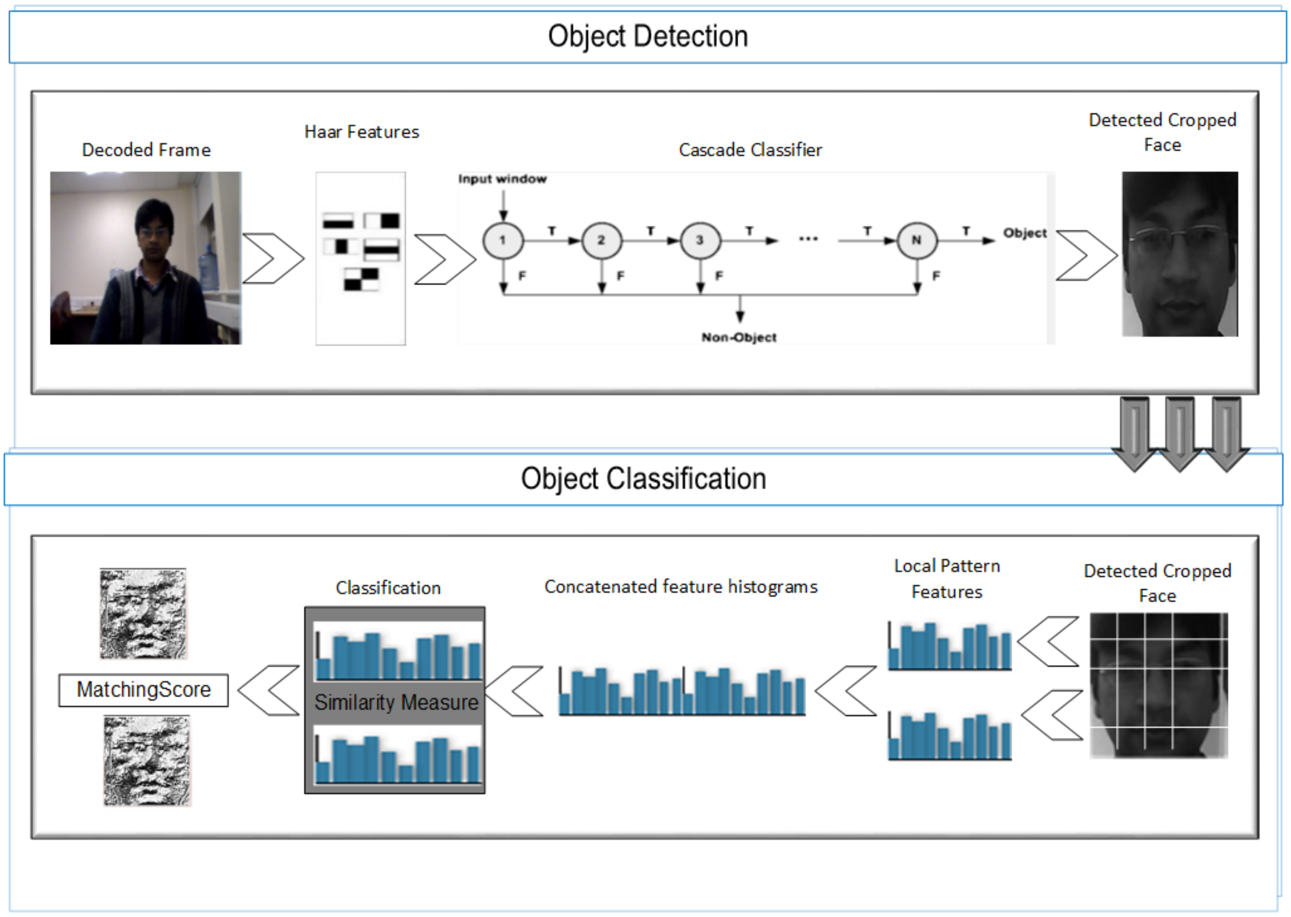


Figure 3.2: Relationship between Object Detection and Classification

also described. The proposed system provides scalable and automated classification of objects in a large number of video streams.

The proposed video analytics workflow applies multiple algorithms for detection and classification. Figure 3.2 shows the relationship between detection and classification algorithms. The algorithms are employed in a workflow that the results produced by one algorithm are processed further by the following algorithm. The objects of interest are extracted from the video frames by detection and are then cropped around the area of detection.

The rest of the frame which contains unwanted information is discarded to save processing time and resources. This algorithm independently operates on all the frames in a sequence. This results in the extraction of all the desired objects from all the video frames. The local patterns of each extracted object are then generated and stored in the associated buffer. Object matching is then performed on the generated local features. The generated results are then stored in the database. In the next subsections we provide more detail on each component of the workflow.

3.3.1 Video Decoding

Video decoding is the first component of the proposed video analytics workflow. The video streams are first fetched from the physical storage and then decoded to extract individual video frames. The recorded video streams are encoded with H.264 encoder [139] to save storage space. Each video stream is usually recorded at 25 frames per second with a data rate and bit rate of 421 kbps and 461 kbps respectively.

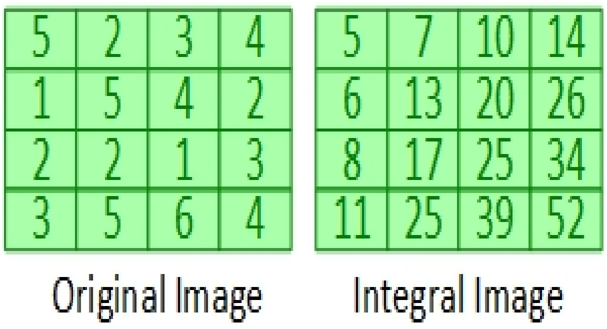
We have used FFmpeg library to decode the video streams. It involves fetching a video stream and extracting individual frames from it. However, it is an I/O bound process and reading and writing large number of video frames can be a time consuming process. There are almost 1800 video frames for a video stream of one minute length.

The number of decoded video frames is also dependent upon the length of video streams being analysed. These frames are then transferred to the processing component for processing. Each frame is processed individually for object detection and recognition. The machine learning algorithms are applied on each frame for detection and recognition purposes.

3.3.2 Object Detection

The decoded video frames are then transferred to the object detection component. The object detection component processes all the frames one by one. The detection of moving object from the video frame provides the exact location of the object in the video frame. The detection process is performed on individual video frames and have no dependence on other video frames. Hence, each frame is processed independent of one another.

We have used the already trained frontal face Haar Cascade Classifier [20] for the detection of human faces from video streams as it does not require to be trained separately. The already



5	2	3	4
1	5	4	2
2	2	1	3
3	5	6	4

Original Image

5	7	10	14
6	13	20	26
8	17	25	34
11	25	39	52

Integral Image

Figure 3.3: Original and integral image

trained classifier allows to save the computation cost of training. The computational cost of the detector is highly dependent on the number of features being evaluated. The small number of features means low computational cost but the classifier will also be less accurate.

A classifier with more features results in a higher classifier accuracy. It was noted during the experiments that a frontal face classifier built on 25 feature stages provides a detection rate of 95 percent. The computation time depends on the resolution of the video frame. So there is a trade-off between the computation cost and accuracy of the classifier. The next subsection provides brief but concise description of Haar Cascade Classifier.

3.3.2.1 Haar Cascade Classifier

The implementation of the Haar Cascade Detection as shown in Algorithm 1 is based on the extraction of haar features from the image. Haar features are computed as single values gathered by computing the sum of pixels under the white rectangle region subtracted by the sum of pixels under the black rectangle region. These regions are adjacent to each other and possess same shape and size.

Integral Image: The implementation of the integral images allows the features to be computed very rapidly. The integral image can be computed by adding all the pixels above and to the left of a specific location. Then by using four array references, the integral image is calculated. Figure 3.3 shows an original image and its equivalent integral image.

Adaboost: Since the number of features generated by Haar is quite large, it becomes necessary

Algorithm 1 Haar Cascade Classifier

```
1: procedure ComputeHaar
2:     for $i \leftarrow 1$ to number of scales in pyramid of images do
3:         Downsample image to create image(i)
4:         Compute integral image, image(ii)
5:         for $j \leftarrow 1$ to number of shift steps of sub-window do
6:             for $k \leftarrow 1$ to number of stages in cascade classifier do
7:                 for $l \leftarrow 1$ to number of filters of stage $k$ do
8:                     Filter detection sub-window
9:                     Accumulate filter outputs
10:                if accumulation per stage fails then
11:                    Reject sub-window as face
12:                    Break K loop
13:            if sub-window passes all stages checks then
14:                Accept sub-window as face
```

to adapt a feature selection approach to reduce the dimension of features. This dimension reduction can be best achieved by Adaboost [140] which is a learning algorithm and works in a cyclic process. It starts by keeping a set of weights that are distributed uniformly over every training example. Then it selects one feature in its first cycle which gives best recognition performance and defines a weak classifier against it. The subsequent cycles assign higher weights to the training examples which were misclassified by the first weak classifier. This enables the newly chosen feature to concentrate more on misclassified examples. This process continues and ultimately ends upon a strong classifier which is actually a linear combination of all the weak classifiers selected during each cycle.

To further improve the algorithm, it is cascaded. As most of the image area is non-face region it groups the features into different stages of classifiers. The region that passes all stages of the cascaded classifier is an object. When an object is detected the area of the object can be used by the object recognition algorithms to match the object for identification.

Image Pyramid: In order to make the classifier scale-invariant, the frame pyramid approach [141] has been used. The pyramid represents the same frame in multiple scales and enables the detector to be scale invariant. Objects with varying image sizes can easily be detected through the pyramid approach. An object pyramid can be constructed by using the down-sampling approach which down-samples the frame by one scale in each iteration. Integral image for each

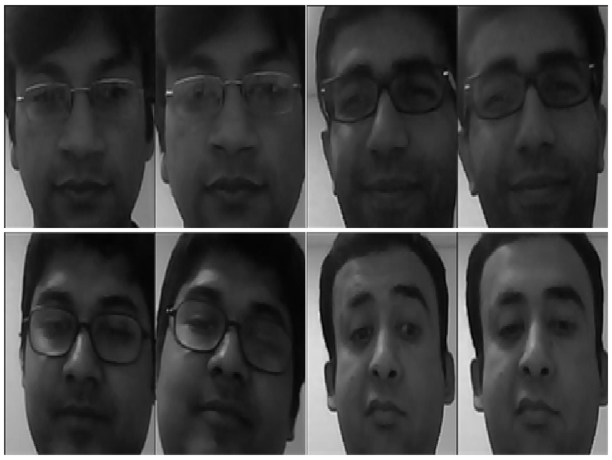


Figure 3.4: Extracted faces from video streams

scale in the pyramid is then calculated to speed up the process of pixels sum. Integral image [142] helps to compute the summation of pixels present in a rectangular region by utilizing only four pixel corners. This approach of using integral images is highly efficient especially for the cases in which pixels sum of many rectangular regions of same image are needed to be computed. Since the detector uses the sliding window approach and pixel sum for each shifted window is required, this approach highly reduces the complexity.

The sliding window slides pixel by pixel on the whole frame in search of a face. The area under the sliding window is passed to the cascaded classifier. Since most of the image area is a non-face region, the features are grouped into different stages of classifiers. The region which passes all stages of the cascaded classifier is a face. The area under the sliding window is required to be passed through all stages of the cascade classifier. If at any stage, the input region is unable to pass the stage by not meeting the required threshold, it is immediately rejected. If the region passes all the stages successfully, then it is considered to be an object.

3.3.3 Object Extraction

The detected objects from the video frames are then cropped around the area of detection to extract detected objects as depicted in Algorithm 2. This is an automated process and does not require any manual labour. The detection of desired object from the whole video frame and its extraction through cropping is an important step for the proposed video analytics workflow.

Algorithm 2 Object Extraction

```
1: procedure ObjectExtraction
2:     for $i \leftarrow 1$ to number of frames in the video stream do
3:         [xmin ymin width height] = ComputeRectangle(i)
4:         Coordinates = [xmin ymin width height]
5:         CroppedObject = Crop(image(i), Coordinates)
```

This helps to narrow down the frame processing area for object classification by eradicating those areas from frames which do not contain objects. Figure 3.4 shows some of the extracted faces from the video streams.

The extraction of desired objects from the video frames through detection helps to improve the performance of the system in two ways: 1) since the frame area is reduced, the analysis algorithm has to now process a smaller sized frame as compared to original one. This reduces the processing time of individual frames and in turn reduces the overall processing time of the whole video. Also, it is much easier and appropriate to scale and normalize the small scale objects and apply transformations (flip, rotate, skew) on it. 2) As the frame has been narrowed down to only object(s) of interest, by removing the unwanted area of the frame, it now contains only the desired object. The illumination effects and noise which have the possibility to be present in the unwanted area will not reflect in the object recognition process.

3.3.4 Adaptive Frame Sampling

The detection of desired objects from video streams helped to perform adaptive frame sampling in our proposed video analytics workflow. The word sampling is used here in terms of down sampling the number of frames. The idea is to discard the frames which do not contain any object in them. In this way we reduce the amount of video frames to be processed and retain only those frames which contain objects in them.

In a video stream, consecutive video frames are more likely to contain the same object with same pose and lightning conditions. No change can occur in such a short interval of time. Or, if there is no object, the consecutive frames will also have no object in them. The frame filtering/sampling by detection helped to discard the redundant frames as shown in Algorithm

Algorithm 3 Adaptive Frame Sampling

```
1: procedure SAMPLE FRAME
2:     Initialize FrameCounter = 0
3:     Initialize FrameConditionCounter = 0
4:     for VideoStream ! = NULL do
5:         if $FrameCounter! = 0$ then
6:             FrameConditionCounter = 0;
7:         if $FrameCounter mod 25 = 0$ then
8:             FrameConditionCounter = 0;
9:         FrameCounter++;
10:        FrameConditionCounter++;
```

3 and leads to less data transfer which in turn reduces the training time.

For consecutive video frames, we took 5 frames out of 25 per second as the fps is set to 25. Therefore, we have a frame sampling rate of 5. The frame sampling helps in discarding redundant frames and leads to less data processing time. In the next section, we describe the feature extraction component of the proposed workflow.

3.4 Feature Extraction

The feature extraction component generates features for all the extracted objects. These features are used for classification of an object. These features represent the extracted objects in a way that they become highly discriminative and invariant to various gray-level changes in the objects. The proposed video analytics workflow fundamentally consists of local binary patterns and local ternary patterns. This component is further extended with the optimization of deep learning based features which are explained in chapter 4.

3.4.1 Local Binary Patterns

We have used LBPH algorithm [143] for the generation of local pattern features. The algorithm makes use of the local binary patterns in order to generate feature vectors. LBP features, as shown in Algorithm 4, are computed by dividing the examined window into cells. Each

Algorithm 4 Compute LBP on CPU

```
 1: procedure ComputeLBP
 2:     $FrameHeight \leftarrow$ number of $rows$
 3:     $FrameWidth \leftarrow$ number of $columns$
 4:     for i++;i<FrameHeight do
 5:         for j++;j<FrameWidth do
 6:             $CentrePixel \leftarrow frameData[i][j]$
 7:             $n \leftarrow TotalNeighbourPixels$
 8:             for $npixel \leftarrow 1$ to $n$ do
 9:                 $npixel \leftarrow NeighbourPixels$
10:                 if $\mathrm{npixel}_a > \mathrm{CentrePixel}$ then
11:                     $npixel \leftarrow 1$
12:                 else
13:                     $npixel \leftarrow 0$
14:                 $FrameData[i][j] \leftarrow \sum_{i=1}^{n} NeighbourPixels[i][j] * 2^n$
15:     Replace $frameData$
```

cell contains a sub-block of 3x3 pixels. Then, each pixel in the sub-block is compared to its neighbouring pixels. If the value of centre pixel is greater than its neighbour pixel, 1 is stored at the location of that pixel. If the values of centre pixel is less than the neighbouring pixel, then the gray value of that pixel is replaced with 0.

Binary Patterns: This procedure makes the sub-block a binary block containing 0 and 1 depending upon its pixel values. This is known as the labeling of pixels. These labelled pixels generate a binary pattern which is converted into a decimal value. The gray value of centre pixel is then replaced with the decimal value. This procedure is repeated on the whole image and an LBP image is obtained. Then, the histogram is calculated and normalized over the frequency of each number occurrence. These normalized histograms give a feature vector of the window.

In order to perform face recognition, the face image is divided into multiple blocks or regions. For each block or region, the LBP histogram is computed as explained above. The feature vector of the whole image is a combination of all the LBP histograms of all regions in an image. Figure 3.5 shows the original faces and the LBP computed faces from video streams.

Uniform Patterns: We have used the extended version of the local binary pattern operator which makes use of uniform patterns. These uniform patterns help to decrease the size of

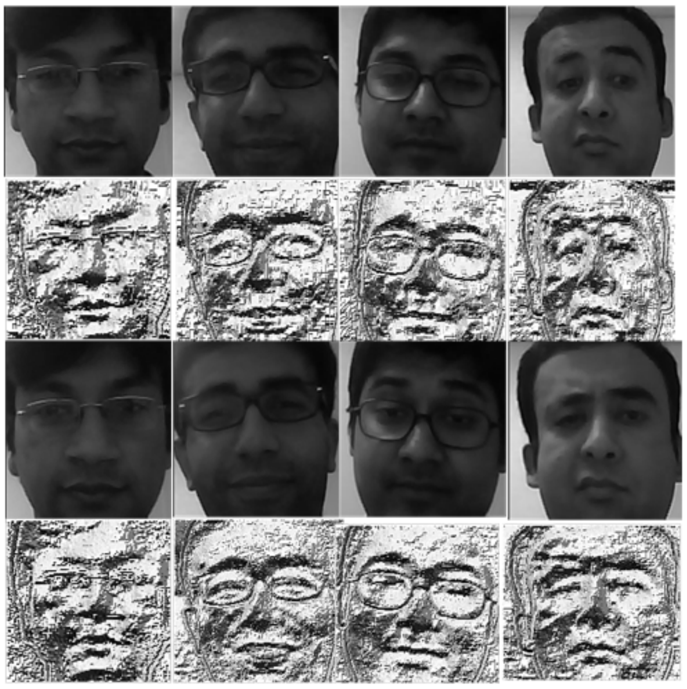


Figure 3.5: Original and LBP faces

the feature vector. Since we are calculating the local binary patterns of a large dataset, the use of uniform patterns helps to lower the computation cost. Uniform patterns work on the phenomenon that some patterns occur more frequently than other patterns.

A pattern is said to be uniform if there are a maximum of two bit-wise transitions from 1 to 0 or vice versa. The patterns 01110000 and 11001111 have two transitions and are, thus, uniform. These uniform patterns are used during the computation of LBP labels with a separate label for each uniform pattern. The rest of the non-uniform patterns are labelled with a single label.

Local Ternary Patterns: Local Ternary Pattern feature [144] is an extension of LBP. In LTP, the pixel difference between central pixel and neighbouring pixels is encoded into a ternary code. This ternary code is further split into positive LBP and negative LBP to reduce the dimensionality. This encoding of pixel difference into a separate state makes it more robust to noise. A pattern histogram, like in LBP, is then created by using the ternary values of neighbouring pixels. LTP features are more robust to noise than LBP but loss of information occurs when splitting into positive and negative LBPs. Also, redundant information resides in the histograms of both LBPs as they are strongly correlated. The next section explains the object classification process from the extracted features.

3.5 Object Classification

The video analytics systems consist of a large set of features. This large feature set mostly contains redundant information. This redundant information does not play any role in increasing the accuracy of the system, but, on the other hand, it increases the processing time. A large amount of memory is also wasted by these redundant features.

Dimensionality reduction is usually employed after feature extraction which plays an important role in reducing the processing time and memory usage of any video analytics system. Another reason of using the dimension reduction method is to remove correlation between features. We divide this section into two subsections. In the first subsection we describe the dimension reduction approaches used in this thesis and the second subsection describes the object classification process.

3.5.1 Dimension Reduction

Dimension reduction approaches help to select features which are highly uncorrelated. The reduction in dimensionality should be carried out in a proper mathematical way to avoid significant loss of useful information. This removal of redundant information is normally termed as dimension reduction. The goal of dimension reduction methods is to minimize the density of data and to bring data in a representation which is more understandable while retaining the same information. After reducing the features dimensions, they should be used for classification. An illustration of dimension reduction can be seen in Figure 3.6.

Broadly, dimension reduction methods can be divided into two categories. First category contains methods that project the high dimensional features onto a low dimensional space. The basis of these methods are techniques from Linear Algebra. However, the projected metrices obtained from these methods are quite dense as they do not suppress the features which are unnecessary or irrelevant. Infact, it assigns them lower weights depending upon the contribution to the final decision. This makes the resultant feature-set quite dense as compared to the second category which is relevant feature selection. Most commonly used discriminant subspace

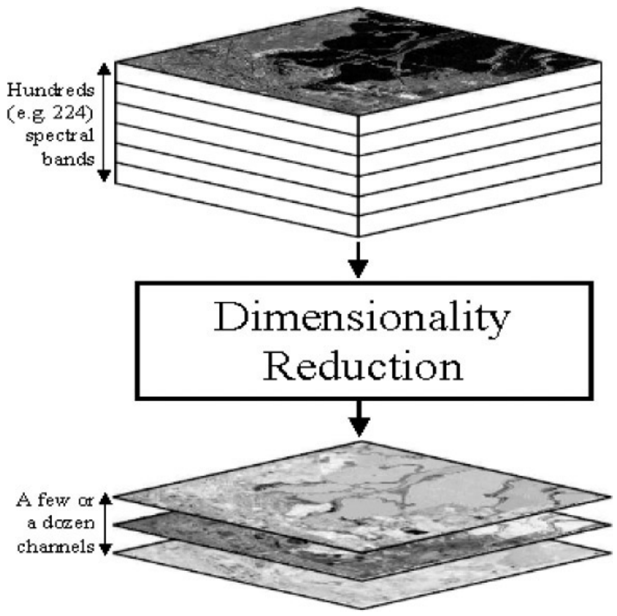


Figure 3.6: Dimension Reduction

projection methods include Principal Component Analysis (PCA) [27], Partial Least Squares (PLS) [145] and discriminative topic models [146]. Algorithm 5 shows the procedure of Principle Component Analysis.

In the relevant feature selection category, the objective is to reduce the set of features by getting rid of unnecessary or irrelevant features without compromising on accuracy. Discarding irrelevant or unnecessary features helps to achieve speedup in the classification process while maintaining accuracy. This is because noise is removed and emphasis can be made on the parts that are helpful for recognition. Most commonly used feature selection methods include L1 regularization [147], greedy selection based on boosting [148] and weight truncation [149].

Method Selection: Both dimension reduction and feature selection methods can be used to enhance the performance. Both the methods have their own advantages. The feature selection method has the advantage that it is simple and it can easily be interpreted. In this method, weights are usually assigned to the features based upon their contribution. All these weights are assigned to the features individually. Depending upon these weights, good features can be selected and the remaining features with lower weights can be discarded. This process ends up with a small number of features as compared to the original ones along with high weights.

However, this method has its own disadvantages. During the feature selection process, the

Algorithm 5 Principle Component Analysis

```
1: procedure PCA
2:     $O \leftarrow$ Obtain the feature matrix
3:     Compute mean and subtract the mean from each column of $O$
4:     $CV \leftarrow$ Compute the covariance matrix $Cov(x)$
5:     Compute Eigen Vectors $(u_1, u_2, \dots u_n)$
6:     Compute Eigen Values $(k_1, k_2, \dots k_n)$
7:     Validate eigen vectors and eigen values
8:     $PC's \leftarrow$ Eigen vectors with largest eigen values
```

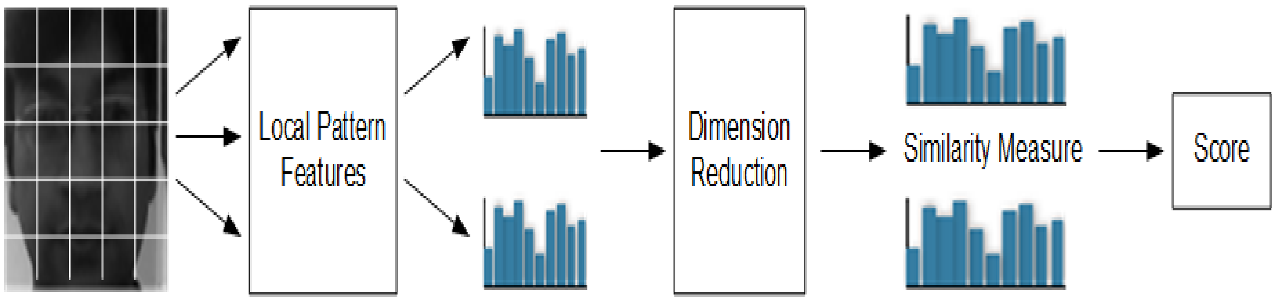


Figure 3.7: Object Matching

correlation between the features is not taken into account. This correlation between the features is of great interest for many classification algorithms. Another disadvantage of this method is the inability of selection of relevant features. The weights assigned to features which are highly correlated to each other are almost the same which leads to the fact that most relevant features are left off.

The dimension reduction on the other hand, does not have such drawbacks and it also provides a way to escape from the problem of curse of dimensionality [150]. By projecting the high dimensional data onto the low dimensional space, statistical relationships between the features can easily be studied. However, interpretability is always an issue in dimension reduction. As the methods in both the categories complement each other in advantages and disadvantages, we have employed the methods from both the categories to yield a better overall performance. Correlation between the features can successfully be figured out by dimension reduction but it fails to remove irrelevant features. The feature selection on the other hand fails to find correlation between features but performs well when the features are fully explanatory. So using both feature selection methods and dimension reductions according to the requirements of the video analytics system leads to good performance results.

Algorithm 6 Object Matching

```
1: procedure COMPUTE SIMILARITY SCORE
2:     Compute LBP Histogram of marked object
3:     Compute LBP Histograms of Objects in Video Streams
4:     for all objects in video streams do
5:         Compute Histogram intersection of Marked Object with Objects in Video Streams
6:         if $IntersectionResult > 0.9$ then
7:             Object Found
8:         else
9:             Object Not Found
```

3.5.1.1 Object Matching

In order to perform correct classification, a good feature description as well as a good similarity metric is required. State-of-the-art techniques can learn a dataset-specific metric. By learning a dataset-specific metric, the system learns the underlying regularities of data which helps to perform a good classification and improves performance. Different distance-based similarity measures have been used in the literature. Histogram intersection and Euclidean distance are the most commonly used distance measures. Many approaches also used Pearson Correlation Coefficient, Chi-squared and L2 measures as distance measures.

In the proposed video analytics workflow, an object matching algorithm, as shown in Algorithm 6, is applied on the local patterns of detected objects for classification. Object matching is performed by comparing the detected object features with the features of a marked object. This comparison of marked object is made with all the objects. Each comparison generates a similarity score of the marked object and the detected object.

The histogram intersection distance measure has been used to generate the similarity measure, as shown in Figure 3.7. Depending upon this similarity measure of two object features, classification decision is made. The proposed object matching algorithm has proved to be generic and is not adapted to any dataset. It is capable of identifying objects from the video streams without requiring any complex similarity metric-learning algorithm, any other supervised learning model or any outside data from other sources. The histogram intersection can be calculated as:

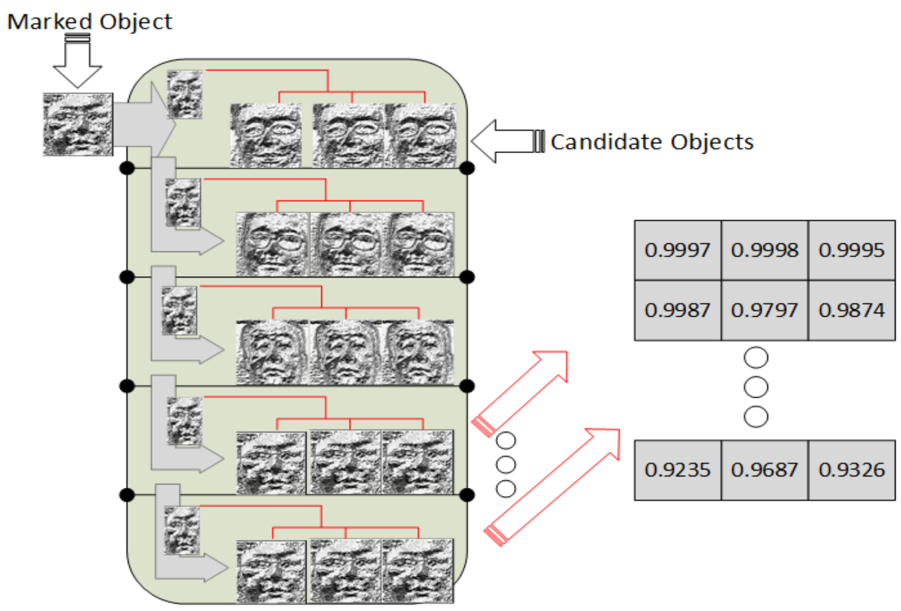


Figure 3.8: Visualization of matching process

$$“D(S, M) = \sum_{b-1}^{B} min(S_b, M_b)” \tag{3.1}$$

where “S” and “M” are a pair of histograms of two video frames containing “B” bins.

3.5.1.2 Matching Scores

Each comparison generates a score of each individual registered in a database, as shown in Figure 3.8. These scores obtained after performing the histogram intersection determine the recognition of a marked person which was being searched in the video streams. We have used a threshold of 90 percent match in our experiments. We obtained over 90 percent accuracy rate in case of matching individual objects. The matching scores for unmatched individuals is 70 percent or below. The matching scores along with locations and time of presence are stored in the database.

The performance of any object classification system can be affected by the facial structure constraints (gender, ethnicity) and the viewing parameters such as illumination and viewpoint. In addition, a number of perceptual complications can occur due to the movement of objects in video streams. The facial movements of a person can be classified as rigid or non-rigid. The rigid movements include tilting, nodding or shaking around the vertical axis. These movements can change the angle of a face from a static point. On the other hand, non-rigid movements take place due to facial expressions and eye-gaze during speech. These movements can distort the identifying features of the face. A smiling facial expression can strongly differ from a surprised facial expression.

This difference occurs due to the relative change in position of the eyebrows with respect to nose, mouth or other features. Converting an image from low resolution to high resolution by applying a number of pre-processing steps including normalization, histogram equalization and object scaling tackles these issues. It was observed during the experiments that because of the discriminative power of the proposed system, it is capable to perform well at low level of perceptual complications. The proposed system has shown performance for various rigid and non-rigid movements by providing high accuracy rates.

3.6 Summary

In this chapter, we presented a video analytics workflow for object detection and classification. Different stages of the proposed workflow are explained in detail and the connection between each stage is described. We also explained the dimension reduction approaches used in this work and the matching process which is used for classification. The proposed system requires minimum human interaction and provides automated object classification from large number of video streams. The system is also capable of coping with the challenges of increased volume of data.

In the following chapter of this thesis, we present cloud-based blur and illumination invariant approach for object classification from image and video data. We explain the use of Bi-dimensional Empirical Mode Decomposition (BEMD) for feature extraction and provide the details of our feature fusion strategy. We discuss the theoretical and implementation details of the proposed approach. We also provide the details of the optimization strategies and techniques for our deep learning based video analytics system in the next chapter.

Chapter 4

Feature Fusion based Video Analytics

4.1 Outline

Chapter 3 discussed a video analytics workflow for efficient processing of large number of video streams. The proposed workflow reduced human intervention by automatically executing cascaded algorithms and enabled the processing of large number of video streams. Different components of the proposed workflow were explained in detail and the connection between each component was described.

In chapter 4, a cloud-based blur and illumination invariant approach for object classification from image and video data is presented. The Bi-dimensional Empirical Mode Decomposition (BEMD) has been adopted to decompose a video frame into Intrinsic Mode Functions (IMFs). These IMFs further undergo first order Reisz transform to generate monogenic video frames. The analysis of each IMF has been carried out by observing its local properties (amplitude, phase and orientation) generated from each monogenic component. A feature fusion strategy is then adopted to perform classification.

We further propose an optimal tuning of the deep learning model to classify objects from video streams. The tuning of the hyper-parameters is optimized through a mathematical model for efficient analysis of video streams. The system is capable of enhancing its own training data by performing transformations including rotation, flip and skew on the input dataset making it more robust and self-adaptive. The use of distributed training mechanism rapidly incorporates large number of distinguishing features from the training dataset - enabling the system to perform object classification with least human assistance and external support.

4.2 Introduction

As mentioned in the problem description in Chapter 1, the video analytics systems have to process data that contain a number of challenges including illumination and blur. These challenges mostly occur due to motion, out of focus, or atmospheric turbulence. Also, since these systems operate under uncontrolled lighting conditions, they are also prone to illumination effects. Rotation angle of the objects being monitored also poses many challenges.

The accuracy of any object classification system is highly dependent on how these challenges are overcome. A good counter-measure to these challenges can lead to more accurate results. However, video de-blurring and de-illumination are resource and time consuming tasks and often bring in new artifacts [18]. It is, therefore, desirable to perform classification with a procedure that is invariant to blur and illumination.

Various approaches have been proposed in the past to tackle the problems of illumination and blur. The most prominent among these approaches are built on top of insensitive moments [151], color constancy [152] and Fourier phase. But these approaches are designed to perform classification globally and do not take into account the local properties of objects. Various methods based on the magnitude, phase and spectral information [153] were also designed but these methods focused on texture analysis whereas blur and illumination invariance was not considered as a design criterion.

We propose Empirical Mode Decomposition(EMD) based feature fusion strategy to overcome the challenges of illumination and blur. Figure 4.1 shows the workflow of our proposed system. We split the input dataset into its Intrinsic Mode Functions (IMFs) by using EMD. Reisz transform is then applied on the generated IMFs to generate the monogenic images.

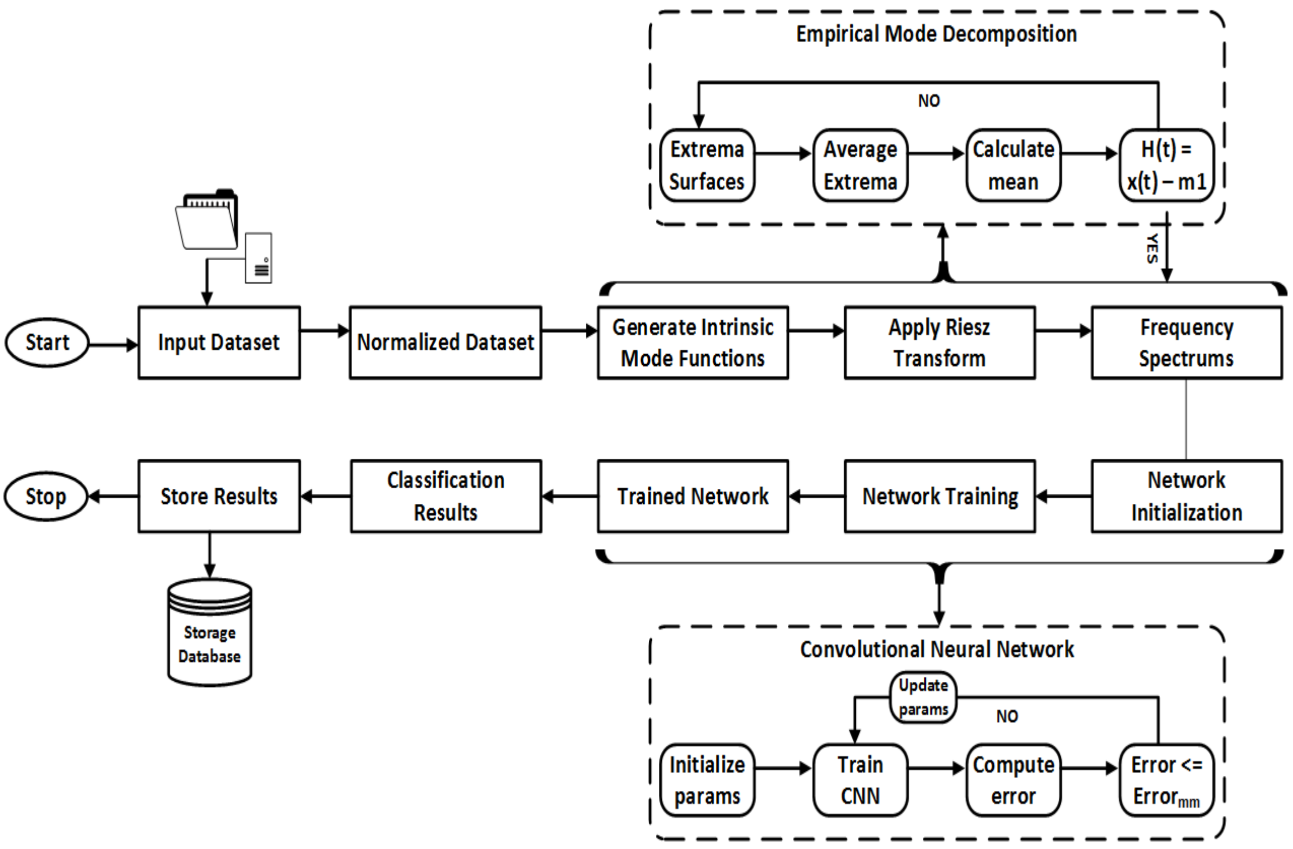


Figure 4.1: Workflow of the Feature Fusion Strategy

The local properties of the monogenic images including phase, orientation and amplitude are then studied and analyzed to have a high accuracy rate with the CNNs. It has been observed that the orientation property of face contributes to the higher accuracy rates. Inspired by this fact, we further propose a feature fusion strategy based on the orientation property of the first two IMFs which leads to further accuracy improvements.

In this chapter we have shown that only first two or three IMFs are sufficient to perform classification under challenging conditions with high accuracy rate. This is advantageous in two ways: i) Reduced feature extraction time as compared to other methods. ii) Illumination and blur invariance, since only lower IMFs are sensitive to variety effects. Then, we used the orientation property derived from each IMF using first order Reisz transform and showed that it provides good results than the phase or amplitude properties.

In order to perform classification from the fused features we have used deep learning based classifier described in Section 4.4. The optimization of the classifier is inspired by a mathematical model for efficient analysis of video streams. The mathematical model helps to observe the effects of different values of hyper-parameters on the deep learning model's performance . We have varied the parameters to different values between suitable ranges and selected the most optimum values to enhance the accuracy of the proposed system. The next section describes the feature extraction procedure and fusion strategy for blur and illumination invariant object classification.

4.3 Feature Extraction in Spatial-Frequency Domain

One of the most successful approaches to address the challenges of illumination and blur is to perform analysis of image or video frames by shifting them from spatial domain to spatial-frequency domain. In spatial domain, processing of video frames is performed by directly using the gray values of pixels. Spatial frequency domain allows the processing of video frames by projecting them on a set of basis functions which are defined by the method itself. This phenomenon expands the video frame into frequency components with both high and low magnitudes. For example, in the case of wavelet transform, the mother wavelet generates the basis functions.

Mother wavelet is shifted and scaled to produce these basis functions. Similarly, in Fourier domain, basis functions are complex exponentials. Variety of methods exist for the conversion of spatial domain signal to spatial frequency domain. These methods includes Fourier Transform [154], Wavelet Transform [155], Wigner distribution [156] and many others.

Since the basis functions are predefined by the methods itself, this makes them infeasible for non-linear and non-stationary processes. These methods are also not adaptive, neither are they data driven. Wu et al. [114] proposed an adaptive and data driven method known as Empirical Mode Decomposition (EMD) which is much focused for image analysis nowadays.

Empirical Mode Decomposition (EMD) doesn't decompose the data on a-priori basis but the basis functions are derived from the data. Hence, this approach is data driven and much more operative on non-stationary data. Non-stationary data is the data whose statistical properties change with time. EMD considers data as superposition of fast oscillations on slow ones. These oscillations are recognized by EMD and a decomposition by utilizing these modes as expansion basis is made.

4.3.1 Empirical Mode Decomposition

Empirical Mode Decomposition is a fully unsupervised approach. It defines its basis functions directly from the data and is not dependent on methods. It has been proved to be successful for biomedical and seismic signals. EMD expands a signal into its frequency components adaptively. These frequency components are termed as Intrinsic Mode Functions (IMFs) and are defined by the signal itself. EMD tries to extract highest frequency components from the original input signal in each mode. It separates the local highest frequencies and stores them into an IMF. The rest of the IMFs contain the remaining frequencies in the lowest order which ends up in a residual part.

Empirical Mode Decomposition allows to visualize spatial-frequency characteristics signal by expanding the parent signal into IMFs. Hilbert transform is applied on the IMFs afterwards to obtain instantaneous frequency which helps to study the local properties. These signals which are obtained after applying Hilbert transform are called analytic signals as they have no negative frequency components. These analytic signals help to obtain local amplitude and phase of the 1D signal.

4.3.1.1 Two Dimensional Empirical Mode Decomposition

In order to apply Empirical Mode Decomposition on the images Two Dimensional Empirical Mode Decomposition (2DEMD) or Bidimensional Empirical Mode Decomposition (BEMD) was introduced [157]. Since BEMD is data driven, it is much more suitable than Fourier or Wavelet transform for video frames. In this case, Riesz Transform is used instead of Hilbert transform [158]. Riesz Transform is the generalization of Hilbert Transform. When it is combined with the image, it produces a monogenic signal which is a local quantitative and qualitative measure of the image. Local amplitude, phase and direction can be calculated from the monogenic signal of each IMF.

Bi-dimensional Empirical Mode Decomposition provides several advantages over spatial domain analysis. Image features can be extracted easily according to the distribution of local phase or energy. Only 1 or 2 IMFs are enough to extract the features as these are the highest frequency IMFs. The feature extraction time is less than the other methods. The illumination and pose changes mainly appear in the residual part. Also, lower BMFs are sensitive to variety of effects. This means variety information is mainly contained in lower BMFs. So we can eliminate lower BMFs and residual to get rid of the variety effects.

4.3.2 Decomposition of a Video Frame

In this research work, we pioneer the use of Empirical Mode Decomposition for video streams. The EMD decomposes the video frame into its intrinsic mode functions. The video streams are first acquired by video capturing sources. These video streams are decoded to extract individual video frames. These video frames are artificially blurred with varying radius. Noise has also been added to the objects with different PSNR values. These objects are then classified by blur and illumination invariant feature descriptor. Figure 4.2 shows the approach of our proposed system.

The input training dataset in our system is represented by X and is given by;

$$\text{``}Training\ dataset\ X = x_1, x_2, \ldots, x_n\text{''} \tag{4.1}$$

Here, "$x_1, x_2, \ldots x_n$" represents the individual subjects present in the training database. The value of "n" goes up to 34 for each subject. Each individual subject in the training database consists of a number of training samples.

Each training sample "i" from each individual subject "x" undergoes Two Dimensional Empirical Mode Decomposition (2DEMD) to have a decomposition into its frequency components which are termed as Intrinsic Mode Functions (IMFs). This decomposition of training samples

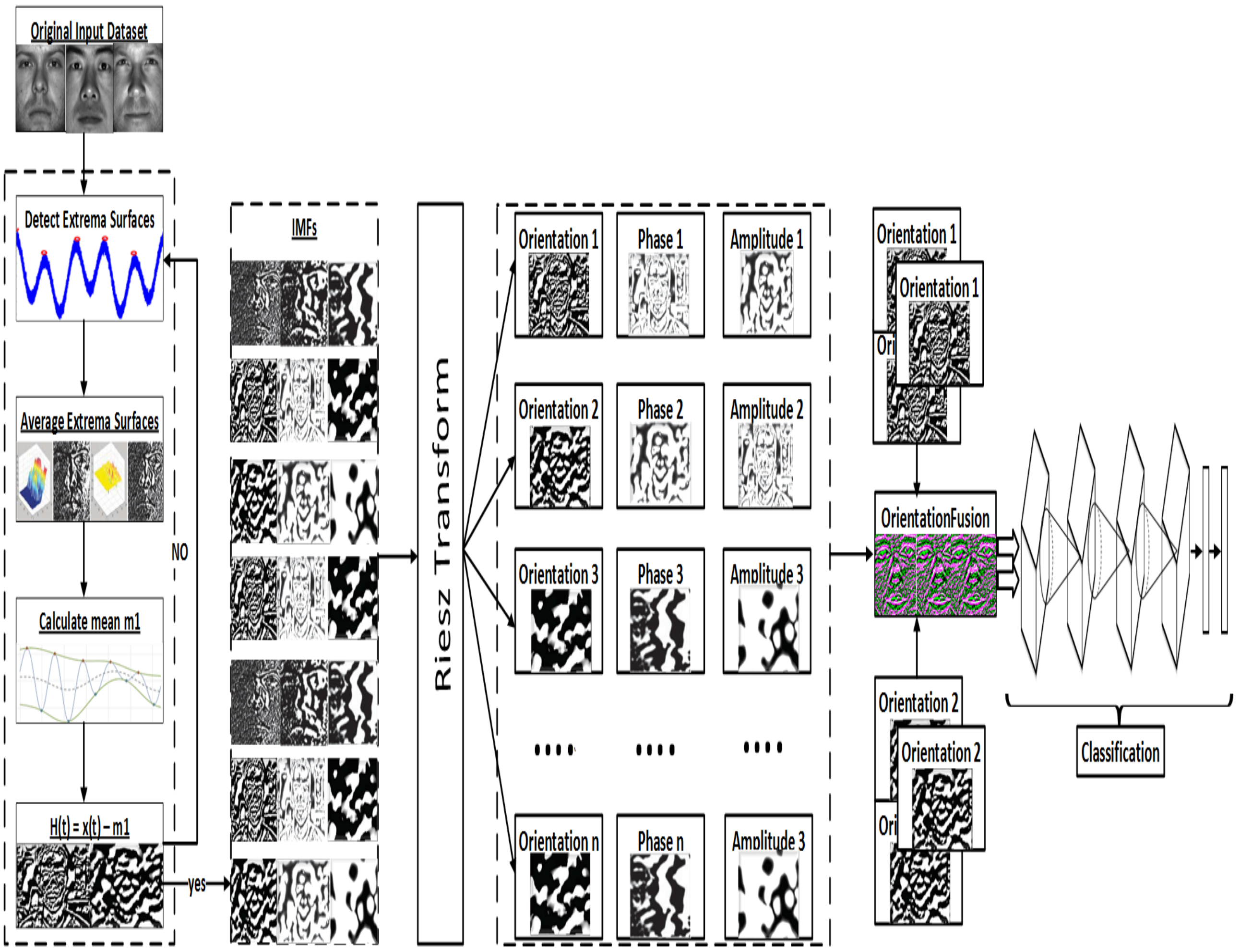


Figure 4.2: Proposed Approach

into IMFs is adaptive as EMD defines its basis functions directly from the data itself. EMD generates these IMFs by the sifting process in which the highest frequency components from the training sample are extracted in each cycle or mode.

Each mode stores the high frequencies as an IMF. These IMFs are stored in decreasing order of their frequencies and the lowest IMF contains the remaining lowest frequencies. This process ends up into the residue which contains the remaining lowest frequencies as shown in Algorithm 7. By combining all the IMFs and the residue, original image or video frame can be obtained. EMD allows visualizing spatial-frequency characteristics signal by expanding the parent signal in to IMFs.

$$
\begin{aligned}
``x_1 &= i_1, i_2, \ldots, i_n \\
x_2 &= i_1, i_2, \ldots, i_n \\
x_2 &= i_1, i_2, \ldots, i_n \\
\vdots & \qquad \vdots \\
x_{34} &= i_1, i_2, \ldots, i_n"
\end{aligned}
\tag{4.2}
$$

Where, "$i_1, i_2, \ldots, i_n$" represents the individual images of each subject present in the training dataset. The value of "n" goes up to 40 which are the total number of training samples of each subject.

Algorithm 7 Empirical Mode Decomposition

```
Input:
    Input Dataset $x_1, x_2, , x_n$, 192 x 168 image size
    Width of each image W
    Height of each image H
    Number of iterations m
    Number of IMFs n
Output:
    result: IMFs

while !residue do
    Let the proto-IMF be x̂(x,y) = x(w,h)
    while IMF ¡= 3 do
        while !criteria do
            Identify local maxima and minima of (w,h)
            Find envelop $e_{min}(x, y)$
            Find envelop $e_{max}(x, y)$
            Mean m(x,y)=($e_{max}(x, y)$ = $e_{min}(x, y)$ )/2
            Extract detail $h_1$ = x̂(x,y) - m(x,y)
            x̂(x,y) = $h_1$
            end
        x̂(x,y) - $\sum_{j=1}^{3} h_j(x, y)$
        end
    end
```

4.3.2.1 Sifting Process

Sifting process is an algorithm which is used to extract these IMFs from the signal. This algorithm tries to extract highest frequency components from the original input signal in each

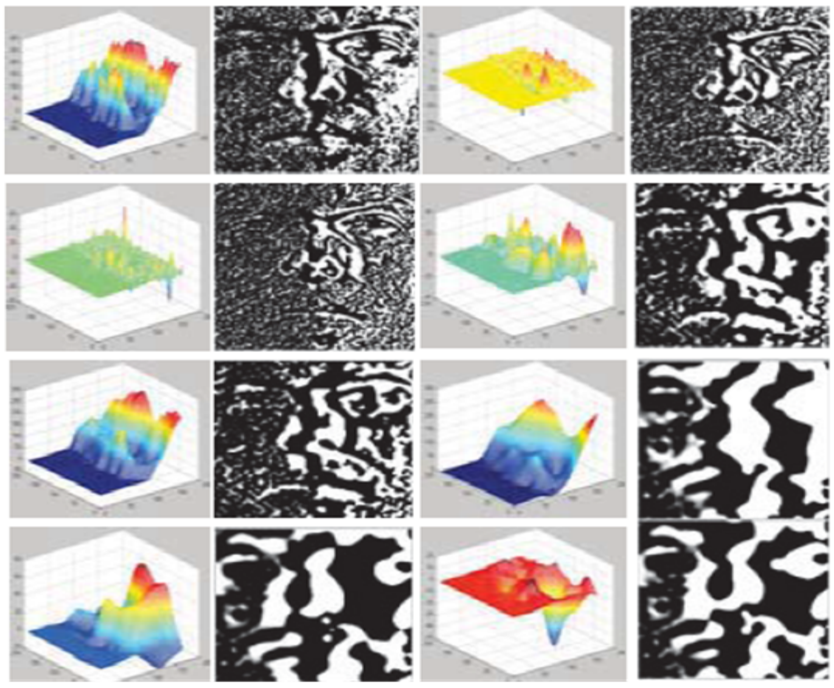


Figure 4.3: Averaged Extrema Surfaces

mode. It separates the local highest frequencies and stores them into an IMF. The rest of the IMFs contain the remaining frequencies in the lowest order. This ends up into the residue which contains the remaining lowest frequencies. By combining all the IMFs and the residue, the original signal can be obtained. The sifting algorithm is defined as follows:

4.3.2.2 Extrema and Envelop Calculation

The sifting process first determines the extrema points from the training sample "$k(i,j)$" , where "i,j" are the dimensions of training sample. These extrema points are connected to form upper and lower envelops. An average of the upper and lower envelop is calculated to produce mean envelop "$mean(i,j)$" and is given by;

$$\text{“}mean(i,j) = (eupper(i,j) + elower(i,j))/2\text{”} \tag{4.3}$$

The two envelopes i.e. the maxima envelope and the minima envelope, are averaged to generate the local mean envelope. Figure 4.3 shows the averaged extrema averaged surfaces of various video frames.

4.3.2.3 ProtoIMF Generation

The mean envelope “$mean(i, j)$” is then subtracted from the training sample “$k(i, j)$” to produce “$T1$” and is given by;

$$“Tl_k = I(x, y)m(x, y)” \tag{4.4}$$

The whole process is repeated till “Tl_k” is a two dimensional IMF. When the mean envelope “$mean(i, j)$” reaches close to zero then this process is stopped, otherwise it keeps on reiterating. The residual is obtained by removing the original training sample “$k(i, j)$” from “Tl_k”. If the residual is represented by “$Res(i, j)$”, it is given by;

$$“Res(i, j) = i(i, j) - Tl_k” \tag{4.5}$$

In order to obtain the next IMF, the whole procedure is repeated on the residual “$Res(i, j)$” by considering it as a training sample. Repetition of this process on all the subsequent residuals results in a number of IMFs in the decreasing order of their frequencies. All the resultant IMFs and the residual can be grouped together to obtain the original training sample.

4.3.2.4 First Order Riesz Transform

In case of one dimensional signals, Hilbert Transform is usually applied on the generated IMFs to obtain instantaneous frequency which helps to study the local properties. These signals which are obtained after applying Hilbert Transform are called analytic signals as they have no negative frequency components. These analytic signals help to obtain local amplitude and phase of the 1D signal.

In case of BEMD, Riesz Transform is used instead of Hilbert transform. Riesz transform is the generalization of Hilbert Transform as shown in Algorithm 8. When it is combined with the image, it produces monogenic signal which is a local quantitative and qualitative measure of

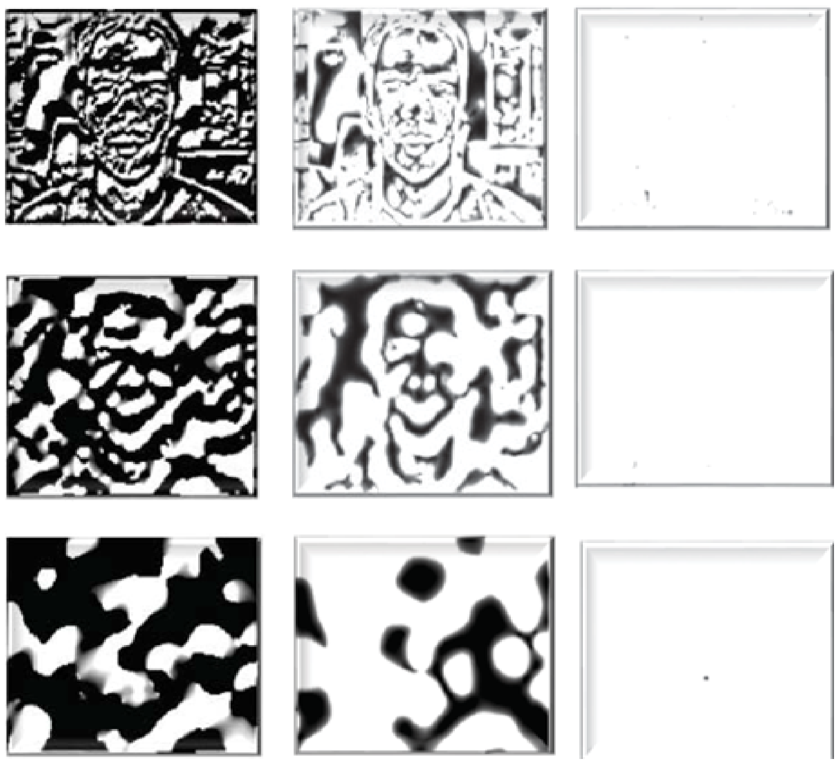


Figure 4.4: Amplitude, Phase and Orientation of first three IMFs

the image. Local amplitude, phase and direction can be calculated from the monogenic signal of each IMF. Figure 4.4 shows the amplitude, phase and orientation spectrums of first three IMFs.

The Riesz transform in the frequency domain is given as:

$$\text{“}f_R(x) = I \times (x/2\pi)x^3 \times f(x) = h_2(x) \times F(x)\text{”} \tag{4.6}$$

The 2D monogenic data which is formed by the original image and its Reisz transform is given by;

$$\text{“}f_m(x) = f(x)(I, j) \times f_R(x)\text{”} \tag{4.7}$$

Let “X_{amp}” , “X_{pha}” , “X_{ori}” be the amplitude, phase and orientation spectrums of all the training samples present in the database. These can be represented as;

Algorithm 8 Riesz Transform

```
1: procedure Riesz Transform
2:     Compute the Laplacian matrix $L$ of the surface
3:     Compute partial eigenvalue decomposition of $L$
4:     Generate $U_p$ and $R_p$ through partial eigenvalue decomposition
5:     Compute the inverse of the fractional Laplacian matrix $L$
6:     Generate the gradient to produce Riesz Transform
```

$$\begin{aligned} \text{“}X1_{amp} &= i_{amp1}, i_{amp2}, \ldots, i_{ampn} \\ X1_{pha} &= i_{pha1}, i_{pha2}, \ldots, i_{phan} \\ X1_{ori} &= i_{ori1}, i_{ori2}, \ldots, i_{orin} \\ \vdots \qquad & \qquad \vdots \qquad\qquad \vdots \\ X34_{amp} &= i_{amp1}, i_{amp2}, \ldots, i_{ampn} \\ X34_{pha} &= i_{pha1}, i_{pha2}, \ldots, i_{phan} \\ X34_{ori} &= i_{ori1}, i_{ori2}, \ldots, i_{orin}\text{”} \end{aligned} \tag{4.8}$$

Here, “x_{amp}” , “x_{pha} ... x_{ori}” represent the amplitude, phase and orientation spectrums of individual subjects present in the training database.

We propose a feature fusion strategy of the orientation property to represent the input video frames as a feature descriptor. The feature fusion strategy is performed on the first two intrinsic mode functions of the orientation property. The two intrinsic mode functions are fused together in order to have a composite intrinsic mode function which could hold the properties of both the two IMFs as shown in Figure 4.5 . The fused IMF is a numeric matrix which represents the combined orientation property. This represents as a feature vector of the whole video frame and is used for classification to improve the accuracy rates.

Let “$XFused_{ori}$” represent the fused orientation spectrum and “*” represent the fusion operation. The fused orientation spectrum of all the subjects is then given as;

$$\text{“}\triangle XFused_{ori} = \sum_{i=1}^{N} (x(i)_{ori1} * x(i)_{ori2})\text{”} \tag{4.9}$$

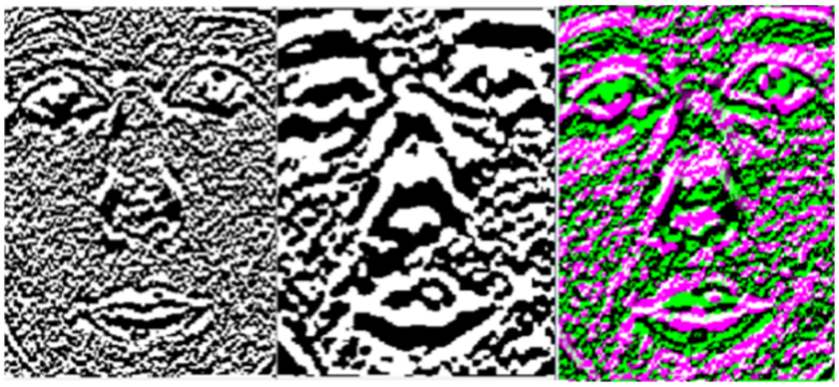


Figure 4.5: Orientation Fusion

This section described the feature extraction procedure through Empirical Mode Decomposition and explained the feature fusion strategy for blur and illumination invariant object classification. In the next section we explain and model the deep learning based Convolutional Neural Network for object classification.

4.4 Deep Learning Modelling

Convolutional Neural Networks (CNNs) have been used recently to perform visual object recognition on video datasets. CNNs proved to be successful on a number of object detection and classification tasks on large video datasets. They also have the generalization capability and can be trained on large scale video datasets belonging to different classes.

A Convolutional Neural Network [159] is a kind of multilayer neural network but it has some additional layers including convolutional and pooling or sub-sampling layers. These additional layers are then followed to fully connected layers as in standard multilayer perceptron. The convolutional layers contain a set of convolution kernels. Each kernel is a filter or mask which is convolved with the image to extract features.

There can be many convolution kernels in each layer. This results in the formation of feature maps. The pooling layer is next to convolution layers. The purpose of this layer is to subsample a small rectangular box or feature map taken from convolution layer in order to reduce variance. A single output from each rectangular block is generated. A number of convolution and pooling

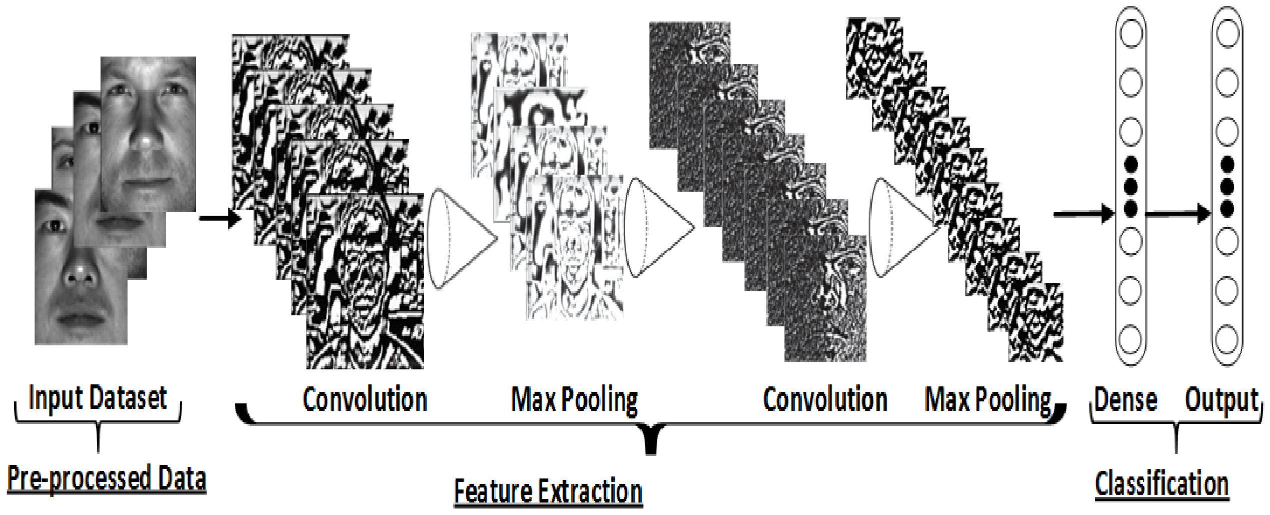


Figure 4.6: Schematic Diagram of the Proposed Network

layers lead to the fully connected Multi-layer Perceptron (MLP) layer.

An MLP is a standard multilayer neural network. This layer takes all the neurons from previous layer and connects it to every single layer within itself. CNNs are most suitable for images as their architecture is designed in a way to take full advantage of the 2D structure. In this case the input to the convolution layer is a 2D image and kernels are applied to it. The convolution with the image produces feature maps. Next, the mapping is performed on each feature map. This further leads to fully connected layers i.e. MLP layer.

We have proposed the architecture of a Convolutional Neural Network for the classification of objects from fused orientation features. The schematic diagram of the Proposed Network is shown in Figure 4.6. The hyper-parameters of the proposed Convolutional Neural Network are further tuned through a mathematical model for optimized performance in the next chapter of this thesis.

4.4.1 Data Augmentation

The labeled training data used in our system is scarce and in order to enhance it for optimal performance, we executed transformations on the input dataset including translation, skew, rotation flip and different levels of contrast variations. The additional training data by using transformations increases the accuracy of the classifier. These transformations are generated by applying affine displacement fields to video frames.

This is achieved by calculating the new location (x,y) with reference to the original location for each pixel of the video frame. For example, if x(x,y)=1, and y(x,y)=0, it shows that the new location of each pixel is shifted to the right by 1. For a displacement field of x(x,y)= αx, and y(x,y)= αy, the video frame will be shifted by α, from the origin point (x,y)=(0,0), where, α can be any non-integer value. Let T be the set of transformations applied on the training dataset. The training dataset with transformations is represented as:

$$TXN = TxN_1, TxN_2, \ldots TxN_n \tag{4.10}$$

Algorithm 9 Training weight vectors on local properties

```
Input:    Input Dataset $x_1, x_2, , x_n$, 192 x 168 image size

Output:    result: Recognition Labels    $f_{co} \Leftarrow FullyConnected$
                     $result \Leftarrow Softmax(f_{co})$
while epoch r: 1 → R do
    while Training image number x: 1 → X do
        Compute J hidden activation matrices $z_1, z_2, ..., z_j$
  •         g($x_k, l + w_k, l + B_k, l$)
        Downsample matrices $z_1, z_2, ..., z_j$ by a factor of 2
  •         g($\downarrow^2 x_k, l + w_k, l + b_k, l$)
        Calculate weight and bias deltas
  •         $\triangle W_t, k = LR \sum_{i=1}^{F} (x_i * D_i^h) + m\triangle W_{(t-1,k)}$
  •         $\triangle B_t, k = LR \sum_{i=1}^{F} D_i^h + m\triangle B_{(t-1,k)}$
        Calculate softmax activation vector 'a'
  •         $l(i, x_i T) = M(e_i, f(x_i T))$
        Compute error $y_x - a$ Back propagate and update network weights
  •          $W_{t+1} = W_t - \alpha\delta L(\theta_t)$
        end
    end
```

4.4.2 Classifier Training

The proposed Convolutional Neural Network has been trained on the fused orientation features of intrinsic mode function. Algorithm ?? shows the training procedure of the network on fused features. The network learns the weights and activation matrices during training for each layer of the Convolutional Neural Network.

The extracted objects from the video frames are stored in a multidimensional data structure provided by an open-source library named as Nd4j [160]. A n-dimensional array (so called tensors) is created to store the pixel values of video frames. It consumes minimum memory and supports fast numerical computing for Java. The loading of data into the memory and training of the network is handled by two separate processes. This makes the data loading process simple and is supported by the nd4j library.

A dataset iterator is defined to iterate over the data present in the memory. The iterator has the capability to iterate over the data which is loaded into the memory. The iterator fetches the data from memory in a vectorised format. The iterator moves to the dataset objects which contains training examples along with their labels. Each dataset object contains multiple examples, depending upon the configuration.

The video frame data is also normalized to have the pixel values between 0 to 1. The normalization of data helps the gradient descent optimization approach to converge properly during network training. The gradient descent requires more than one example at a time during the training as more examples will help to create a gradient that encompasses more errors than a single example. A good gradient, when using gradient descent approach, greatly helps to improve the training, makes the learning consistent and helps to converge to a usable result.

4.4.2.1 Convolutional and Sub-sampling Layers

The network consists of multiple alternating layers of convolutional and sub-sampling layers. The convolutional and sub-sampling layers can be given as:

$$Conv_k, l = g(x_k, l * W_k, l + B_k, l) \tag{4.11}$$

Similarly the sub-sampling layer is given as;

$$Sub_k, l = g(\downarrow x_k, l * w_k, l + b_k, l) \tag{4.12}$$

Where g(.) represents the activation function which is 'ReLU' in our system. W and B are the weights and biases of the system. The convolution operation between input and the weights of the network is represented by '*'. The sub-sampling layer contains all the inputs in the downsampled form. The activation function of the layers is given as:

$$h = max(0, a) where a = Wx + b \tag{4.13}$$

4.4.2.2 Weight and Bias Deltas

The weight and bias deltas for the convolutional layers are given as:

$$\triangle W_t, k = LR \sum_{i=1}^{F} (x_i * D_i^h) + m\triangle W_{(t-1,k)} \tag{4.14}$$

for bias;

$$\triangle B_t, k = LR \sum_{i=1}^{F} D_i^h + m\triangle B_{(t-1,k)} \tag{4.15}$$

The weight and bias deltas for the sub-sampling layers are given as:

$$\triangle W_t, k = LR \sum_{i=1}^{F} (\downarrow x_i * D_i^h) + m\triangle W_{(t-1,k)} \tag{4.16}$$

and the bias for sub-sampling layer;

$$\triangle b_t, k = LR \sum_{i=1}^{F} D_i^h + m\triangle b_{(t-1,k)} \tag{4.17}$$

The loss function in our case which we try to minimize is given by:

$$L(x) = LR \sum_{x_i->X} \sum_{x_i->T_i} l(i, x_i T) \tag{4.18}$$

where l(i,xT) is the loss function that we try to minimize during network training. We employed stochastic gradient descent which tries to reduce the loss function during training. It is given by:

$$W_{t+1} = W_t - \alpha\delta L(\theta_t) \tag{4.19}$$

where w is the weight change with respect to the gradient of the loss function and α is the learning rate. The gradient of the loss function changes rapidly due to the variance present in our training examples after each iteration, so we apply momentum term to keep it smooth and is given by:

$$V_{t+1} = \rho v_t - \alpha\delta L(\theta_t) \tag{4.20}$$

$$W_{t+1} = W_t + V_{t+1} \tag{4.21}$$

The convolutional neural network has the softmax layer as the output layer of the network and optimizes negative log likelihood. This can be given as:

$$l(i, x_i T) = M(e_i, f(x_i T)) \tag{4.22}$$

where f(x,T) denotes the function to calculate output values and 'e' is the basis vector.

Algorithm 10 Object Classification

```
        Input:
            Input Target Object                      x, 192 x 168 size
            Output Target Label                      T 1-in-k vectors $y_1, y_2, , y_t$
            Number of Trained Patterns               R
            Similarity Function                      f(.)

        Output:
                result: Classification Labels    $f_{co} \Leftarrow FullyConnected$
                                                  $result \Leftarrow Softmax(f_{co})$
  while Training vector x: 1 → R do
          Load the trained classifier            TR(x)
          Apply pre-processing steps on target object's sample
          Pass the sample through the trained network
          Calculate softmax activation vector 'a'      $l(i, x_i T) = M(e_i, f(x_i T))$
          Compute similarity index               $y_x - a$
          Generate the set of possible probabilities    $i$
  end
```

4.4.2.3 Object Classification

The trained classifier is then used to perform object classification for our video analytics system. The number of object classes to be determined by the system are represented as:

$$Y^{(i)} = 1, 2, \ldots K \tag{4.23}$$

where 1,2, . . . K represents the total number of possible classes. Our hypothesis function for the classification task is given by:

$$h_\theta(x) = 1/(1 + \exp(-\theta^T x)) \tag{4.24}$$

where θ represents the trained model parameters that minimizes the cost function. The hypothesis function estimates the probability P(y = k — x) for each value of $K = 1, 2, \ldots K$. The system then estimates the probability of the class label and output a vector with 'K' estimates probabilities. The hypothesis function takes the form:

$$h_\theta(x) = \begin{cases} P(y = 1|x; \theta) \\ P(y = 2|x; \theta) \\ \ldots \\ P(y = K|x; \theta) \end{cases} \tag{4.25}$$

A sample of the target object, which is to be classified in the video streams, is passed through the trained classifier after training. The trained classifier returns the probabilities or matching scores of the possible labels for the object, as explained in Algorithm 10 . The object with the highest probability (above an empirically determined threshold) indicates the classification of the object.

4.5 Conclusion

A blur and illumination invariant video analytics system based on Convolutional Neural Network has been proposed in this chapter. The proposed system overcomes the challenges of blur and illumination by employing a feature fusion strategy on orientation properties of intrinsic mode functions. It has been observed that the orientation property leads to higher accuracy rates. We further investigated that the feature fusion strategy of the orientation properties of first two IMFs further improves the accuracy rates.

We further presented a deep learning model based on fused features to perform object classification from video streams. The system is based on Convolutional Neural Network whose parameters are optimally tuned for accurate classification of objects. It can enhance its own training data by performing various transformations including rotation, flip and skew on the input dataset. The system learns features from large amounts of input data by performing training in parallel on a multi-node in-memory cluster.

In the following chapter of this thesis, we present the hyper-parameter tuning of the proposed system and describe the distributed training procedure in detail. The efficient tuning of the hyper-parameters of the proposed system through mathematical model makes it highly accurate and robust to classification errors. We have shown the factors that contributed to achieve high accuracy such as optimal selection of learning rate, regularization, normalization and optimization algorithms. The design of multi-layer network, including number of layers and their parameters, also played a major role in achieving high accuracy in the system. We also present the experimental setup in the next chapter which is used to execute and evaluate the proposed system.

Chapter 5

Video Analytics Optimization

5.1 Outline

In Chapter 4, a cloud-based blur and illumination invariant system for object classification from image and video data was presented. The Bi-dimensional Empirical Mode Decomposition (BEMD) was adopted to decompose a video frame into Intrinsic Mode Functions (IMFs). The analysis of each IMF was then carried out by observing its local properties (amplitude, phase and orientation) generated from each monogenic component. A feature fusion strategy was adopted to fuse monogenic component features to be fed into the Convolutional Neural Network. A Convolutional Neural Network was then optimally tuned to perform object classification.

Chapter 5 will provide a detailed description of the hyper-parameter optimization, distributed training and experimental setup that is used to execute and evaluate the proposed system. The image and video datasets used in the proposed system for evaluation and validation purposes are also described. The dataset enhancement techniques are also presented along with the evaluation methodology.

5.2 Introduction

Deep learning has emerged recently as an influential tool to achieve high accuracy and precision in computer vision related tasks. They also have the generalization capability and can be trained on large scale input dataset belonging to different classes. However, in the analysis of video stream data, deep learning algorithms face major challenges such as availability of large amount of labelled data, fine tuning of hyper-parameters and training time of the deep network.

Deep learning based video analytics systems can involve many hyper-parameters, including learning rate, activation function and weight parameter initialization. A trial-and-error approach is mostly followed in selecting these parameters, which makes it time consuming, and at times, may provide inaccurate results. They also struggle to perform well on the challenging datasets and their accuracy severely drops, especially for the case of expression and illumination variant datasets.

The proposed video analytics system is built upon a deep learning model whose optimization is inspired by a mathematical function for efficient analysis of video streams. The mathematical model helps to tune and to observe the effects of different values of hyper-parameters on the deep learning model's performance. We have varied the parameters to different values between suitable ranges and selected the most optimum values to enhance the accuracy of the proposed system.

After presenting the theory, mathematical model and implementation details of the proposed video analytics system, we also present the experimental setup used to evaluate the proposed system. We have used the cloud computing paradigm to perform the training of the proposed video analytics system. Multiple nodes of the cluster have been used to train partial models on each node. This reduces the training time as compared to training the whole system on a single node. The parameters of the underlying cluster are finely tuned to achieve maximum utilization of available resources. This enables the system to perform rapid computation and helps to process large amounts of data.

The training process is further enhanced by utilizing iterative map-reduce framework instead of

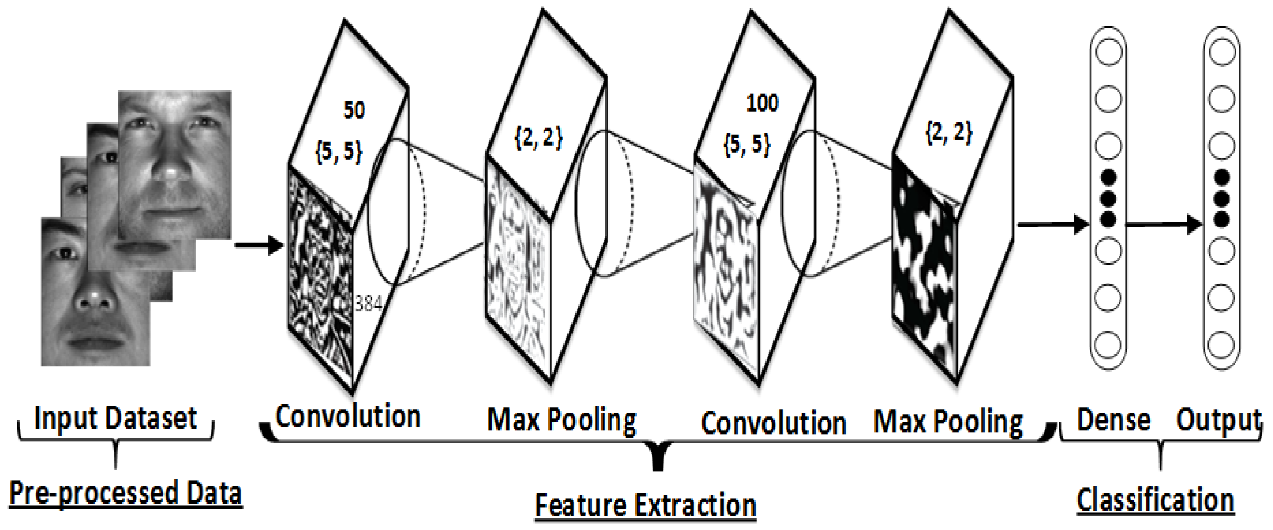


Figure 5.1: Convolutional Neural Network Architecture

simple map-reduce. We have shown that distributed training by utilizing iterative map-reduce is an efficient way to reduce the training time of the system. The partial models have been trained on each node of the cluster and their results are combined on the master node. The classifier, after training the network, is used further for performing object classification.

In order to evaluate the proposed system we have performed a number of experiments on publically available image and self-generated video datasets. The dataset is further enhanced to increase the number of video frames by applying transformations including rotation, flip and skew. These transformations help to generate more data from the labelled data without bearing any further labelling cost. More training data helps to expose the deep network against more training samples and reduces the chances of over-fitting.

5.3 Configuration Model for the Proposed CNN

We present the architecture of our proposed Convolutional Neural Network model in this section. The model architecture is designed on the basis of experimentations. The proposed model provides higher accuracy rates along with high precision. The architecture of the Convolutional Neural Network model used in the proposed system is as follows: The first convolutional layer

Table 5.1: Model Configuration

Layer Info.	**CNN 1**
Input Size	1
Layer Size	50
# Parameters	1300
Weight Init.	XAVIER
Updater	RMSPROP
Kernel Size	[5,5]
Stride	[1,1]
Padding	[0,0]
Activation	relu

of the network receives and filters the training examples with a dimension of 192 x 168 x 1 with a total of 50 kernels as shown in Figure 5.1.

Each kernel has a size of 5 x 5 x 1 and a stride of 1 x 1. Stride controls how the depth columns around the spatial dimensions are assigned. The following convolutional layer receives and filters the input with 100 kernels. Each kernel in this layer has a size of 5 x 5 and a stride of 1 x 1. The model configuration of the model is tabulated in Table 5.1.

The kernels of the subsequent layers are associated to the kernels of the previous layers. Each of these convolutional layers constitute of nonZeroBias. These convolutional layers are followed by a max-pooling layer with a size of 2 x 2. All of these layers are followed by a fully connected layer. The neurons in the fully-connected layer are associated to the neurons of the previous layer. ReLU non-linearity layer follows all the layers of the network. More detailed specifications of these layers are listed in Table 5.2.

The convolutional layers and pooling layer are followed by the fully connected layers. The fully connected layers have a total of 4096 neurons in them. The kernels and neurons of the following layers are connected to the kernels and neurons of preceding layers. We have also added two response normalization layers which follow the first two convolution layers. There are three max pooling layers in total which follow the first two local response normalization layers and the last convolution layer. ReLU non-linearity layer follows all the layers of the network.

Table 5.2: Layer Configuration

Layer Info.	**Dense Layer**	**Output Layer**
Input Size	7200	500
Layer Size	500	34
# Parameters	3600500	17034
Weight Init.	XAVIER	XAVIER
Updater	RMSPROP	RMSPROP
Activation	relu	softmax

5.4 Distributed Training Model for CNN

The distributed training model of our proposed video analytics system is presented in this section. The proposed work is targeted to work on large number of video streams. The main challenges of large scale video analysis involve storage, computation power and memory requirement. Therefore, it is deployed on a cloud-based infrastructure. Deployment on the cloud-based infrastructure helps to cope with these challenges.

Cloud computing is a good approach to tackle the scalability challenge as well. It is achieved with large scale data centres which are hosted on remote servers. The storage requirement for large amount of video data is always high. Cloud storage is helpful to meet the storage requirement. Cloud computing also enables an on-demand and on-the-fly analysis of video streams.

Also, the system is compute intensive as it is built upon convolutional neural network which requires large training times. We have tackled this problem by optimizing the code, tuning the hyper-parameters properly and by introducing parallelism with multiple nodes. Parallelism is achieved by distributing the dataset into small subsets, and then, passing over these subsets of data to separate neural network models as shown in Figure 5.2.

The next subsection provides details of parallel and distributed training on the cloud infrastructure. We also discuss the tuning parameters of cloud nodes as well as the deep learning model in the later sections of this thesis.

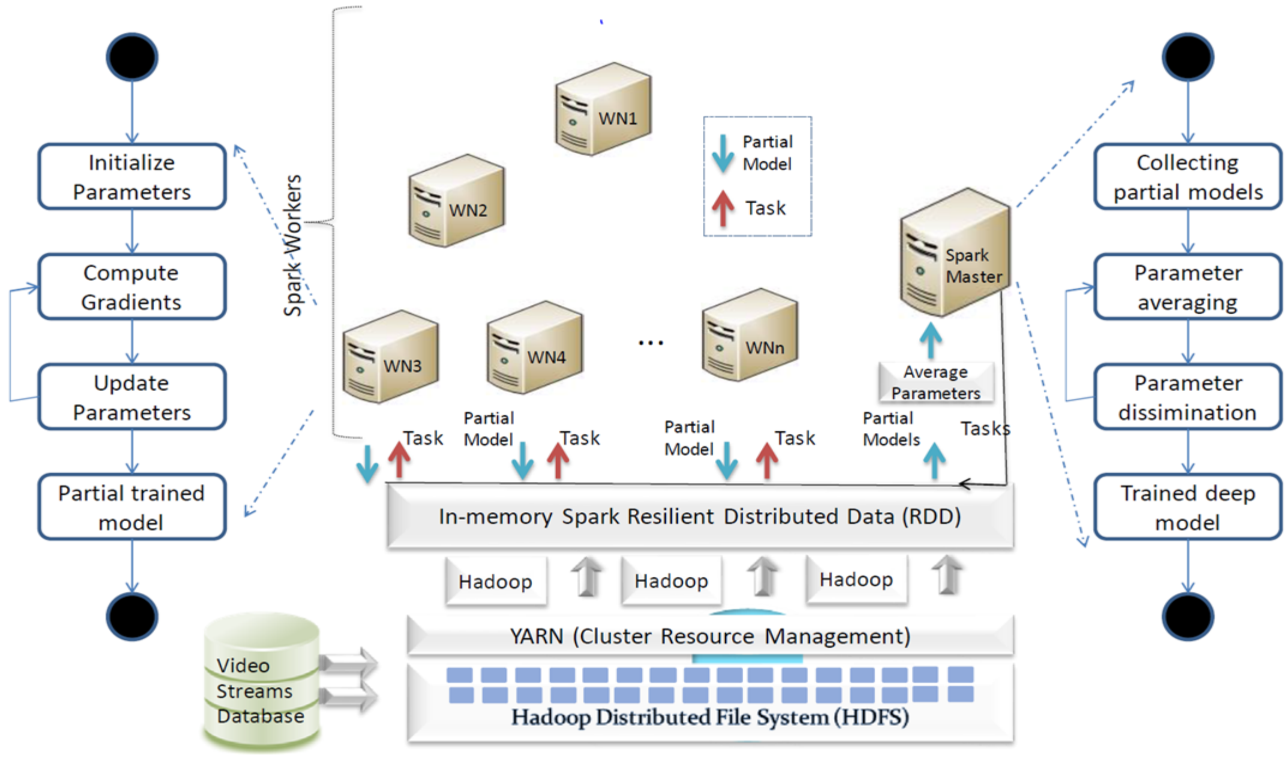


Figure 5.2: Distributed Training Procedure

5.4.1 Training Procedure

To achieve parallelism on cloud nodes, the Hadoop MapReduce framework is used. Hadoop has its own mechanism for storing files termed as Hadoop File System (HDFS). The video streams are first required to be transferred into the HDFS. The MapReduce framework is used to analyze video streams by executing object classification algorithms on them. The analysis results are then stored in the database.

The Hadoop MapReduce framework has a NameNode responsible for load balancing among the nodes. The Data/Compute Nodes are used for storing and processing the data. The JobTracker in each compute node is used to track the tasks, and in case of failure, to reschedule the tasks. The JobTracker is also aware of the data locality and can, therefore, distribute the jobs to the data nodes, where the data resides. This improves the performance and ensures data locality during the analysis task.

Each cloud node is responsible for executing a combination of map and reduce tasks. The object classification approaches are executed by the map task. The map task is also responsible for

Table 5.3: Configuration Parameters

Spark Configuration		**Model Configuration**	
Parameters	**Values**	**Parameters**	**Values**
spark.worker.cores	1	Number of Layers	15
spark.worker.instances	1	nonZeroBias	1
spark.eventLog.enabled	TRUE	DropOut	0.5
spark.scheduler.mode	FIFO	OptimizationAlgo	SGD
spark.serializer	KRYO	Activation	RELU
spark.rpc.message.maxSize	250	Regularization	L2
spark.locality.wait	0	Momentum	0.9
Averaging Frequency	1	Seed	42
Batchsize per Worker	12	Learning Rate	0.0001

generating analysis results by performing classifications on the video streams. The function of reduce task in our system is to write analysis results on the output file.

The training process starts by first loading the training dataset into the memory. The master node is responsible for loading network configuration and the initial parameters into the memory. The master is termed as driver node as well because it is responsible to drive other nodes of the cluster by distributing parameters among them. The network configuration holds the information about the division of data into subsets.

5.4.2 Mini-Batches and Serialization

The dataset is divided into various subsets of data. Each subset is further divided into various minibatches depending upon the configuration. Training is performed on each subset by allocating each minibatch to each worker. Since the dataset is large in size, it was not possible to load the whole dataset into memory at once. So we have first exported the minibatches of datasets to disk (HDFS) known as dataset objects. Each dataset is then used by each worker to perform training. Each worker trains a partial model and the averaged results through iterative averaging from all the worker nodes are sent back to the master node. The master node holds the fully trained model capable of performing object classification on the test data.

The datasets are exported in batched and serialized form. We have used kryo serialization to perform serialization of our dataset. This approach of saving the dataset to disk is much

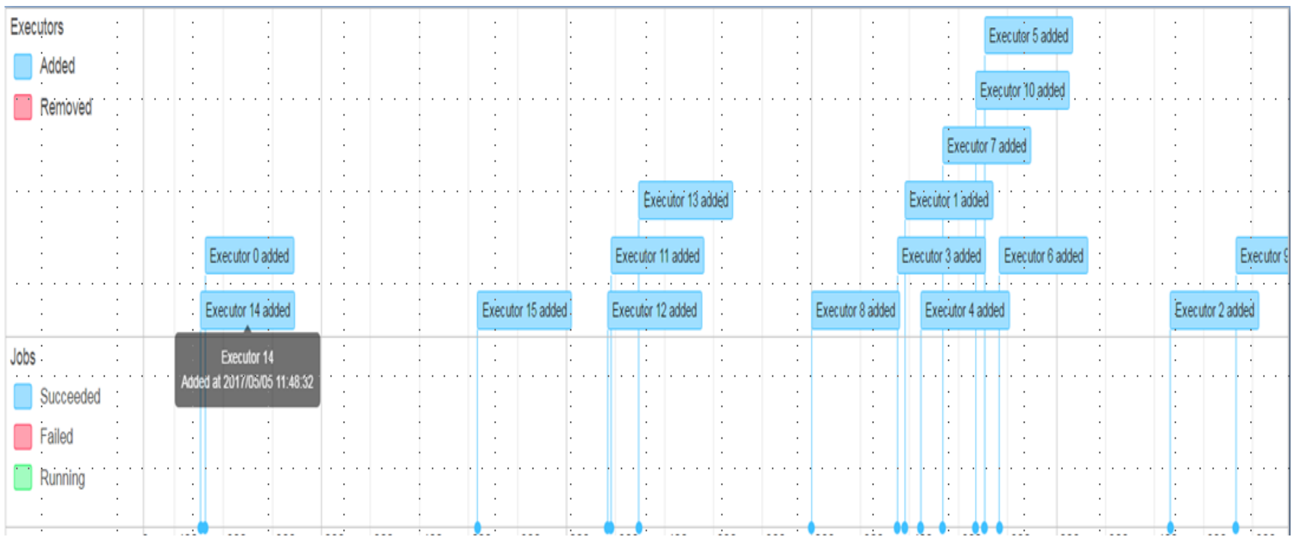


Figure 5.3: Spawning of Multiple Executors

more efficient and faster as compared to loading the whole dataset in memory. This approach consumes less memory and reduces split overhead. The dataset object has a number of examples based on the size of dataset object. Kryo serialization takes the least amount of time to serialize objects and improves performance. It can serialize objects much quickly and efficiently and offers more compact serialization than Java. The serialization framework provided by Java has high CPU and RAM consumption which makes it inefficient for large scale data objects.

5.4.3 Averaging of Trained Parameters

The separation of training data into subsets and then training the model with these subsets of data by averaging parameters is a feasible approach for our system because we operate with limited worker nodes in our cloud and the parameters for estimation are also small. We use the same model for each worker node but train them on different data shards (mini-batches). We then obtain the gradient for each split of the mini-batch from each model and compute the overall average using parameter averaging. This technique works faster for small networks as in our proposed system and is ideal for scenarios involving matrix computations which happens quite often in Convolutional Neural Networks.

It is also desirable to control the rate at which the parameters are averaged and redistributed among the nodes. If the rate of averaging parameters is set to low, this can cause an overhead

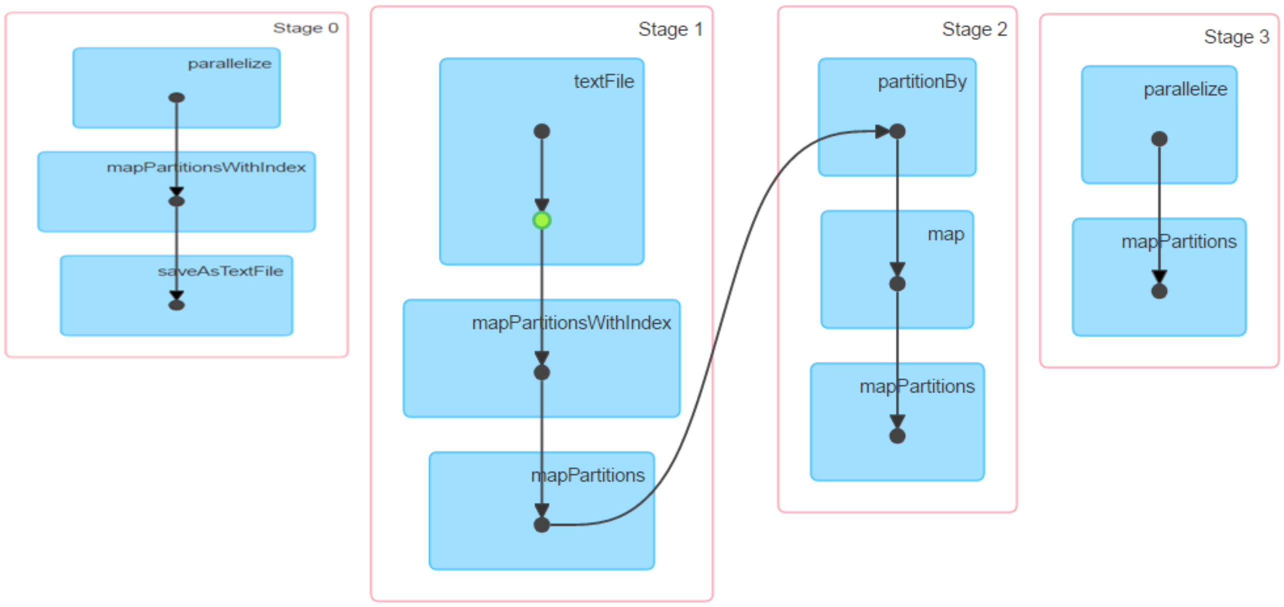


Figure 5.4: Stages of Iterative Reduce

in initialization as well as network communication. Similarly, if the rate is set to be very high, it can degrade performance as the parameters will deviate extensively. In our case the optimal performance is achieved with a frequency of 16 minibatches. The minibatches are loaded asynchronously to avoid the delay in loading into the memory. We have configured its value to be 16 as a higher value can result in more use of memory.

A number of executors are launched during training as shown in Figure 5.3. The executors access a RDD object in each iteration. All the nodes in the cluster execute analysis tasks through iterative map-reduce. The iterative map reduce supports multiple map and reduce operations in an analysis task. These tasks are executed in multiple stages and their further mappings in each stage as depicted in Figure 5.4 and 5.5.

5.5 Hyper-Parameter Tuning of CNN

Rate of Parameter Averaging: It is quiet important to set the rate of parameter averaging. If this is too low, this will create overhead in parameter initialization and will cause delay in network communication. Similarly, if it is high, it will degrade the performance. In the proposed video analytics system, good performance is obtained with 16 mini-batches. These mini-batches

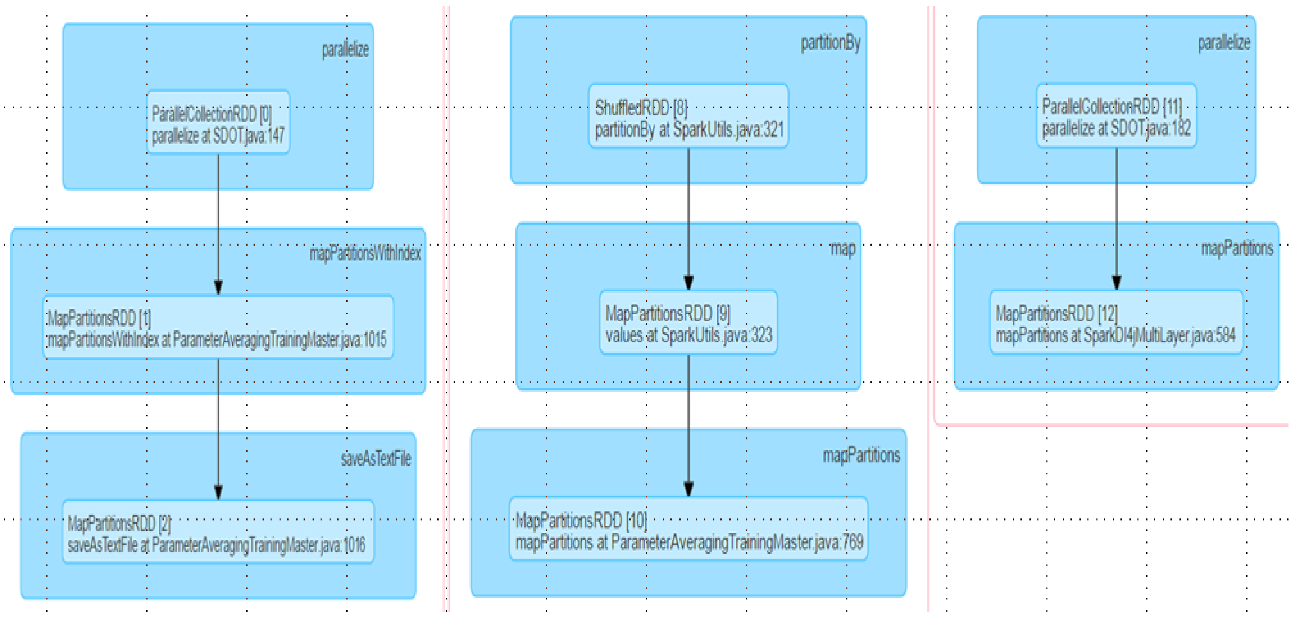

parallelize
ParallelCollectionRDD [0]
parallelize at SDOT.java:147
mapPartitionsWithIndex
MapPartitionsRDD [1]
mapPartitionsWithIndex at ParameterAveragingTrainingMaster.java:1015
saveAsTextFile
MapPartitionsRDD [2]
saveAsTextFile at ParameterAveragingTrainingMaster.java:1016
partitionBy
ShuffledRDD [8]
partitionBy at SparkUtils.java:321
map
MapPartitionsRDD [9]
values at SparkUtils.java:323
mapPartitions
MapPartitionsRDD [10]
mapPartitions at ParameterAveragingTrainingMaster.java:769
parallelize
ParallelCollectionRDD [11]
parallelize at SDOT.java:182
mapPartitions
MapPartitionsRDD [12]
mapPartitions at SparkDl4jMultiLayer.java:584

Figure 5.5: Operations inside each Stage

are started in an asynchronous fashion which reduces the delay.

Data Repartitioning: An important parameter which is required to be tuned for the training process is to select when data is to be repartitioned. In order to utilize all the resources of the cluster efficiently, it is important to select the number of partitions properly. The values which each partition will hold also need to be configured carefully. We have chosen a value of 0.6 based on the experimentations as it ensures the correct number of partitions (balanced partitions). We have not used the default repartition strategy of Spark as it does not ensure that each partition is balanced.

Locality Configuration: The locality configuration is also defined as the proposed algorithm has high demand of computation, so single task per executor is executed. It is, therefore, much suitable to shift data to executor which is free. The default configuration of Spark waits for a free executor. This requires the data to be copied across the network.

Another important note is that we have avoided the allocation of memory on JVM heap space by passing pointers for various numerical tasks. It is not required to load the data from JVM heap to execute operations on it; neither it requires data transmission (processed results) back to JVM. This helps to avoid the data transfer time and a decrease in overall execution time of the system is seen. This also avoids memory overhead required for each task.

Iterative Mapreduce: The iterative MapReduce framework utilized in this work executes multiple analysis tasks. These analysis tasks are executed in multiple stages and each stage performs further mapping operations. The analysis tasks can be rescheduled if a task failure occurs. The Spark context is utilized to load the video dataset and is then stored into multi-dimensional arrays.

The multi-dimensional arrays represents the data in the form of tensors which are then passed through multiple layers of Convolutional Neural Network for training. The proposed system is based on Convolutional Neural Network and is highly iterative so the single pass of MapReduce does not perform quite well. It fully exploits the advantages of iterative MapReduce and performs MapReduce operations in a cascaded way (preceding MapReduce becoming the input to subsequent MapReduce and so on).

Local Response Normalization: We have adopted the Local Response Normalization to aid generalization. Local Response Normalization simulates the behavior of actual neurons and generates a competition amongst neuron outputs for big activities. We have added two response normalization layers which follow the first two convolution layers. The three max pooling layers follow the first two local response normalization layers and the last convolution layer. The local response normalization is given by:

$$b_{(x,y)} = a_{(x,y)}/(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^2_{(x,y)})) \tag{5.1}$$

where $a_{(x,y)}$ is the convolutional kernel's output and $b_{(x,y)}$ is the output of local response normalization.

Regularization: L2 regularization has been introduced to tackle the problem of over-fitting which penalizes the network weights and controls them in becoming too large. L2 regularization adds

$$\lambda_2 \sum_i \theta_i^2 \tag{5.2}$$

where network weights are represented by theta and lambda is the lagrange multiplier.

Rectified Linear Unit(ReLU): Instead of using the standard hyperbolic tangent non-linearity, we adopted 'ReLU' as suggested by [161]. ReLU is much more appropriate than tanh especially in case of bigger datasets as the network trains much faster. Traditional hyperbolic tangent non-linearity does not allow training the system on bigger datasets.

The ReLU non-linearity is also used in [162] and proved to be successful than traditional saturating neuron models. The ReLU function has a range of [0,infinity], so it has the capability to model positive real numbers. The advantage of using ReLU is that it does not vanish as the value of x increases as compared to sigmodal function. The max function is:

$$\text{“}1\ if\ x > 0;\ 0\ if\ x < 0\text{”} \tag{5.3}$$

Max Pooling: We have used max pooling in the pooling layer to perform sample based discretization or downsampling of an input representation (feature maps from convolutional layer in our case). Max pooling decreases the dimensionality, reduces the number of parameters to learn and also cuts down the overall computational cost.

Learning Rate: The learning rate is set to be 0.0001. This value has been selected carefully on the basis of experimentation. It is observed during the experimentations that if the learning rate is set too high, it can cause divergence of the model away from the minimum error. This can halt the learning process. On the other hand, if the value of learning rate is set to a small value, it causes slow convergence on an error minimum.

Number of epochs: The learning rate is set to be 0.0001. This value is chosen very carefully based on the experimentation. A higher value of learning rate can result in the divergence of the network model away from the error minimum. This will cause the learning process to stop. A small value of the learning rate leads to a slow convergence on an error minimum.

The number of epochs and iterations are set to be 5 and 3 respectively. Epoch is the complete pass through our video dataset during the network training. It ensures that the network has seen each example present in the dataset once. Iteration, on the other, hand is a single update of the network parameters. Each epoch during the training phase contains three iterations in our setup.

Mini-batches of Data: The high volumes of data makes it infeasible to load the data into the memory at once. So a number of mini-batches of the dataset have been used in order to tackle the memory requirement problem. As we are working on a large scale video dataset, the volume of data is quite high and it is not practically possible to store whole data at once in the memory.

Also, the mini-batches of dataset help to have more updates on the network in one epoch. We have used a mini-batch of 12 in our work. The mini-batch is large and capable enough to represent the input video data and contains all the classes of the objects.

Stochastic Gradient Descent: We employed stochastic gradient descent which tries to reduce the loss function during training which is given by:

$$W_{t+1} = W_t - \alpha \delta L(\theta_t) \tag{5.4}$$

where w is the weight change with respect to the gradient of the loss function and α is the learning rate.

Momentum: The gradient of the loss function changes rapidly due to the variance present in our training examples after each iteration, so we apply momentum term to keep it smooth and is given by:

$$V_{t+1} = \rho v_t - \alpha \delta L(\theta_t) \tag{5.5}$$

$$W_{t+1} = W_t + V_{t+1} \tag{5.6}$$

5.6 Experimental Setup

We presented the training procedure and hyper-parameter tuning of the proposed system in the previous section. In this section we present the experimental setup used to execute and evaluate the proposed system as shown in Figure 5.6. The experimental setup mainly consists of cloud configuration, image and video datasets and evaluation parameters.

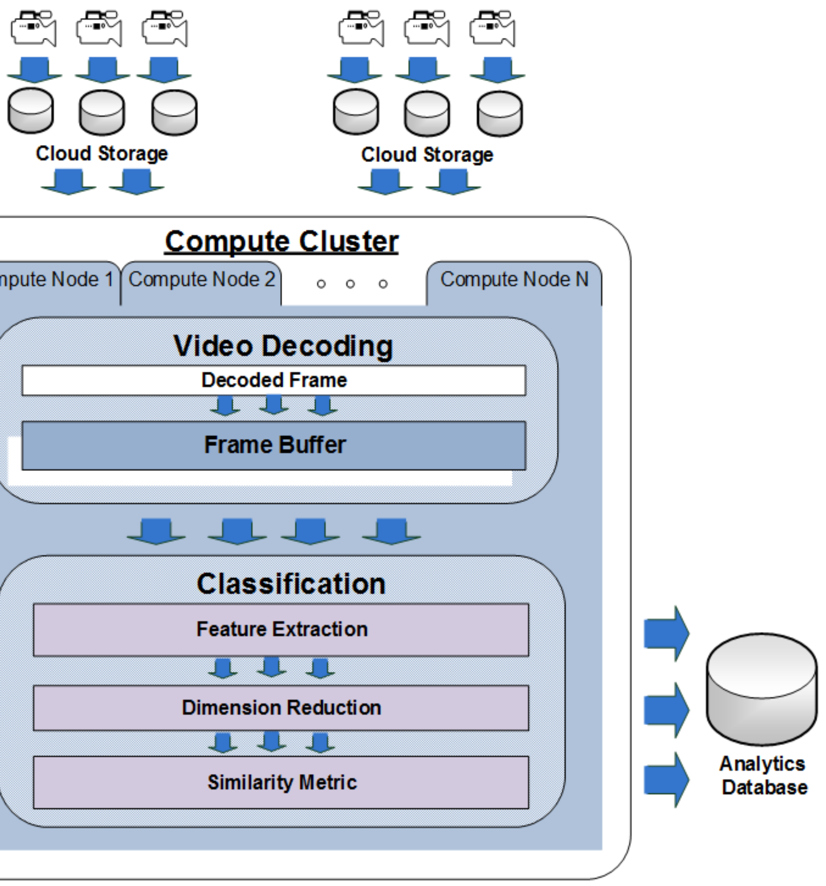


Figure 5.6: Experimental Setup

5.6.1 Model Deployment in Cloud

We have deployed our system on the cloud resources available at the University of Derby for evaluation. The configuration of the cloud resources are as follows: The cloud instance has Ubuntu LTS 14.04.1 and is running OpenStack Icehouse. There are six server machines and each server machine is equipped with 12 cores. Each server is running with 6-core Intel Xeon Processors at 2.4 Ghz. It has a storage capacity of 2 Terabyte with 32GB RAM. OpenStack facilitates with a dashboard to manage and control the resources such as storage, network and pool of computers. It is a Gigabyte Ethernet between the nodes with an average latency of 10ms for packets travelling from one VM to another.

The virtual machines are equipped with multi-core CPU processing capacity in which most of the video analytics operations are performed. The configuration of each node is as follows: 4 VCPU running at 2.4GHz with 8GB RAM. Each node is configured with a storage capacity of

100GB. This cluster is used to deploy and evaluate the proposed system. This experimental setup helps to measure the performance of the system for scalability; reliability; accuracy; performance and robustness with varying cloud configurations. The results generated by these experiments can help to deploy the system on a much bigger infrastructure as per requirements of an application.

5.7 Datasets for Training and Testing the Model

To perform rigorous evaluation of the proposed object classification system, self-generated and publically available datasets have to be used. Results from multiple datasets help to evaluate the reliability and steady of the proposed system. The self-generated or publically available datasets being used in the study should have multiple subjects and should be captured under variant conditions. The datasets should possess challenges such as illumination, facial expressions, blur and video surveillance context.

5.7.1 Self-generated Datasets

In this research study, we have used video datasets as well as image datasets to evaluate the proposed system. We have a self-generated video dataset consisting of videos of human faces of various individuals as shown in Figure 5.7. The video streams recorded for the experiments are relatively simple but do have challenges such as pose and illumination variations.

The video streams are by default in the encoded format with H.264 format having an fps of 25, data rate and bit rate of 421 kbps and 461 kbps respectively. These are decoded to generate frames which contain side, front and rear views of individuals. Most of the video streams in the video dataset have time duration of around 120 seconds. This makes a total of 3000 video frames in each video stream.

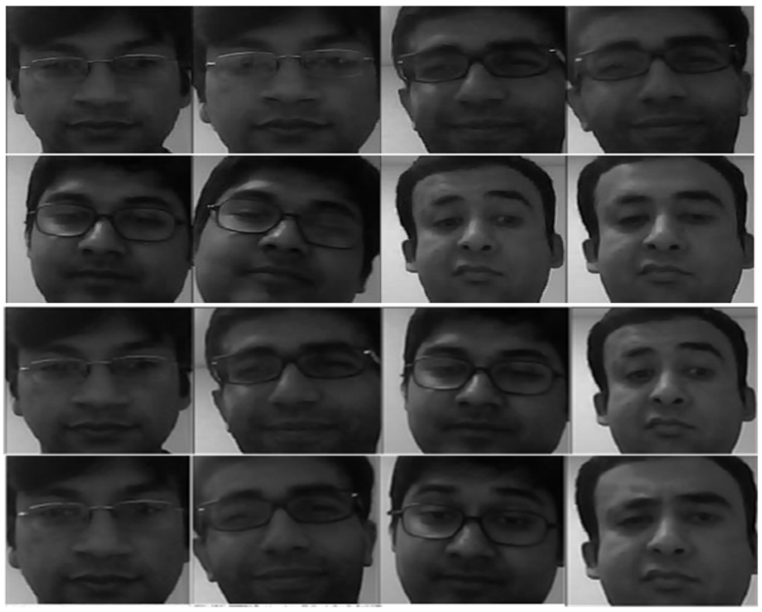


Figure 5.7: Self Generated Dataset

5.7.2 Surveillance Performance Evaluation Initiative Dataset

We have also used Surveillance Performance Evaluation Initiative (SPEVI) dataset which is another video dataset containing significant statistical data variability, but a limited number of data items. We have used its single person/face visual detection and tracking dataset. The dataset is composed of several sequences with different illumination conditions and resolutions. These sequences are shot with a hand held camera.

5.7.3 BioID Dataset

We have considered two large publically available image databases namely BioID [163] and Yale [164] to assess the accuracy and stability performance of the proposed system. During the recording of BioID Face Database, special importance has been given to real world situations. Therefore, this database contains a diversity of face sizes, background conditions and illumination effects as shown in Figure 5.8. The database comprises of 1521 gray level images captured at resolution of 384x286 pixels. For testing purposes, the images are artificially blurred by using a Gaussian blur mask with various sigma values (0, 0.25 . . .).

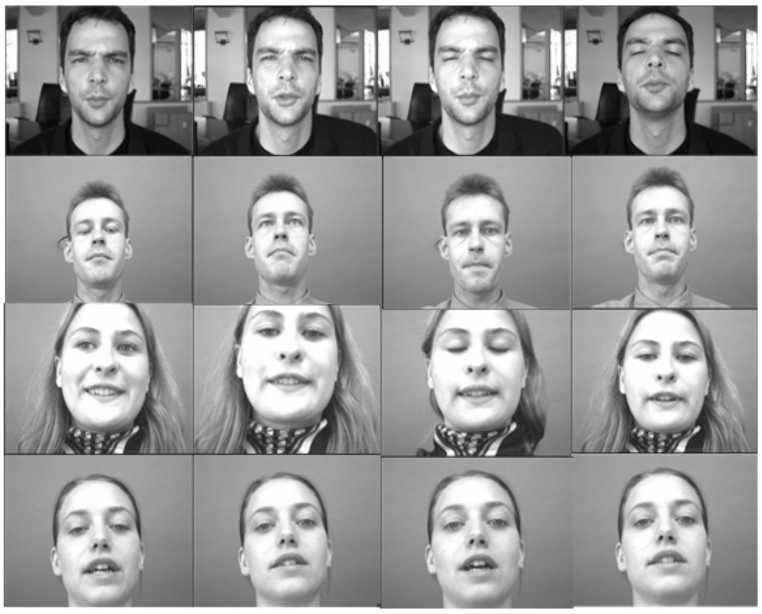


Figure 5.8: BioID Face Database

5.7.4 Yale Dataset

The Yale Face Database contains images from various subjects with different poses and illumination conditions as shown in Figure 5.9. All the images are manually aligned and cropped having a resolution of 168x192 pixels. Every subject demonstrates variations in illumination conditions (left-right, center-right, right-right) and facial expressions (normal, sad, happy, and sleepy).

5.7.5 DataSet Quality Enhancement Techniques

The pre-processing techniques are employed to enhance the quality of video or image data which is to be classified. These technique are more often used when the underlying data is captured under more challenging conditions and contains challenges such as, illumination and blur. The illumination and blur factors, if present in the data being analysed, can severely degrade the system's performance. These factors can completely distort the image or video frame and are also capable of changing the appearance of the object present in the video stream.

In order to select the most appropriate pre-processing techniques for the proposed system, we tested many of them on the proposed system. The techniques providing the best results were

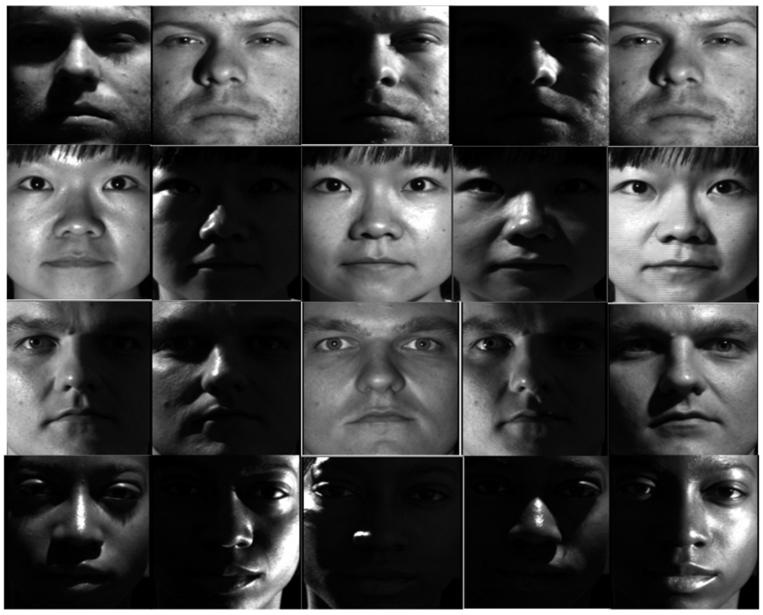


Figure 5.9: Yale Face Database

Algorithm 11 Histogram Equalization

```
1: procedure HISTOGRAM EQUALIZATION
2:     $h \leftarrow$ Compute the histogram of the object
3:     $CDF_h \leftarrow$ Compute the CDF of $h$
4:     Find the transformation which satisfies
5:
           $$T(r) = (L-1)\sum_{n=0}^{r} h(q) = (L-1)CDF_h(r)$$
6:     Apply the transformation to each pixel
```

then selected for experiments. Histogram equalization is used for shallow networks and to test the performance of the system on blurred data, artificial guassian blur was used. Normalization was applied on the video frames, in case of deep networks.

5.7.5.1 Histogram Equalization for Contrast Adjustment

The histogram equalization technique is used for publically available image datasets as well as for self-generated video dataset. The image datasets especially cropped value has high variations in illumination effects. In most of the images, very little fine details are available.

Also, some objects after cropping from the self-generated video dataset, are small in size. The other noise removal techniques distort the edges in them and no improvement in the accuracy is observed. Histogram equalization is a simple pre-processing technique but works well for our proposed system as compared to other existing techniques. It retains most the visual features during equalization and improves the object's global contrast.

In histogram equalization, the occurrence of the intensity values of the images are calculated. These occurrences can be drawn in the form of a histogram. These occurrences of the intensity values of the images are then distributed in a normalized guassian fashion so that they can occur equally in the resultant image as depicted in Algorithm 11. This is normally performed with the help of a two-stage process including histogram normalization and mapping of intensities.

The maximum intensity value of the image from the histogram is first identified and then the number of pixels in the image having that intensity value are also identified. A probability function is then defined to estimate the probability of pixel having the highest intensity value. These probabilities are further used for histogram equalization.

5.7.5.2 Gaussian Blur for Noise Reduction

The BioID Face Database contains images which have variations in pose and expressions. In order to test the performance of the classifier on more challenging datasets we have further introduced artificial blur by performing a convolution operation with the Gaussian blur mask. The Gaussian blur, which is also sometimes called as Gaussian smoothing, occurs when the image is convolved by a Gaussian function in order to reduce the fine details present in the image. This can also be used to reduce the noise present in the input data.

In this work, the values of the Gaussian blur mask (known as sigma) is ranged from 0 to 5, with zero being no blur and 5 being the maximum blur effect. With the help of this additional processing on the input dataset, it becomes possible to observe the joint effects of pose, expressions and blur on the performance of the classifier. For testing purposes, the images are artificially blurred by using a Gaussian blur mask with various sigma values (0, 0.25 . . .).

5.7.5.3 Data Normalization for Training

In case of deep learning algorithms, we have scaled and normalized the input video frames at a size of 150*150 pixels before feeding them into the deep neural network. The normalization is performed to have the pixel values between the range of 0 to 1 instead of 0 to 255. The Convolutional Neural Network can perform better with the normalized input data. This is because of the fact that Convolutional Neural Network used in this work is based on gradient descent algorithm and uses back propagation for learning. The normalization is performed as follows:

$$V = v - min_A / max_A - min_A (nmax_A - nmin_A) + nmin_A \tag{5.7}$$

where min and max represents the minimum and maximum values of an attribute 'A'.

During back propagation, CNN computes the error vector using the input and output data and multiplies it with the learning rate and weight matrices for each training example. The normalization of the data between a specified range helps to keep the distribution of features similar to each other. The corrections made by the learning rate during training will not differ from each other. Normalization is usually performed for all gradient-based learning schemes.

5.7.5.4 Data Augmentation to Increase Dataset Size

Deep learning algorithms, including Convolutional Neural Networks usually necessitate the input training data to be of a larger size. It helps the CNN to train on much more training examples and increases its generalization capacity. But the labeled training data used in our system is scarce and in order to enhance it for optimal performance, we executed transformations on the input dataset including translation, skew, rotation flip and different levels of contrast variations. The additional training data by using transformations increases the performance of the classifier.

These transformations are generated by applying affine displacement fields to video frames.

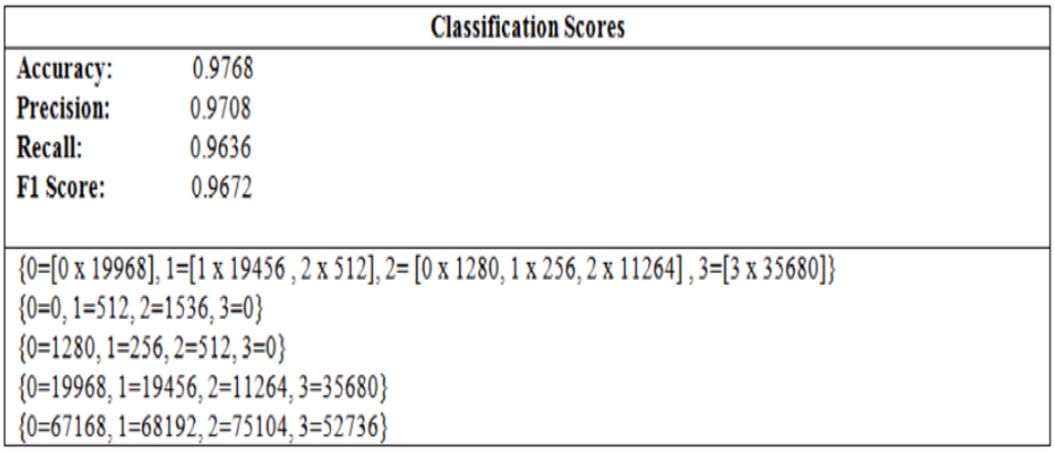
This is achieved by calculating the new location (x,y) with reference to the original location for each pixel of the video frame. For example, if x(x,y)=1, and y(x,y)=0, it shows that the new location of each pixel is shifted to the right by 1. For a displacement field of x(x,y)= αx, and y(x,y)= αy, the video frame will be shifted by α, from the origin point (x,y)=(0,0), where α can be any non-integer value.

5.8 Evaluation Methodology

We have used evaluation metrics, including efficiency metrics and effectiveness metrics, to verify the performance of the proposed video analytics system. The efficiency matrix measures the performance of the system by the following performance characterization: Hyper-parameters tuning of the system and system's scalability. The scalability of the system is analysed by measuring the time to transfer data to cloud, execution time of the system, and overall time of analysis of data. We discuss the training of the proposed model and visualize its training parameters during model training.

We have discussed the performance of the system during training and visualize the performance graphs by tuning the selected parameters. We analyse the results generated by tuning the hyper-parameters of the deep model to various values and propose the parameters which could potentially produce best results. The visualization of weight vectors and other parameters during training help tuning parameters properly.

The trained system on the proposed parameters is then evaluated with the effectiveness metrics. The effectiveness metrics are adopted to measure the quality of the proposed system in our experiments, are the commonly used ones including precision, recall and F-measure. The effectiveness metrics are further explained in the sub-sections below:



Classification Scores	
Accuracy:	0.9768
Precision:	0.9708
Recall:	0.9636
F1 Score:	0.9672

```
{0=[0 x 19968], 1=[1 x 19456 , 2 x 512], 2= [0 x 1280, 1 x 256, 2 x 11264] , 3=[3 x 35680]}
{0=0, 1=512, 2=1536, 3=0}
{0=1280, 1=256, 2=512, 3=0}
{0=19968, 1=19456, 2=11264, 3=35680}
{0=67168, 1=68192, 2=75104, 3=52736}
```

Figure 5.10: Confusion Matrix

5.8.1 Confusion Matrix

The evaluation of the proposed video analytics system and its algorithms is an important step to understand how good the system is in determining the correct object classification. In order to evaluate the performance of the proposed system and algorithms, we have used a number of performance measures. Most importantly, we have used confusion matrix which measures the performance of the system by counting the number of true positives, false positive, true negatives and false negatives. The confusion matrix represents the actual value of a class (labels) and the predicted value of that class by the classifier in a tabular format as shown in Figure 5.10.

The confusion matrix table suggests how good the classifier is in determining the right classes at right times. This is performed by counting the number of true positives, false positive, true negatives and false negatives. Accuracy, recall, precision and F1 score can be calculated by using these four counts as follows:

$$\begin{aligned} Accuracy &= (TP+TN)/(TP+FP+FN+TN) \\ Precision &= TP/(TP+FP) \\ Recall &= TP/(TP+FN) \\ F1 &= 2TP/(2TP+FP+FN) \end{aligned} \tag{5.8}$$

The true positives in the confusion matrix are the positive predictions of the object by the classifier when the actual outcome or label was also positive. False positives are the positive predictions of the object by the classifier when the actual outcome or label was negative. True negatives are the negative predictions of the object by the classifier when the actual outcome or label was negative. False negatives are the negative predictions of the object by the classifier when the actual outcome or label was positive.

False positives are also termed as 'type I errors' and false negatives are also termed as 'type II errors'. These metrics are also helpful for an in-depth analysis about the performance of the system. These metrics are further used to calculate various evaluations of the proposed system including accuracy, recall, precision and F1 score.

5.9 Conclusion

Chapter 5 described the hyper-parameter tuning and learning procedure of the proposed video analytics system. It also described the experimental setup used to execute and evaluate the proposed system. We also discussed the video and image datasets used to asses the performance of the proposed system. We explained the criteria and characteristics of the selected datasets and highlighted their distinguishing features in terms of blur, illumination, variations in pose and low resolution. Afterwards, we explained the pre-processing techniques which are used to enhance the quality and size of the datasets.

In chapter 6, the results of the proposed video analytics system and its algorithms will be presented. The results are demonstrated both graphically and analytically to explain that the proposed system can classify objects from large number of video streams with high accuracy. It also shows the visualization of training parameters during training depicting the optimization of classifier. The results also describe that the proposed system can also perform better even under controlled illumination condition and outperforms the existing techniques.

Chapter 6

Results and Discussion

6.1 Outline

Chapter 5 discussed the optimization and experimental setup of our proposed video analytics system. We provided a detailed description of the tuning of the hyper-parameters of deep learning network used in this research work. We highlighted the parameters that have a major contribution in improving the accuracy and performance of the proposed system. A detailed description of the experimental setup being used in this work was also presented.

In Chapter 6, we present the results of the proposed video analytics system. Section 6.2 describes the approach which is adopted to present the results. Section 6.3 explains the results obtained by executing the experiments with our proposed video analytics workflow. Section 6.4 shows the performance of the system under uncontrolled illumination conditions in terms of accuracy, precision, recall and F1 score. Section 6.5 explains the results from the deep learning pipeline. It shows the deep learning pipeline performance on various parameters from which best parameters are selected for object classification. Section 6.6 describes the results of our proposed video object detection and classification system in terms of execution time, training time, performance and scalability.

6.2 Introduction

The preceding chapters of this thesis presented the theoretical and mathematical details of proposed system and its algorithms. An in-depth explanation about the implementation details of the system was also provided. We now detail the results and outcomes of the practical investigations of the system in this chapter and also provide an analysis of the results which were obtained through experimentations as follows:

Firstly, we present the results of the proposed video analytics workflow which is used for object detection and classification. We show the the performance of the workflow in terms of speedup in processing time, video stream decoding time and execution time required for each frame.

Secondly, we improve the performance of the proposed video analytics system by shifting it from spatial domain to spatial frequency domain. Spatial frequency domain allows the processing of video frames by projecting them on a set of basis functions which leads to further performance improvements. We detail and discuss the performance measures of Amplitude, Phase and Orientation Properties and compare their accuracies. We further explain the performance of the proposed system in terms of accuracy, precision, recall and F1 score. The results from orientation fusion are then provided and compared with state-of-the-art.

Thirdly, we show how we have tuned and improved the hyper-parameters of the proposed video analytics system with the help of deep learning based algorithms. We discuss the training of the proposed model and visualize its training parameters during model training. The visualization of weight vectors and other parameters during training helps for tuning parameters properly. The performance of the deep learning pipeline on various parameters are analysed and discussed in detail and the best parameters are selected for object detection and classification system.

Fourthly, we explain in detail the performance related results of the proposed video analytics system and present our findings in terms of execution time, training time and scalability. The result explains the scalability and robustness of the whole system by analyzing decoded video streams and transferring the video data from local storage to cloud nodes. It also measures the time required to analyze video data on the cloud nodes.

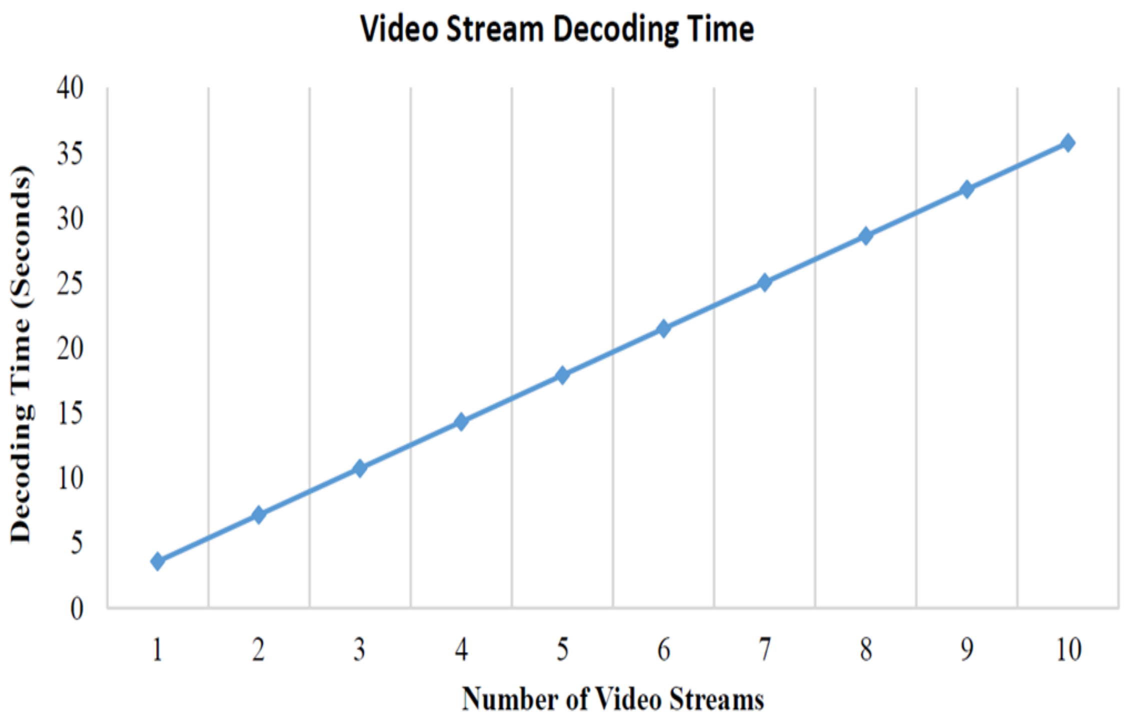


Figure 6.1: Video Reading and Decoding Time

6.3 Video Analytics Results

This section explains the results obtained by executing the experiments with our proposed video analytics workflow. We show the speedup achieved by introducing the cropping process in the proposed workflow. We then show the performance of the workflow for video stream reading and decoding time and execution time required for each frame.

6.3.1 Video Reading and Decoding Time

This experiment was performed to evaluate the time required to complete the video reading and decoding time in the proposed workflow. As the video streams are increased from 1 to 10, we analysed that video reading and decoding time increased with the addition of video streams as shown in Figure 6.1. It is because a video stream in our workflow is acquired by the video processing manager from the physical storage. This means that a constant transfer of video streams from the physical storage to the memory of video processing manager is required.

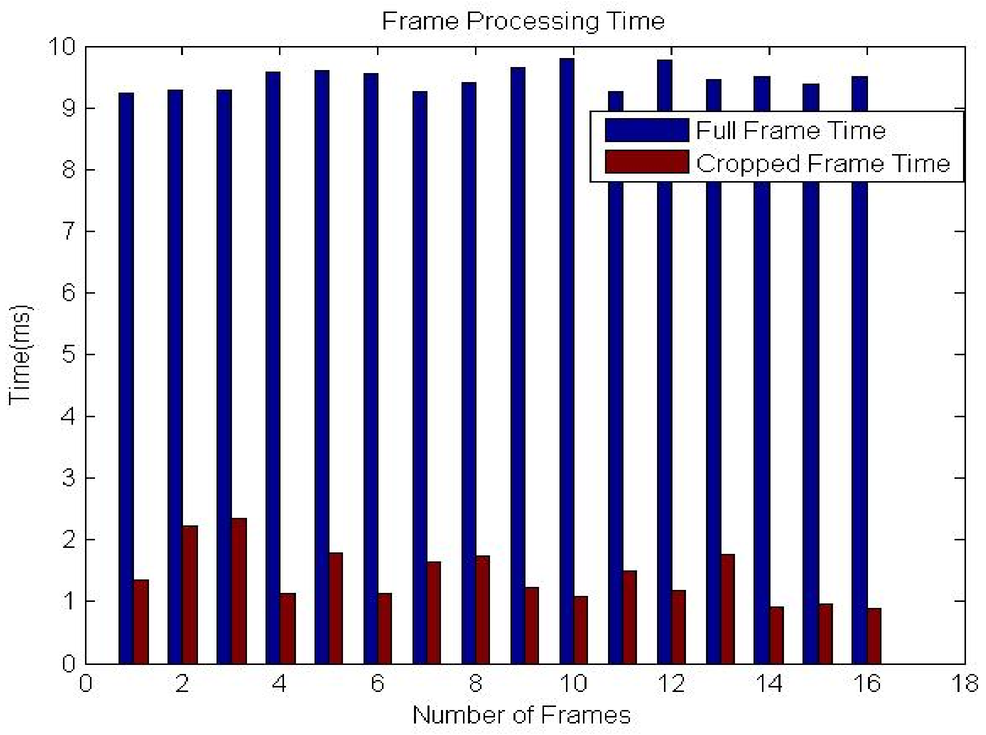


Figure 6.2: Cropped Frame Processing Time

6.3.2 Cropped Frame Processing Time

A significant amount of speedup is achieved in the processing time of each frame due to the object detection approach. Cropping of a video frame around the detected object helps to reduce the processing area for the LBPH algorithm. The resolution of the overall video frame is decreased, which in turn reduces the overall processing time of each frame. The processing time of each individual frame before cropping and after cropping is calculated and is shown in Figure 6.2.

The decrease in processing time is due to the decrease in resolution because of cropping. The video used in this experiment had a frame resolution of 640 x 480. However, the detected object which was extracted from the whole frame and later used by LBPH for comparison had a resolution of around 160 x 160 in most of the cases. This decrease in resolution improved the total frame processing time by almost 90 percent.

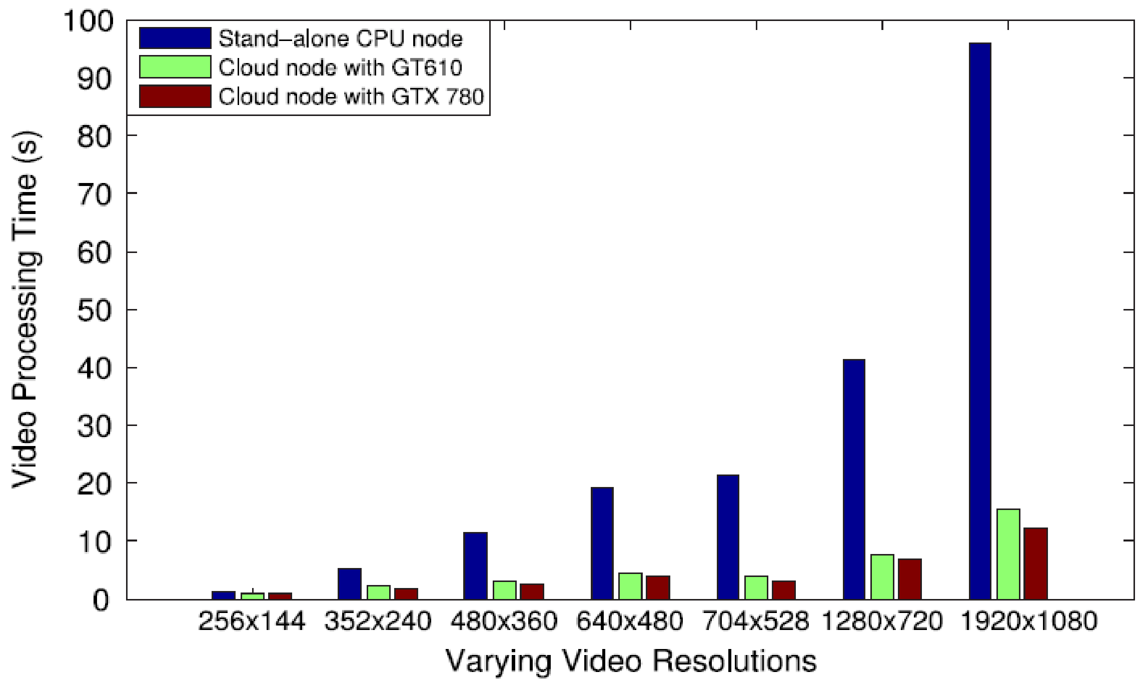


Figure 6.3: Processing Time with Varying Resolutions

6.3.3 Processing Time with Varying Resolutions

The processing time of a video frame is highly dependent on the resolution of a video frame. For a high resolution video frame, more computation time is required as more data is needed to be processed as shown in Figure 6.3. We have tested different video streams with varying resolutions on the system and computed the total processing time. This includes the time required to process the frame as well as the video decoding time.

6.3.4 Video Object Classification

After training the classifier, it is further used to perform classification of objects from the video streams. The marked or target object which is to be identified from the video streams is passed through the trained classifier after performing all the preprocessing steps on it to make it appropriate for the classifier as described in Chapter 5.

The classifier returns the probabilities of the possible labels but not the labels itself. The labels of all the objects present in all the video streams are already stored in the database beforehand. The classification process ends up in generating the probabilities of the matched objects. The

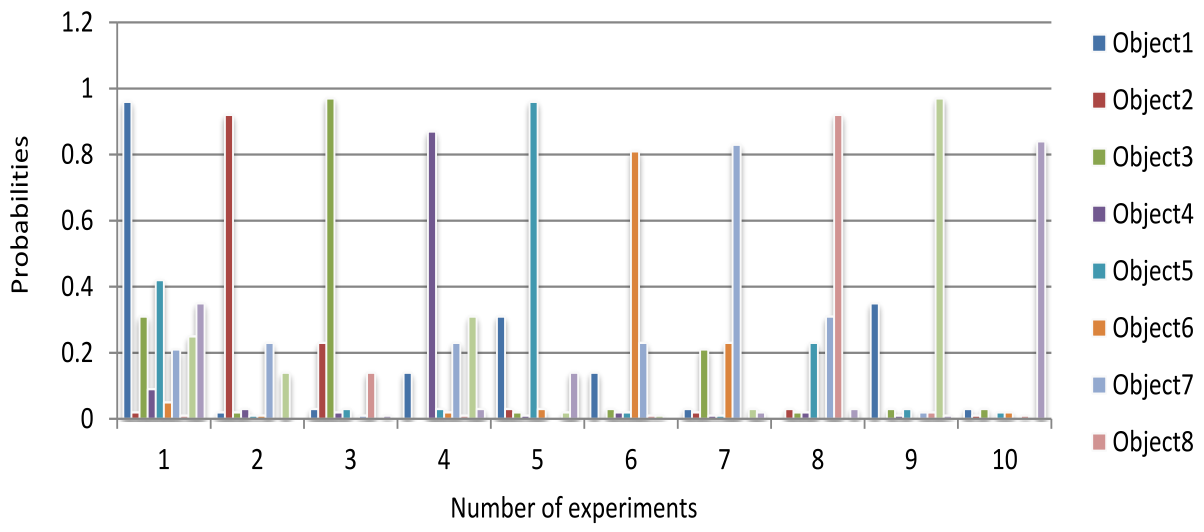


Figure 6.4: Classification of Marked Object

object with the highest probability indicates the classification of the desired object which is being searched from the video streams. Very low probabilities against all the objects indicate that the target object is not present in all of the video streams in the database.

Figure **??** depicts the probabilities of some of the objects generated by the classifier. The 10 experiments are represented on each index of the x-axis. Different probabilities generated by each experiment are represented on y-axis of the graph. The marked objects which were fed into the trained network are listed on the right hand side of the graph.

We have shown results from 8 different objects for this set of experiments. The probabilities generated by the classifier against each object are shown in different columns of the table. The probabilities near to 1 depict a closer match of marked object and the probabilities close to 0 depict the unavailability of objects in the video stream database.

In order to track a specific object from a number of video streams, we checked the generated probabilities or matching scores against each video stream. The trained classifier generates a high matching score against the marked object if its training instances is present in the database. We can, therefore, set an empirically determined threshold on the basis of the matching score. The matching scores greater than or equal to the threshold reveal that the object is present in

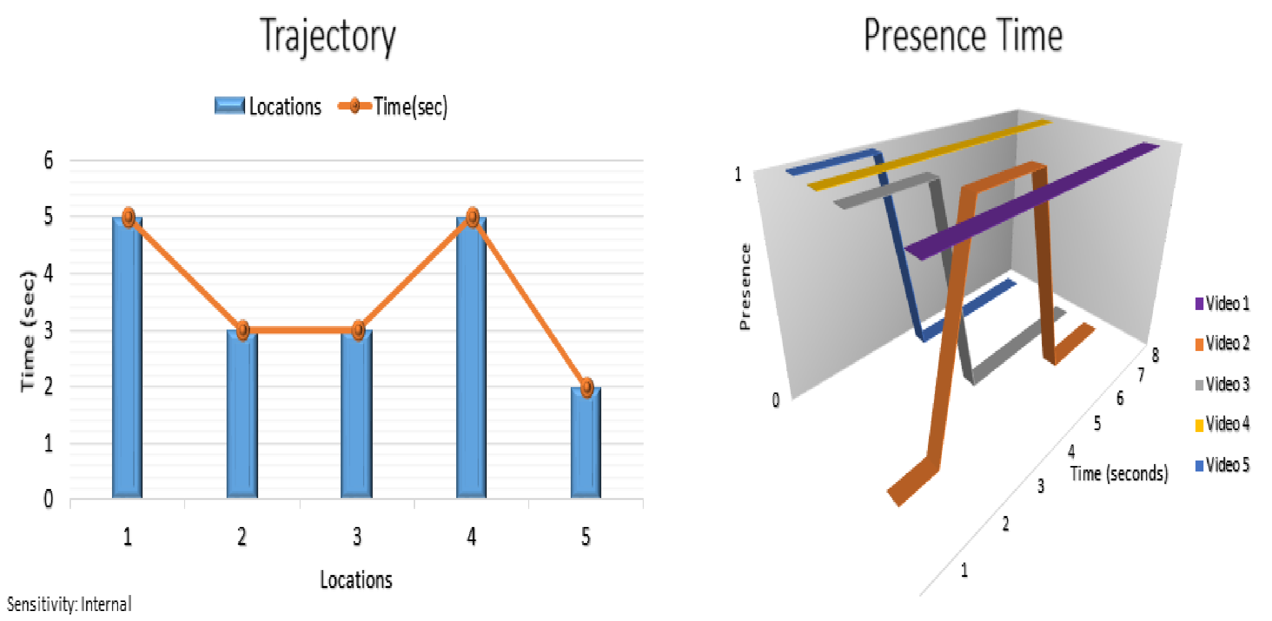


Figure 6.5: Object Characteristics

the specific video stream.

We have also measured the estimated presence time of the object in the video streams and recorded its location as well. The meta-data about the location of the video stream was stored in the database along with the video itself. Once an object is searched in a video stream, its location is also recorded through the meta-data.

Figure 6.5 shows the presence of the object and the trajectory in multiple video streams. The presence of object in the video stream is represented by 1 while its absence is represented by 0. It can be seen that the object remained present throughout in video 1 and video 4. However, in remaining videos it remained there for a fraction of time.

Figure 6.5 also provides a visual representation about the tracking of the object as it has been mapped on a graphical representation. This graphical representation is actually the summary of multiple videos in which the specific object has been detected. Each bar in the graph represents the amount of time that the marked object has spent at a specific location while its travel movement can be observed by the line graph.

6.4 Spatial-Frequency based Video Analytics Results

We first present and discuss the results of the proposed system on the basis of Amplitude, Phase and Orientation Properties. We then present the results of the feature fusion strategy of the proposed system. After that, we compare it with the two state-of-the-art models and measure the improvements in terms of Accuracy, Precision, Recall and F1 Score. A discussion on the performance characterization of the resultant confusion matrix in terms of True Positives, False Positives, True Negative and False Negative is also provided.

6.4.1 Performance Measures of Amplitude, Phase and Orientation Properties

We present the results of our proposed system in this sub-section and measure the performance of the classifier in terms of following performance characterizations: Accuracy, Recall, Precision and F1 score. We have calculated these performance measures for first three IMFs of amplitude, phase and orientation properties and made a comparison of these to evaluate the best performing properties. The property which contributes much in improving the accuracy of the classifier is then further used for fusion.

The number of epochs during the training of the classifier have been varied from 5 to 40. We have also calculated the training time of classifier for each epoch to have an estimate of the total training time. A detailed discussion of all the results is presented with the help of a confusion matrix. To show the efficacy of the proposed classifier, it has been compared with two most famous state-of-the art CNN models.

Table. 6.1 shows the performance of the classifier on the amplitude property. The Accuracy, Recall, Precision and F1 scores are tabulated for the first three IMFs. The number of epochs has been varied from 5 to 40. It can be seen from the table that the amplitude property could not perform better and remained unable to classify the test patterns. Even with the increasing number of epochs, it remained unable to improve the accuracy of the classifier significantly.

Table 6.1: Performance Measures of Amplitude Property

IMF1 Amplitude					**IMF2 Amplitude**				**IMF3 Amplitude**			
Epochs	**Acc**	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**
5	0.0191	0.0088	0.0191	0.0121	0.0206	0.0117	0.0206	0.0149	0.0162	0.0051	0.0162	0.0878
10	0.0147	0.0087	0.0147	0.0109	0.0235	0.0149	0.0235	0.0183	0.0235	0.0105	0.0235	0.0145
15	0.0162	0.0068	0.0162	0.0096	0.025	0.1108	0.025	0.0408	0.0279	0.0887	0.0279	0.0425
20	0.0294	0.1538	0.0294	0.0494	0.025	0.0119	0.025	0.0161	0.0309	0.057	0.0309	0.0401
25	0.0441	0.2107	0.0441	0.073	0.0279	0.0217	0.0279	0.0244	0.0324	0.0363	0.0324	0.0342
30	0.0588	0.2448	0.0588	0.0949	0.0471	0.152	0.0471	0.0719	0.0324	0.1225	0.0324	0.0512
35	0.075	0.2534	0.075	0.1157	0.0662	0.2121	0.0662	0.1009	0.0324	0.0396	0.0324	0.0356
40	0.0941	0.317	0.0941	0.1451	0.0794	0.1992	0.0794	0.1136	0.0338	0.0393	0.0338	0.0364

Table 6.2: Performance Measures of Phase Property

IMF1 Phase					**IMF2 Phase**				**IMF3 Phase**			
Epochs	**Acc**	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**
5	0.0485	0.0569	0.0485	0.0524	0.0868	0.279	0.0868	0.1324	0.1191	0.3468	0.1191	0.1773
10	0.0912	0.3333	0.0912	0.1432	0.2441	0.6559	0.2441	0.3558	0.2353	0.4541	0.2353	0.31
15	0.1868	0.6105	0.1868	0.286	0.4721	0.7504	0.4721	0.5796	0.2971	0.4449	0.2971	0.3562
20	0.3309	0.7582	0.3309	0.4607	0.6412	0.7871	0.6412	0.7867	0.3338	0.4481	0.3338	0.3826
25	0.4588	0.7824	0.4588	0.5784	0.725	0.81	0.725	0.7651	0.3559	0.4513	0.3559	0.3979
30	0.5838	0.8142	0.5838	0.68	0.7691	0.8225	0.7691	0.7949	0.3853	0.4785	0.3853	0.4269
35	0.6676	0.8135	0.6676	0.7334	0.8015	0.835	0.8015	0.8179	0.4132	0.5164	0.4132	0.4591
40	0.7235	0.8132	0.7235	0.7658	0.8191	0.8436	0.8191	0.8312	0.4235	0.5307	0.4235	0.4711

Table 6.3: Performance Measures of Orientation Property

IMF1 Orientation					**IMF2 Orientation**				**IMF3 Orientation**			
Epochs	**Acc**	**Precision**	**Recall**	**F1**	**Acc**	**Precision**	**Recall**	**F1**	**Acc**	**Precision**	**Recall**	**F1**
5	0.1368	0.2477	0.1368	0.1762	0.2397	0.3485	0.2397	0.284	0.1574	0.3061	0.1574	0.2079
10	0.3015	0.6058	0.3015	0.4026	0.5838	0.6945	0.5838	0.6344	0.2956	0.4697	0.2956	0.3628
15	0.5162	0.747	0.5162	0.6105	0.7544	0.8146	0.7544	0.7834	0.4044	0.5224	0.4044	0.4559
20	0.6868	0.8166	0.6868	0.7461	0.8426	0.8668	0.8426	0.8546	0.4529	0.5561	0.4529	0.4992
25	0.7882	0.8542	0.7882	0.8199	0.8786	0.8974	0.8786	0.8838	0.4985	0.5641	0.4985	0.5293
30	0.825	0.8983	0.825	0.8601	0.9015	0.9212	0.9015	0.9112	0.5309	0.5734	0.5309	0.5513
35	0.8426	0.9054	0.8426	0.8729	0.925	0.9377	0.925	0.9313	0.5426	0.5857	0.5426	0.5633
40	0.8441	0.9047	0.8441	0.8734	0.9338	0.9414	0.9338	0.9376	0.5765	0.6116	0.5765	0.5935

The same trend of performance was observed for Precision, Recall and F1 score for all the epochs. We believe that the reason for such a poor performance is due to the scarce availability of the data points in the amplitude property. The amplitude property does not provide significant number of data points required to perform an accurate classification. With the reduction in the noisy components, it also discards the useful data points which could aid in performing the accurate classification. This results in the drop of the overall accuracy rate of the classifier.

Table. 6.2 shows the performance of the classifier on the phase property. The Accuracy, Recall, Precision and F1 scores are again calculated for first three IMFs with the number of epochs varying from 5 to 40. It can be seen from the table that the phase property performed much better to classify the test patterns as compared to the amplitude property. The increasing number of epochs improved the accuracy of the classifier. Especially after 20 epochs the accuracy improved significantly and at epoch number 40, it reached to an accuracy of 0.72 percent.

Improved performance rates were observed for precision, recall and F1 scores as well. One reason could be the availability of much higher number of data points for phase property as compared to the amplitude property. Although the phase property for third intrinsic mode function does not contain enough data points but first two remained enough to have a descent classification rate.

The performance of the classifier on the orientation property is depicted in Table 6.3. The same performance measures Accuracy, Precision, Recall and F1 scores are tabulated for the first three IMFs of orientation property. The number of epochs are again varied from 5 to 40 for these set of experiments as well.

A significant amount of improvement in the overall accuracy rate of the classifier has been observed as compared to both amplitude and phase properties. Even at epoch number 10 the classification accuracy started at a reasonable rate of 0.3015 as compared to amplitude and phase and kept on improving to 0.84 till the 40th epoch.

We figured out that these improvements are due to the maximum availability of data points and minimum amount of presence of noisy frequencies in the orientation property. The third

Table 6.4: Orientation Fusion

Orientation Fusion				
Epochs	**Accuracy**	**Precision**	**Recall**	**F1 Score**
5	0.1897	0.5317	0.1897	0.2796
10	0.5338	0.7673	0.5338	0.6296
15	0.8044	0.8611	0.8044	0.8318
20	0.9103	0.9272	0.9103	0.9187
25	0.9544	0.9605	0.9544	0.9574
30	0.9676	0.9705	0.9676	0.9691
35	0.9779	0.9791	0.9779	0.9785
40	0.9794	0.9806	0.9794	0.98

intrinsic mode function kept reasonable number of data points which contributed towards the improvements in higher accuracy rates.

6.4.2 Performance Measures of Orientation Fusion

Inspired by this fact, we have performed a feature fusion strategy on the orientation property to further improve the accuracy rates. The feature fusion strategy is performed on the first two intrinsic mode functions of the orientation property. The two intrinsic mode functions are fused together in order to have a composite intrinsic mode function which could hold the properties of both the IMFs. The fused IMF is a numeric matrix which represents the combined orientation property and is used for classification.

Table. 6.4 shows the performance of the classifier on the fused intrinsic mode function. The number of epochs are varied from 5 to 40 and Accuracy, Precision, Recall and F1 scores are recorded and tabulated in the table. It can be seen that the overall accuracy rate of the fused IMF is greater than all the previous accuracies of the amplitude, phase and orientation properties. From epoch number 10, the classification accuracy was recorded to be 0.53 which is significantly better than the amplitude and phase and orientation.

The accuracy kept on improving to 0.97 till the 40th epoch. Improvements were observed for Precision, Recall and F1 scores as well. The Precision was recorded to be 0.9806 at 40th epoch. Similarly, the Recall and F1 scores were recorded to be 0.97 and 0.98 respectively. We believe that these improvements are due to the further addition of data points in the orientation property and reduction in the presence of noisy frequencies. The fused intrinsic mode function contains significant information in terms of data points which leads to further improvements in higher accuracy rates.

It is also to be noted that the dataset used in the experiments to validate the proposed system is mostly self-generated consisting of videos of human faces (head-and-shoulder) of various individuals. The video streams recorded for the experiments are relatively simple (captured under controlled and uncontrolled environmental conditions with faces posing towards a camera) and does not pose challenges such as occlusion or head pose rotation. The simplicity of the dataset played a major role in achieving high performance and accuracy rates of 95-98 percent. However, other factors such the use of spatial-frequency domain features and tuning of hyper-parameters also played a major role in achieving the reported performance and accuracy rates.

6.4.3 Execution Time and Confusion Matrix

The execution times for different number of epochs varying from 5 to 40 are depicted in Table 6.5. The execution time is directly dependent on the number of epochs. An increase in the number of epochs increases the execution time as well. It takes between 12 to 15 mins to complete 5 epochs by the system for each intrinsic mode function. The execution time doubles by doubling the number of epochs.

For 10 epochs the execution time increases between a range of 24 to 30 minutes for different intrinsic mode functions. The decision about the total number of epochs for training depends on the experimentation. There is always a trade-off between the performance and the execution time of the system. For our proposed system, 40 epochs provided a good accuracy rate in a reasonable amount of execution time.

Table. 6.6 shows the overall performance of the proposed system with the help of a confusion matrix in terms of Accuracy, Precision, Recall and F1 score. The overall accuracy of the system with the fused features is recorded to be 0.9794. The confusion matrix depicted the precision

Table 6.5: Execution Time (Mins)

	Amplitude			**Orientation**			**Phase**		
Epochs	**IMF 1**	**IMF 2**	**IMF 3**	**IMF 1**	**IMF 2**	**IMF 3**	**IMF 1**	**IMF 2**	**IMF 3**
5	13	15	13	12	15	13	12	14	12
10	24	30	25	25	29	27	24	28	25
15	37	44	42	43	44	36	36	44	37
20	49	48	52	50	49	49	49	49	48
	61	74	60	62	72	62	64	70	61
30	74	82	73	75	88	78	74	72	73
35	89	84	94	86	98	86	86	85	85
40	101	98	99	99	101	98	102	94	99

Table 6.6: Confusion Matrix

Classification Scores
Accuracy: 0.9794
Precision: 0.9806
Recall: 0.9794
F1 Score: 0.98
{0=[0 x 19, 18], 1=[1 x 20], 2=[16, 2 x 18, 26], 3=[3 x 19, 31], 4=[4 x 20], 5=[5 x 19, 13], 6=[6 x 20], 7=[7 x 20], 8=[8 x 19, 30], 9=[9 x 20], 10=[16, 10 x 19], 11=[11 x 19, 27], 12=[12 x 20], 13=[26, 13 x 19], 14=[14 x 20], 15=[15 x 20], 16=[16 x 20], 17=[17 x 20], 18=[18 x 20], 19=[19 x 20], 20=[20 x 20], 21=[20, 21 x 19], 22=[22 x 20], 23=[23 x 20], 24=[24 x 20], 25=[25 x 20], 26=[26 x 20], 27=[27 x 20], 28=[28 x 20], 29=[24, 29 x 19], 30=[30 x 20], 31=[12 x 2, 31 x 18], 32=[32 x 20], 33=[33 x 19, 7]} {0=1, 1=0, 2=2, 3=1, 4=0, 5=1, 6=0, 7=0, 8=1, 9=0, 10=1, 11=1, 12=0, 13=1, 14=0, 15=0, 16=0, 17=0, 18=0, 19=0, 20=0, 21=1, 22=0, 23=0, 24=0, 25=0, 26=0, 27=0, 28=0, 29=1, 30=0, 31=2, 32=0, 33=1} {0=0, 1=0, 2=0, 3=0, 4=0, 5=0, 6=0, 7=1, 8=0, 9=0, 10=0, 11=0, 12=2, 13=1, 14=0, 15=0, 16=2, 17=0, 18=1, 19=0, 20=1, 21=0, 22=0, 23=0, 24=1, 25=0, 26=2, 27=1, 28=0, 29=0, 30=1, 31=1, 32=0, 33=0} {0=19, 1=20, 2=18, 3=19, 4=20, 5=19, 6=20, 7=20, 8=19, 9=20, 10=19, 11=19, 12=20, 13=19, 14=20, 15=20, 16=20, 17=20, 18=20, 19=20, 20=20, 21=19, 22=20, 23=20, 24=20, 25=20, 26=20, 27=20, 28=20, 29=19, 30=20, 31=18, 32=20, 33=19}, {0=660, 1=660, 2=660, 3=660, 4=660, 5=660, 6=660, 7=659, 8=660, 9=660, 10=660, 11=660, 12=658, 13=659, 14=660, 15=660, 16=658, 17=660, 18=659, 19=660, 20=659, 21=660, 22=660, 23=660, 24=659, 25=660, 26=658, 27=659, 28=660, 29=660, 30=659, 31=659, 32=660, 33=660}

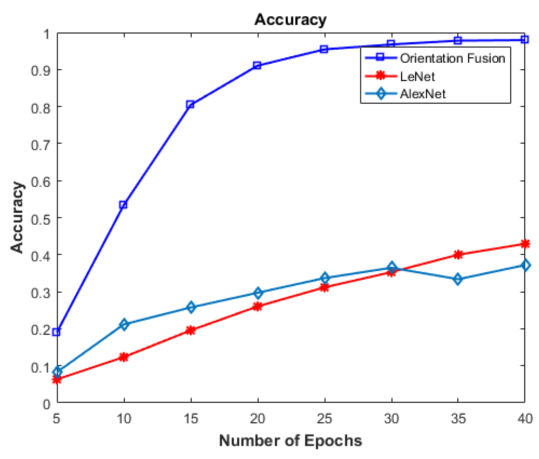


Figure 6.6: Accuracy

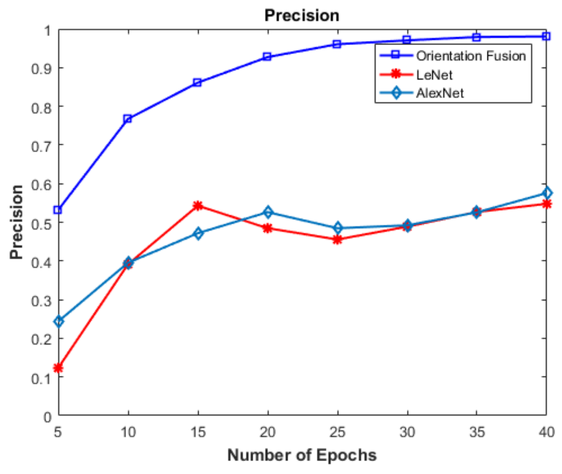


Figure 6.7: Precision

of the system to be 0.9806 which shows that the proposed system is accurate as well as precise. The Recall and F1 scores of the system observed to be 0.9794 and 0.98 respectively.

Most of the test samples from all the subjects were classified correctly by the classifier as depicted in the confusion matrix. There were few samples from some subjects that were miss-classified. In most of the cases only one sample has been miss-classified. The samples labeled as yaleB03 are miss-classified two times as yaleB17 and yaleB27. Also the samples labeled as yaleB32 was miss-classified as yaleB33 two times. We believe that this miss-classification is due to the severe illumination effect in the test samples.

6.4.4 Comparison with AlexNet and LeNet

We have also compared the performance of the proposed system with the state-of-the-art well known deep learning models AlexNet [55] and LeNet [165]. Figures 6.6, 6.7, 6.8 and 6.9 demonstrate and compare the performance improvements of the proposed system with the AlexNet and LeNet models. As it can be seen from Figure 6.6 that the proposed orientation fusion approach provides much higher accuracy rates as compared to AlexNet and LeNet.

From the very start, at iteration number 5, the accuracy of the system is recorded to be 0.2 while the accuracy of AlexNet and LeNet was below 0.1. The accuracy kept on improving with the increasing number of epochs. At epoch number 25, a significant improvement can be

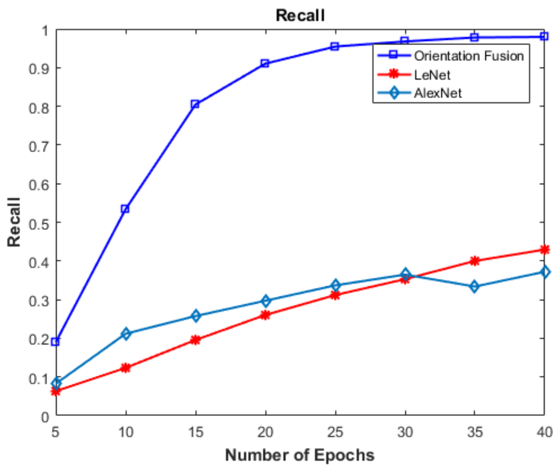


Figure 6.8: Recall

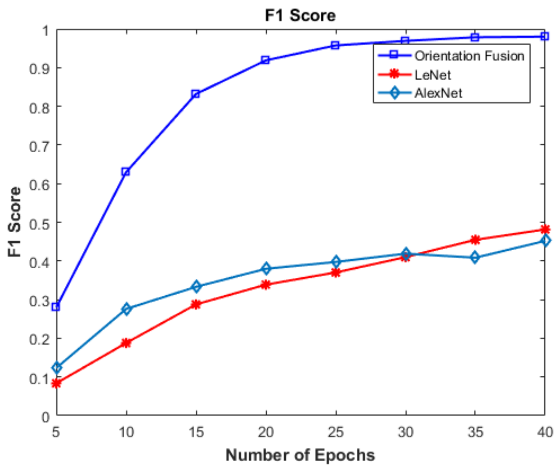


Figure 6.9: F1 Score

observed in the graph as compared to the other two models which kept on improving till the last epoch.

A similar kind of behavior can be observed in the precision, recall and F1 score graphs in Figures 6.7, 6.8 and 6.9. The precision of the system started from 0.55 from epoch 5 and showed a linear improvement over increasing number of epochs. Rapid improvements have been observed till epoch number 25 in the precision curve which kept on improving gradually till the last epoch.

The recall and F1 score curves depict a similar trend in their curves and shows significant improvements as compared to AlexNet and LeNet. The AlexNet performed a bit better than the LeNet till epoch number 30 but it could not get above 0.3. A drop in the recall curve for AlexNet has been observed after 30 epochs. The same trend was observed for F1 score curve.

We have also compared the execution time of the proposed system with the execution times of existing models. Figure 6.10 shows the execution time of the proposed system and the two models. It can be seen that the AlexNet model takes much more time for execution as compared to the proposed system. The reason behind this is the complexity of the AlexNet model which constitutes of 12 layers. The proposed orientation fusion approach and the LeNet model takes almost the same amount of time as the architecture of both the models is quite similar.

The proposed orientation fusion approach showed significant improvements over the two models on a challenging dataset. The images present in the publically available Yale Face Database

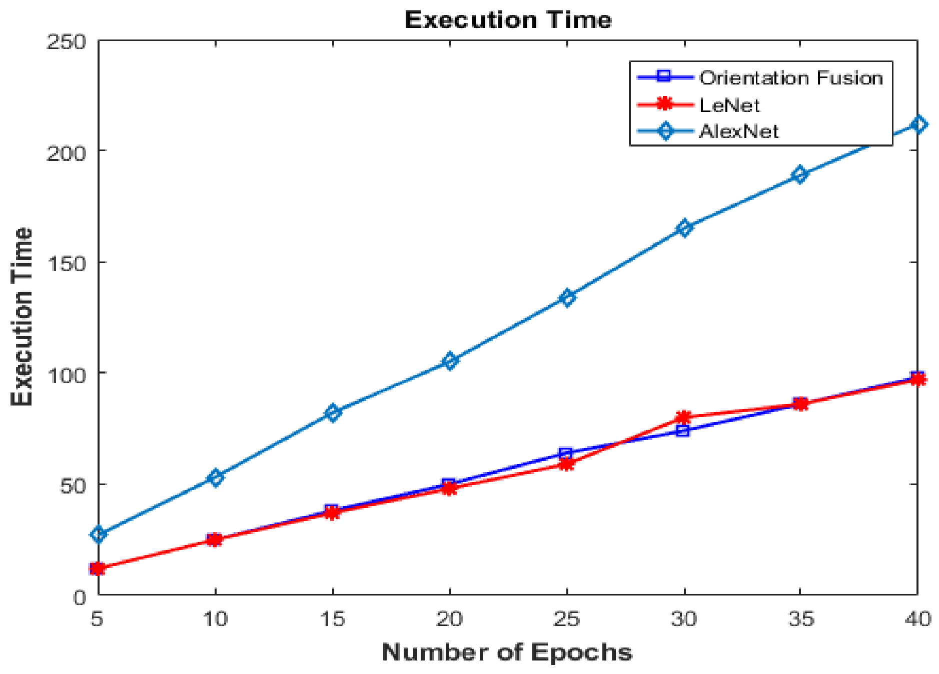


Figure 6.10: Execution Time

has significant variations in expressions, pose and illumination conditions. The illumination conditions imposes effects from different angles including left-right, center-right and right-right.

We believe that the reason behind these improvements is that the illumination effects mostly resides in the low frequency components of the spectrum. The empirical mode decomposition separates the images into individual intrinsic mode functions in the decreasing order. It then becomes easier to discard the low frequency components from the image and retain only the high frequency components. The fusion of the two intrinsic mode functions containing the highest frequencies is sufficient enough to correctly classify most of the training samples with high accuracy rate and precision.

The large difference with respect to the state-of-the-art in Figures 6.6-6.9 is also due to the fact that the training data required to train the state-of-the-art deep network is quiet large. This is due to the number of layer and parameters involved in the architecture of the deep network. In our experiments we have trained them with fewer data as compared to data on which these networks were originally trained. The size of the proposed network in this work is small as compare to the state-of-the-art deep network. The use of the spatial-frequency domain features enabled the system to achieve high accuracy and less training time even with a small sized deep

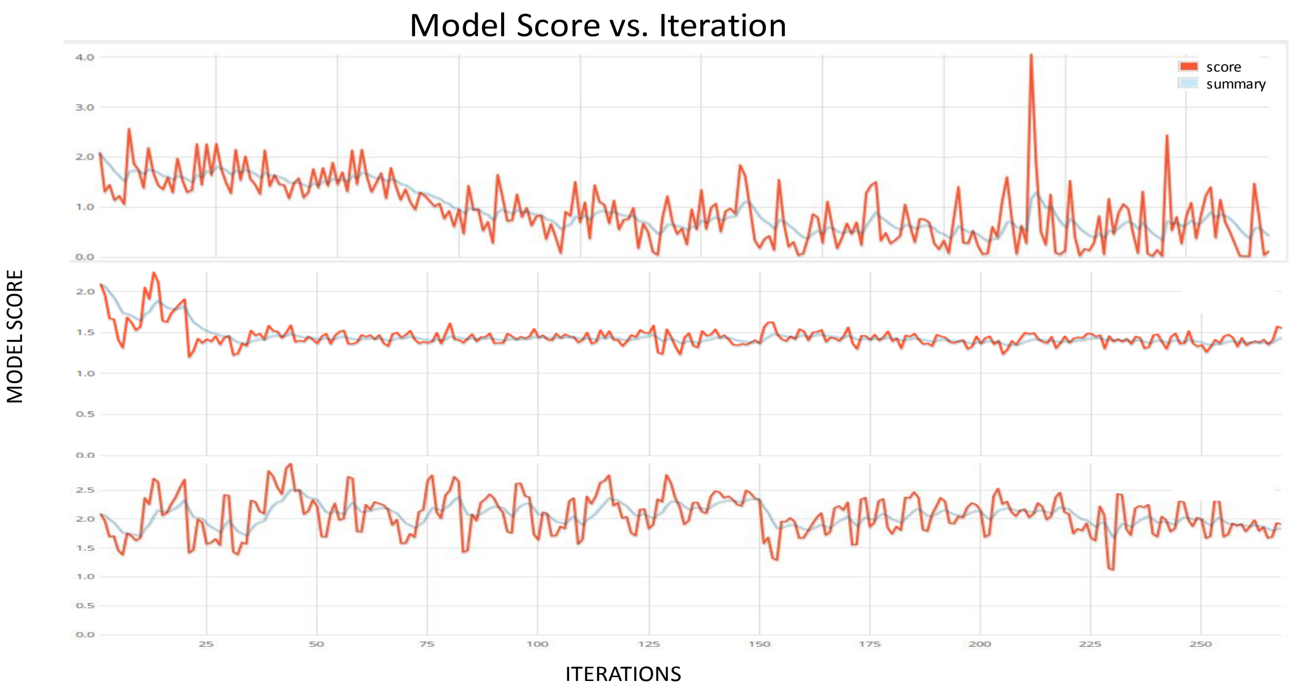


Figure 6.11: Model Scores

network.

6.5 Deep Learning based Video Analytics Results

We present and discuss the results of our deep learning based video analytics system in this section. We first analyze the results generated by tuning the hyper-parameters of deep model to various values and propose the parameters which could potentially produce best results. The trained system on the proposed parameters is then evaluated with different performance characterization.

6.5.1 Hyper-parameter Tuning

There are a number of parameters which can be tracked during the training of a deep network. These parameters provide intuitions about the settings of different hyper-parameters and help to decide that whether the setting should be changed in order to have more efficient learning. The parameters are tracked and represented in the form of graphs over multiple time stamps

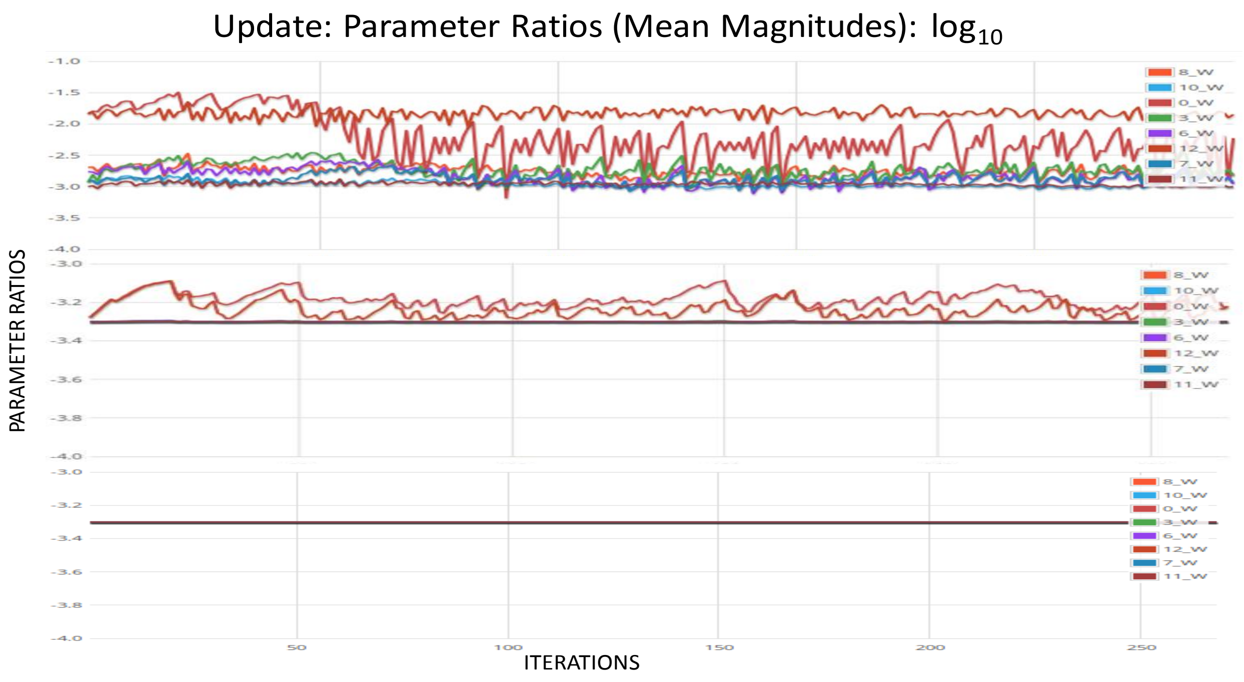


Figure 6.12: Parameter Ratios

in order to observe the trend in the behavior of the system.

The x-axis of the plot in Figure 6.11 represents iterations and the number of iterations depends on the settings of batch size. While the loss function value $L(x) = LR\sum_{x_i->X}\sum_{x_i->T_i} l(i, x_iT)$ of current mini-batch is depicted on the y-axis of the plot in Figure 6.11. The loss function value is evaluated during the forward pass of the back-propagation on the individual batches. The gray line in the graph represents the running average of the loss on each iteration. It gives a better visualization to analyze the trend in the graph of the loss function.

It can be seen in the first graph of figure that it goes down after each iteration over time, depicting that the learning rate is properly tuned. The learning rate is tuned on the basis of experimentations until the score moves towards stability. We varied the value of LR to different values including 1e-2, 1e-4 and 1e-6. The effects of these values of LR on $L(x)$ are plotted in Figure 6.11 and it can be seen that 1e-2 proved to be a good learning rate .

The decreasing trend of the graph is also an indication that the training data is normalized properly. L2 normalization scheme with stochastic gradient descent $W_{t+1} = W_t - \alpha\delta L(\theta_t)$ is the most appropriate approach for our network training. α is the learning rate which has been varied on the above mentioned values w is the weight change with respect to the gradient of

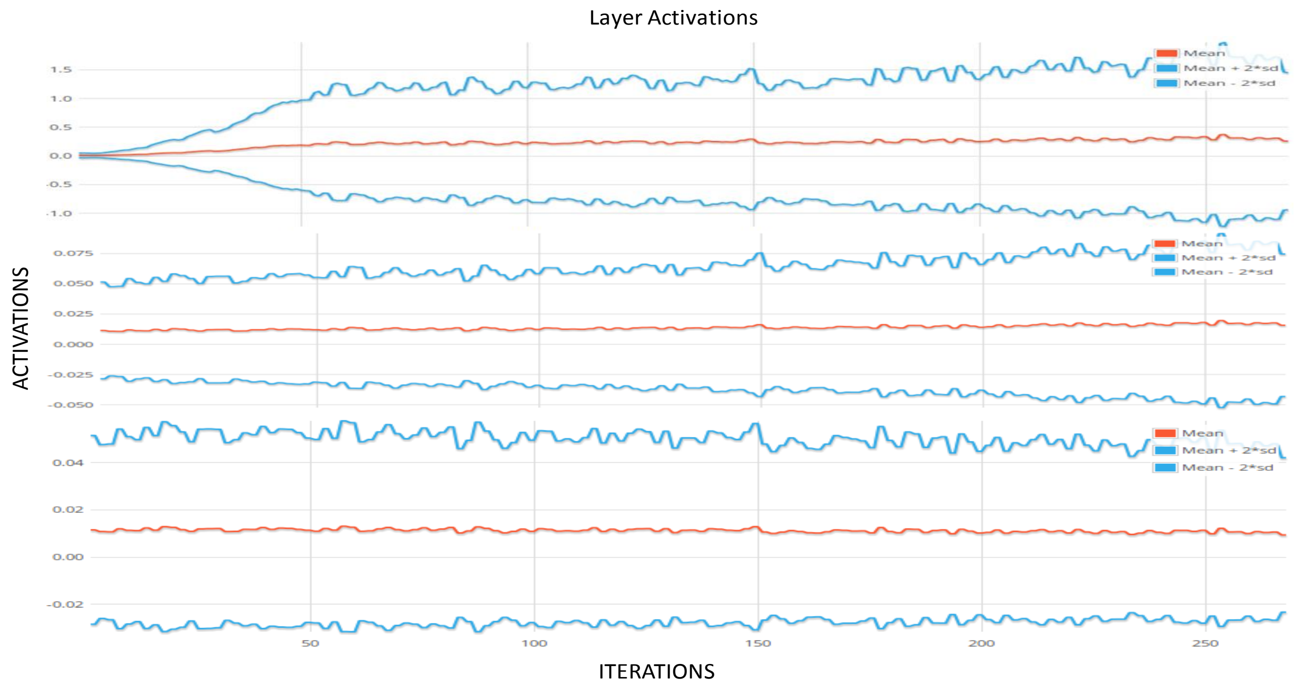


Figure 6.13: Layer Activations

the loss function. The weights are initialized at random for all the experiments. The selected gradient descent approach does not let the score to increase, which normally happens if the learning rate is set too high.

The bottom two graphs do not show a proper decreasing trend over multiple iterations. These graphs were produced with a LR of 1e-4 and 1e-6 respectively. It can be seen clearly that the graphs follow a stable state over the iterations and do not show a decreasing trend for learning rate values of 1e-4 and 1e-6. Both the graphs do not fall below 1.0 of the y-axis. The last graph with a learning rate of 1e-6 does not even fall below of 1.5 on the y-axis and is the depiction of bad learning rate, normalization and regularization schemes.

Another important parameter which can be used to track in order to have an intuition about the efficient learning of the system is the ratio of weights (updates). It is not beneficial to track the raw gradients but the updates of the weights. It can also be helpful to track this ratio for every set of parameters. Figure 6.12 shows the ratio of mean magnitudes of the parameters which is the average value of parameters at various iterations is shown along the horizontal axis.

It is suggested that the ratio of parameters at various time stamps of iterations should be

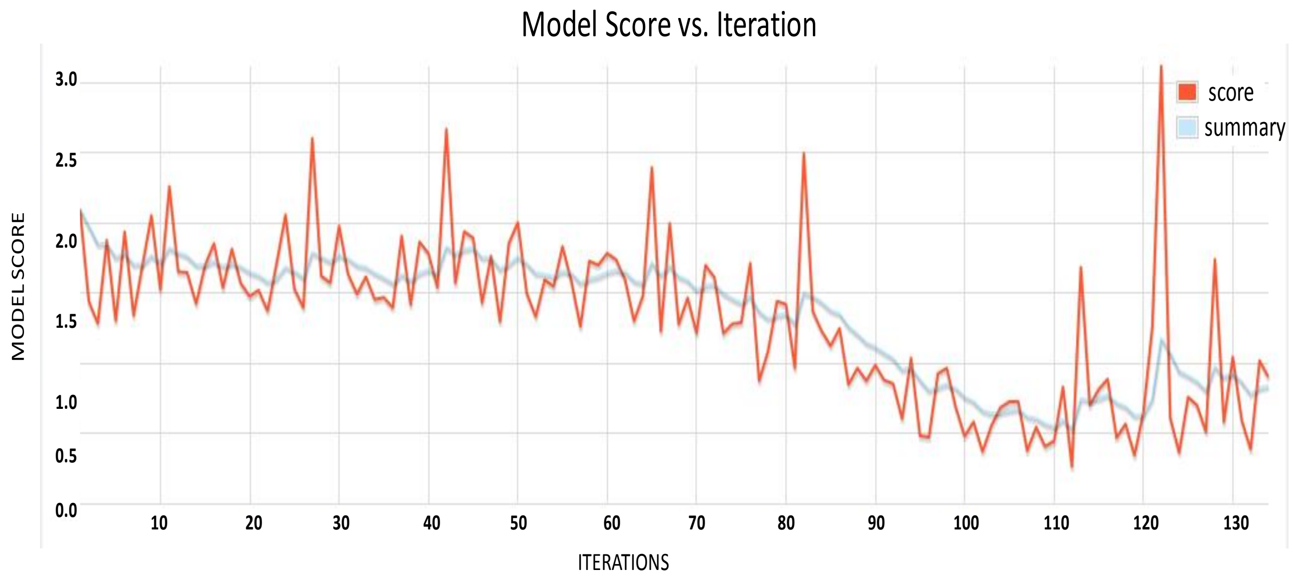


Figure 6.14: Model Scores at Various Iterations

around -3 on a log10 chart. This is an indicative of a good LR and appropriate initialization of other network hyper-parameters. A high divergence of this ratio from the specified value is an indication of unstable parameter initialization and selection. This means that the parameters are unable to learn appropriate features from the training dataset.

It can be seen from the first graph of figure that the parameters at various time stamps of iterations are around -3 on the log10 chart. Some parameters started from -2.5 on the y-axis but a convergence towards -3.0 can be easily observed from the figure. Especially 0_W tends to converge very rapidly towards -3.0 after some time stamps of iterations.

Figure 6.13 shows the mean magnitudes of the parameter ratios of first Convolutional layer used in the network. It can be seen from the layer activations graph that the graph stabilizes after almost 80 iterations which depicts that the network is stable and is not prone to exploding activations problem. The stability of the layer activation graph also shows that the weights of the layers have been initialized correctly with proper regularization scheme.

It is observed that the graph stabilizes after few iterations depicting that the model can cope the problem of vanishing or exploding activations. The stability of the graph after some iterations also shows that the weights of the layers have been initialized correctly and proper regularization scheme i.e. $\lambda_2 \sum_i \theta_i^2$ is adopted. The value of λ is varied from 5 * 1e-2 to 5 * 1e-8 but the value of 5 * 1e-4 provided the best results. Please note that this is the ratio for the first Convolution layer of the network.

It can be seen that the activations at various time stamps of iterations are between the suggested region. This shows that network is in a good learning state from the very first layer with proper learning rate and other network hyper-parameters. The convergence of this ratio as seen from the chart is an indication of stable parameter initialization and selection. On the other hand the bottom two graphs do not show a stability trend.

6.5.2 Deep Learning Model Training on Tuned parameter values

We have trained the system on the proposed hyper-parameters for our video object classification pipeline and evaluated the performance. Figure 6.14 shows the value of loss function at various iterations on the current minibatch. The graph is drawn against training scores of the network and training iterations. It can be seen that the graph converges which shows that the learning rate $LR = 0.0001$ is a well selected learning rate.

The decreasing trend of the graph is also an indication that "L2 normalization scheme $\lambda_2 \sum_i \theta_i^2$" with "SGD $W_{t+1} = W_t - \alpha\delta L(\theta_t)$" is a good approach for the training of our network. A bit of a noise in the graph is observed but it is very low variation in a small range and is not an indicative of poor convergence of learning.

Figure 6.15 shows the ratio of mean magnitudes of the parameters which is the average value of parameters at various iterations shown along the horizontal axis. It is suggested that the ratio of parameters at various time stamps of iterations should be between -3.0 and -4.0 on a log10 chart. This is an indicative of a good LR and appropriate initialization of other network hyper-parameters.

A high divergence of this ratio from the specified range is an indication of unstable parameter initialization and selection. This means that the parameters are unable to learn appropriate features from the training dataset. It can be seen from the figure that the parameters of all layers at various time stamps of iterations are between the specified range.

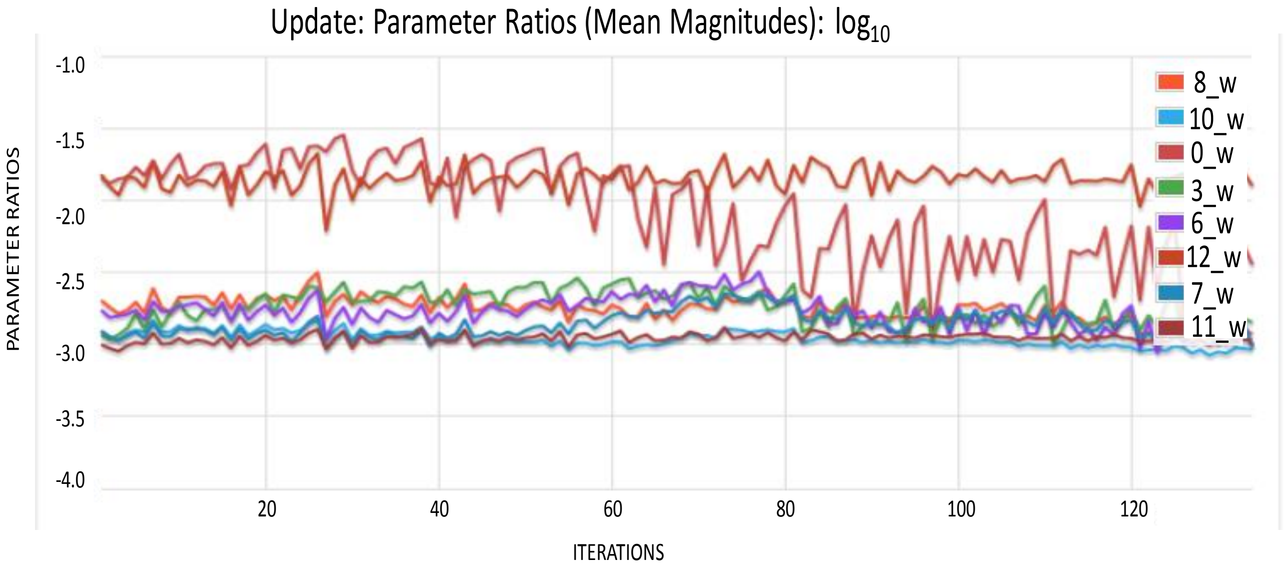


Figure 6.15: Parameter Ratios at Various Iterations

Figure 6.16(a), 6.16(b) and Figure 6.16(c) show the standard deviations of layer activations, gradients and updates of parameters. A stable trend is observed in these graphs which show that the system is capable of coping with the problem of vanishing or exploding activations. It also shows that the weights of the layers have been well selected and regularization scheme is properly adopted.

The histogram of layer parameters and layer updates are depicted in Figure 6.17 and Figure 6.18 respectively. A normal gaussian distribution is observed in the histograms of layer parameters. An approximate gaussian distribution in the histogram of weights for different layers show that the weights have been initialized correctly, updating in each iteration and there is sufficient regularization in the network.

An approximate gaussian distribution is also observed in the histogram of layer updates. These updates are the gradients which are generated after applying the regularization, momentum and learning rate. The momentum is given by $V_{t+1} = \rho v_t - \alpha \delta L(\theta_t)$ and the value of ρ is varied to 0.6, 0.8 and 0.9 for the generated results. The value is finally set to 0.9 in the above graph. Similar to the layer parameters histogram, an approximate gaussian distribution in the layer updates histogram represents that the network is not prone to exploding gradient problem. This is mainly because of the usage of gradient normalization which we added in the network.

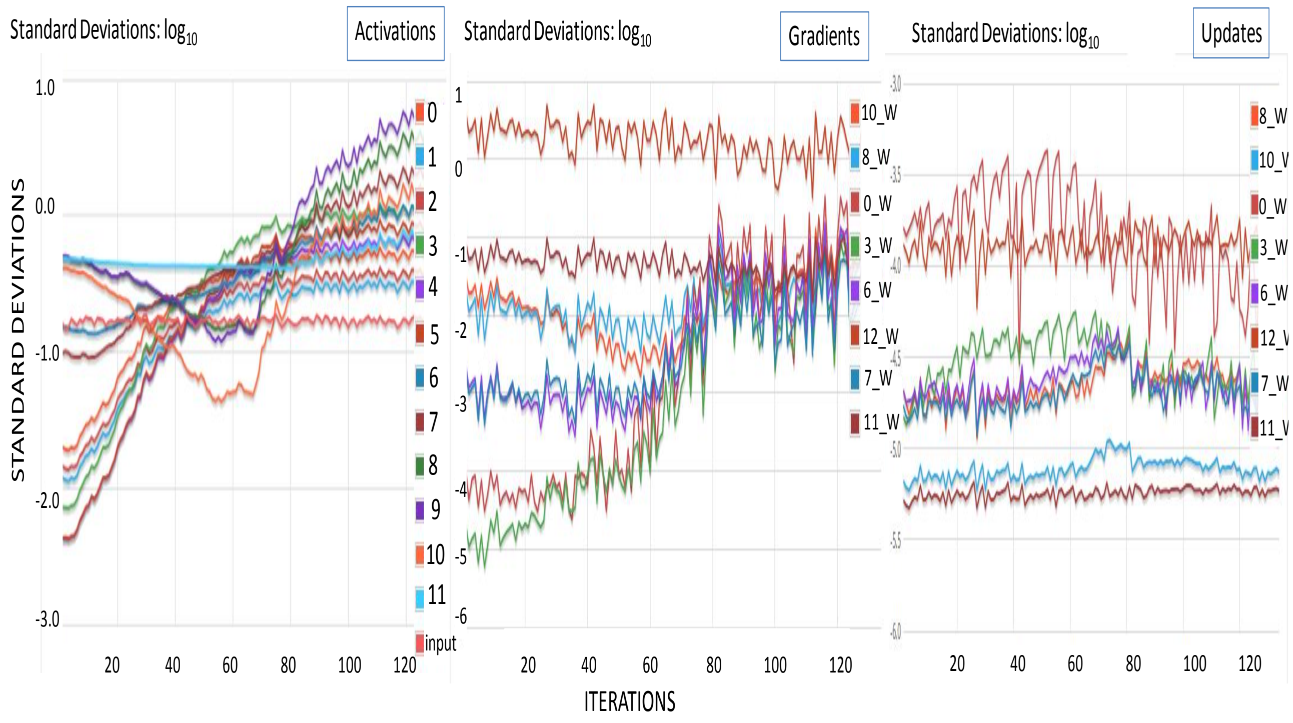


Figure 6.16: Standard Deviations of activations and Parameter Updates

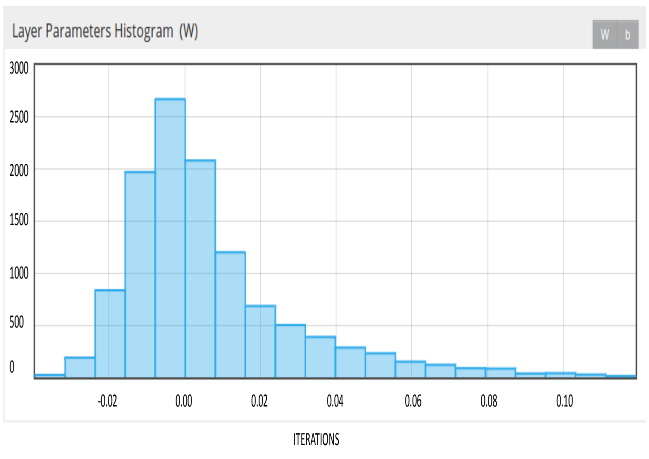


Figure 6.17: Histogram of Layer Parameters

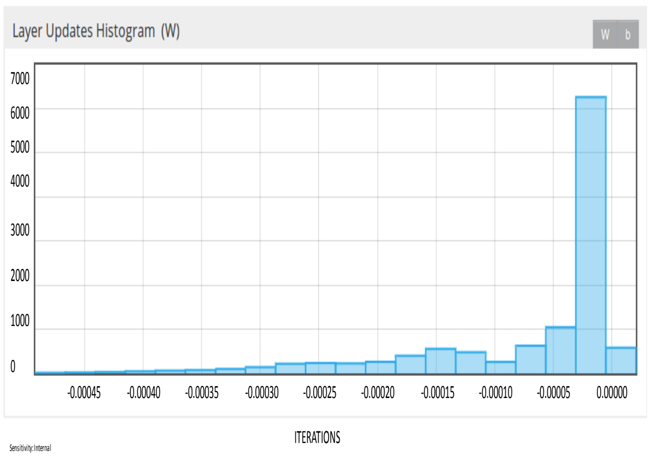


Figure 6.18: Histogram of Layer Updates

Figure 6.19 depicts the parameter ratios of first layer. The layer information is also shown alongside the graph in Figure 6.20 . It can be observed that the convolution layer achieves almost a stabilized state after 80 iterations. Ratios are shown for both the weights and biases of the layer. It can be seen that the ratio is between the suggested region i.e. 0.0 to -3.0 on a log10 chart.

Figure 6.21(a), Figure 6.21(b) and Figure 6.21(c) show the standard deviations of layer activations, gradients and updates of parameters for the first Convolution layer of the network. The

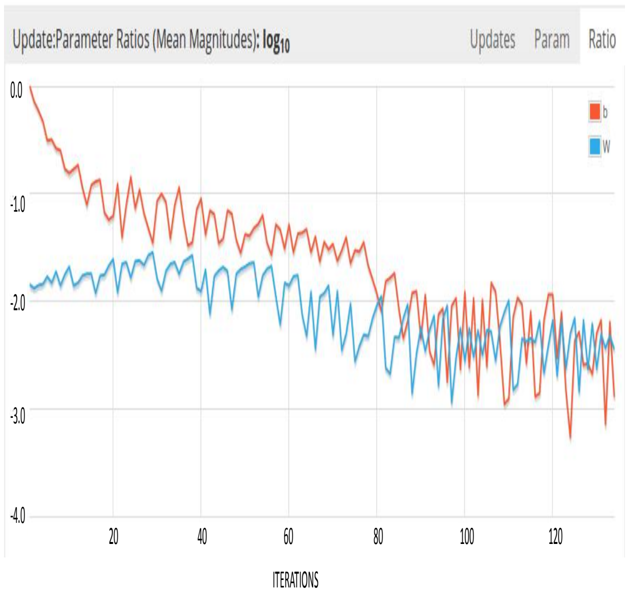
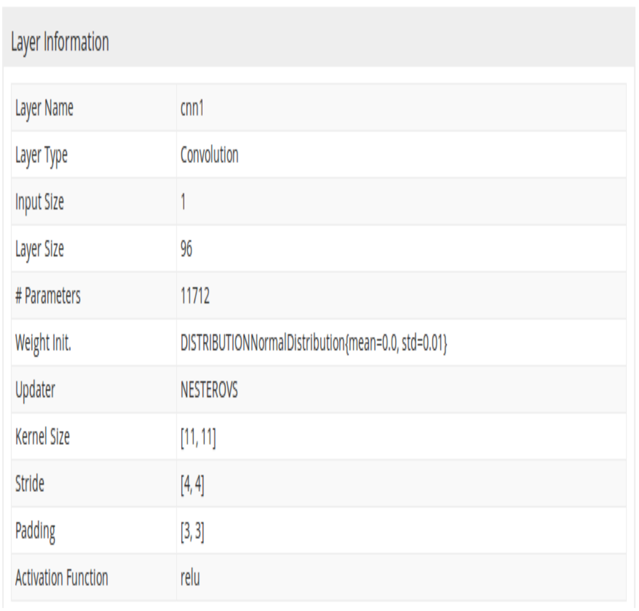


Figure 6.19: Parameter Ratios of First Layer


Layer Information

Layer Name	cnn1
Layer Type	Convolution
Input Size	1
Layer Size	96
# Parameters	11712
Weight Init.	DISTRIBUTIONNormalDistribution{mean=0.0, std=0.01}
Updater	NESTEROVS
Kernel Size	[11, 11]
Stride	[4, 4]
Padding	[3, 3]
Activation Function	relu

Figure 6.20: First Layer Information

memory chart of on heap JVM memory and off-heap memory usage is depicted in Figure 6.22. The proposed system made use of off-heap memory and most of the memory is not allocated on the JVM heap but outside of the JVM. This helps to perform the numerical operations faster as data needs not to be copied to and from the JVM but pointers can be passed around for numerical computations avoiding data copying issue.

6.6 Video Analytics in Cloud

This section explains the results obtained by executing the experiments with the proposed video analytics system. The results explain the scalability and robustness of the whole system by analyzing decoded video streams and transferring the video data from local storage to cloud nodes. It also measures the time required to analyze video data on the cloud nodes and gathering the results after the accomplishment of analysis.

The proposed system is executed on the cloud infrastructure as described in the experimental setup section. The input data is first loaded in the RDDs of spark. Spark launches a number of executors and the RDD objects are accessed by each executor in an iteration. The cache manager is responsible to handle the results of the iterations. It maintains a memory pool and

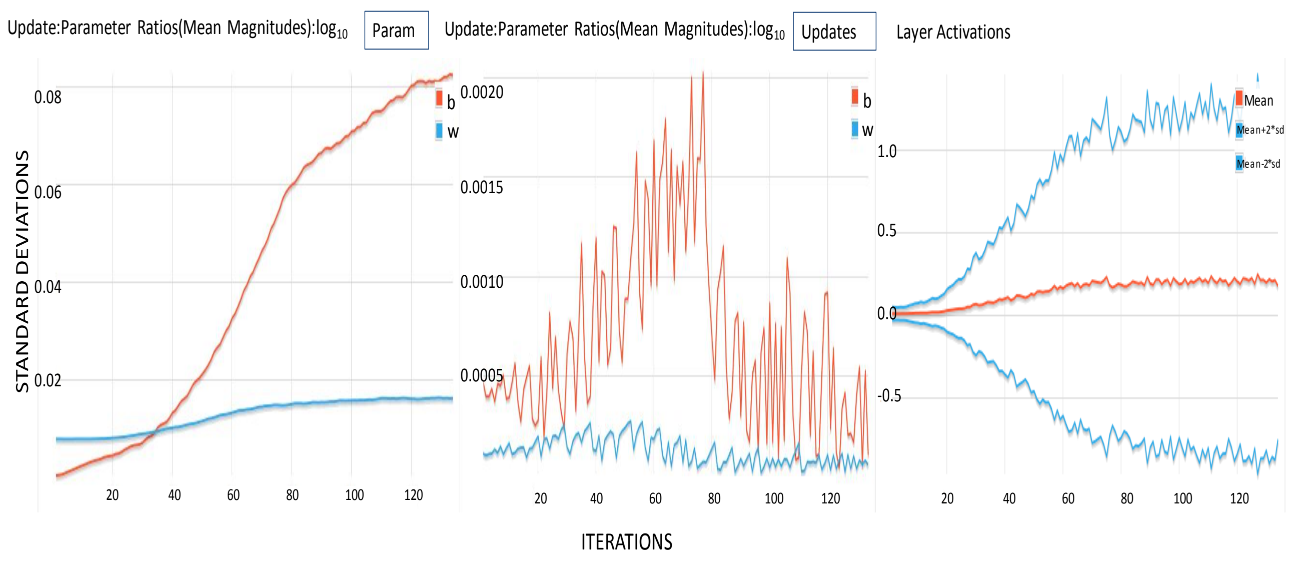


Figure 6.21: Standard Deviations for First Layer

retains the iteration results in it. In case the data is not applicable anymore, it is not needed to be retained in the memory and can be saved on the disk. In this way Spark manages the data and keeps a part of data in memory and the rest of the data is stored on the disk.

We have used the iterative MapReduce framework to perform the analysis of the video streams. Each node in the cloud can execute one or more than one analysis task on the input dataset. In iterative reduce, an analysis task comprises of multiple map and reduce tasks. These multiple map and reduce tasks perform the classification of objects from the input dataset. Spark executes these task in multiple stages and each stage performs further mapping operations. The iterative MapReduce framework is also responsible to schedule the map and reduce tasks and also rescheduling of tasks in case of task failure.

Data Bundle: The video streams are first decoded to extract individual video frames from the input video. Each video stream is recorded at 25 frames per second. The number of decoded video frames is dependent upon the length of video stream being analyzed. The total size of decoded video frames used in the experiments varied from 5GB to 100GB. These large set of individual frames data is not suitable to be directly fed into the spark cluster with iterative reduce framework. The individual video frames are small in size and iterative reduce is designed to work on large data files. Processing of smaller files with iterative reduce only results in the

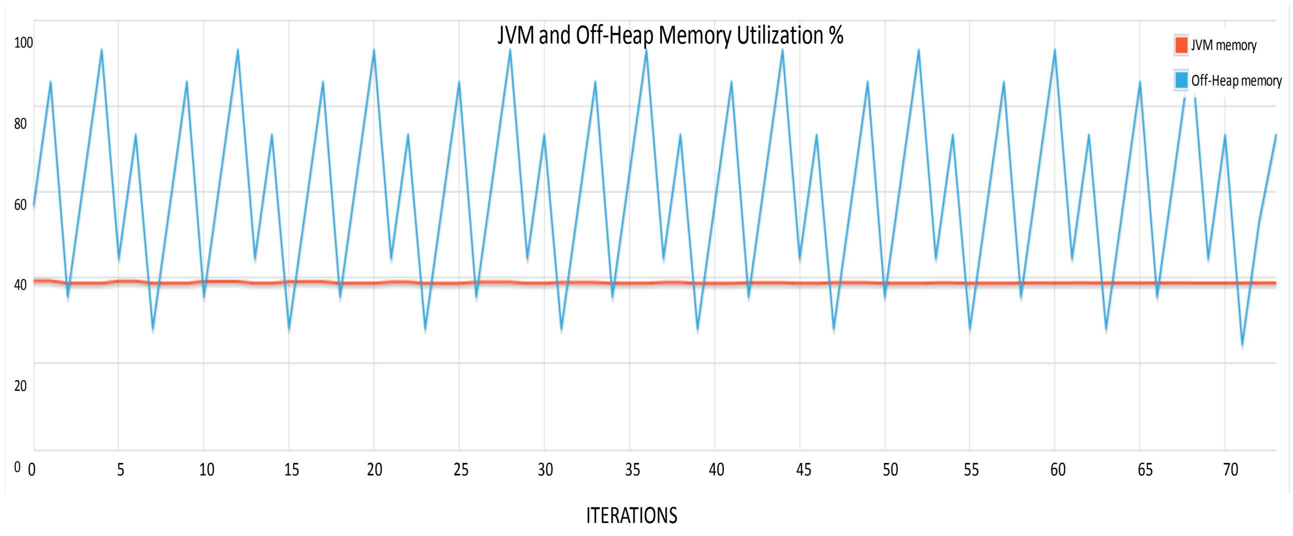


Figure 6.22: Usage of Java Heap Memory

loss of overall performance of the system. These small files are bundled into a large file and then transferred to the cloud nodes for processing.

Data Bundling Time: We have bundled the individual frames by using a batch process and then transferred it to compute cloud for processing. The time required to bundle the data varies with the amount of video frames being considered. This time is directly dependent to the size of the dataset. For a dataset of 10GB to 100GB, the time of batch process varied from 0.25 hours to 3.8 hours as shown in Figure 6.23. Addition of more video frames in the dataset increases the time as well. However, this process needs to be executed only once and the resultant data can be retained in the cloud storage for future analysis.

Data Transfer Time: The data is needed to be transferred to cloud data storage to perform analysis tasks on it. The transfer time to cloud data storage depends on a number of factors such as; network bandwidth and cloud data storage block size. This time is also dependent on the size of the data being transferred. To have an estimate of the transfer time, we measured the transfer time for various sizes of data and plotted in Figure 6.24. It can be observed from the figure that the transfer time varied from 0.36 to 2.18 for a dataset size of 20GB to 100GB. We have also measured this time by changing the cloud storage block size from 128 MB which is the default size to 256MB. However, very little improvement has been recorded in the transfer

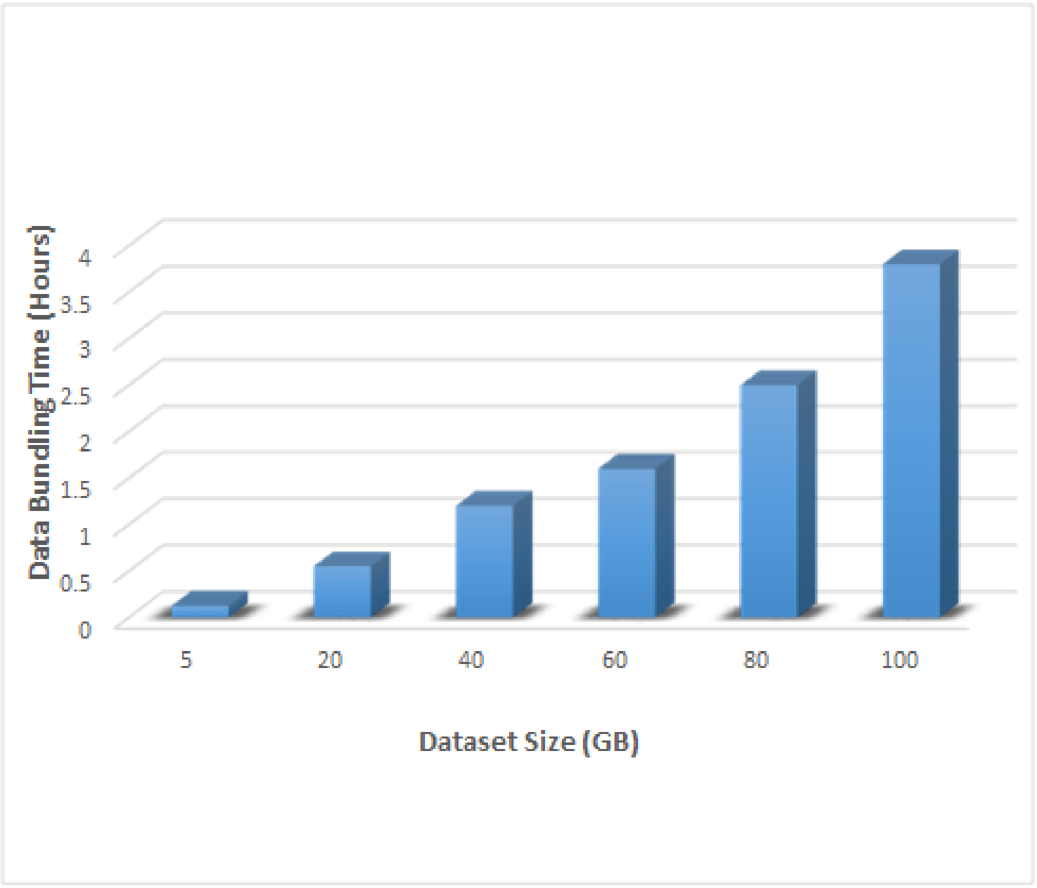


Figure 6.23: Data Bundling Time

time by varying the block size.

Training Time on Cloud Nodes: We have performed the training on multiple nodes of cloud and measured the scalability and robustness of the proposed system. To have a good estimate of the training time we have executed multiple tests on multiple sizes of datasets and plotted their average execution time in Figure 6.25. The average execution time gives a measure of how much training time on average is required to train the proposed system on a specific size of dataset. The dataset sizes have been varied from 20GB to 100GB to measure the time on various cloud nodes. It has been observed that the execution time increases by increasing the size of dataset.

The same set of experiments is then repeated by changing the block size. The change in the block size causes a change in the number of partitions of the dataset. So the experiments were repeated with different block size to see if there are any improvements in the execution time of

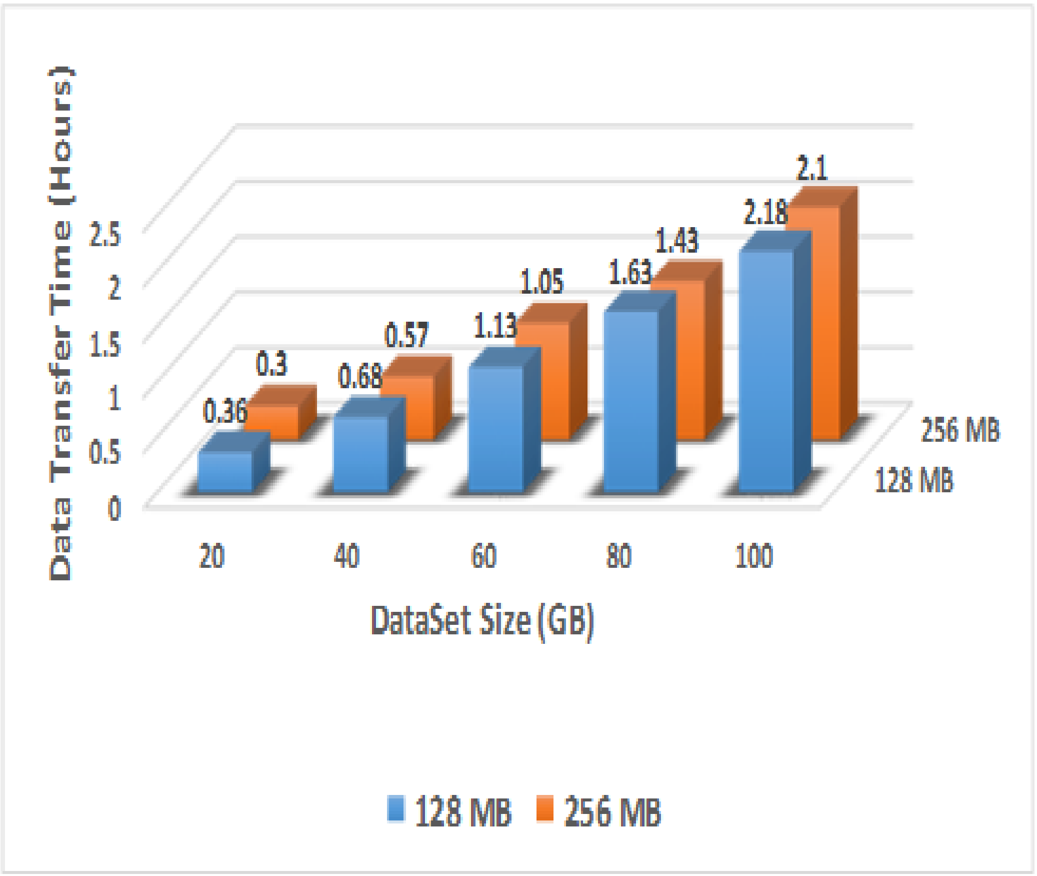


Figure 6.24: Data Transfer Time

the system. It can be seen from the figure that the execution time varied from 1.45 hours to 7.29 hours for a block size of 128MB. For the block size of 256MB, the execution time showed a very little improvement from 1.43 hours to 6.8 hours for the same size of dataset. So the variation in block sizes has a minor impact on the execution time of the system.

Robustness with changing cluster size:

In order to test the scalability of the system and to have an estimate of the execution time taken by the system on average on cloud infrastructure, we have executed the system and repeated the experiments by varying the number of cloud nodes. With each experiment we have increased the number of nodes in the cluster and observed the effects on the overall execution time of the system. The overall execution time of the system with an increase in the number of nodes helped to determine the amount of time required to process the total amount of data. It also gives an estimate that how much each node is contributing in the processing the total amount

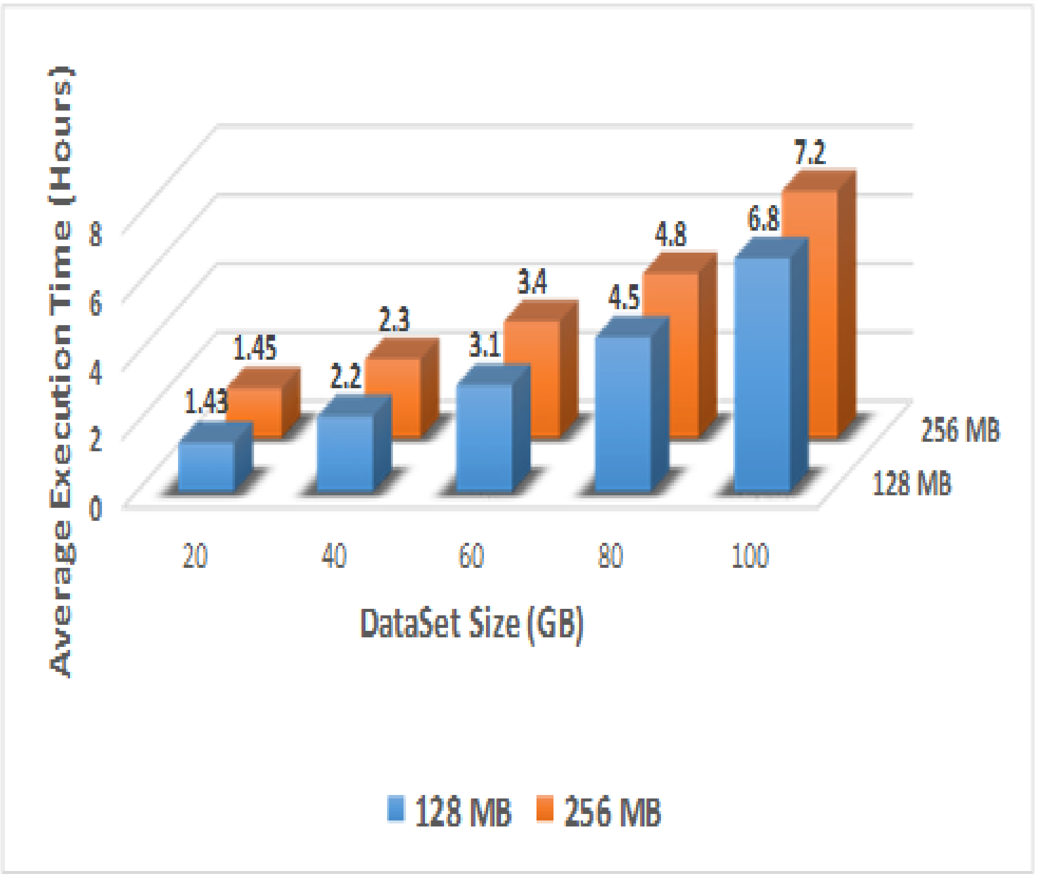


Figure 6.25: Average Time

of data. This provides an estimate on how many nodes should be required to process a specific amount of data in a given amount of time.

Figure 6.26 shows the amount of time required by multiple nodes to process the data. It can be seen from the figure that the total amount of time required to process the data decreases with an addition in the number of nodes of the system. Addition of each node in the cloud decreases the overall execution time. However, this also increases the network communication between nodes. More nodes can be added or removed from the system in order to further increase or decrease the overall processing time. However, it should be noted that having few nodes will increase the processing cost on each node and having too many nodes will increase the communication cost between them.

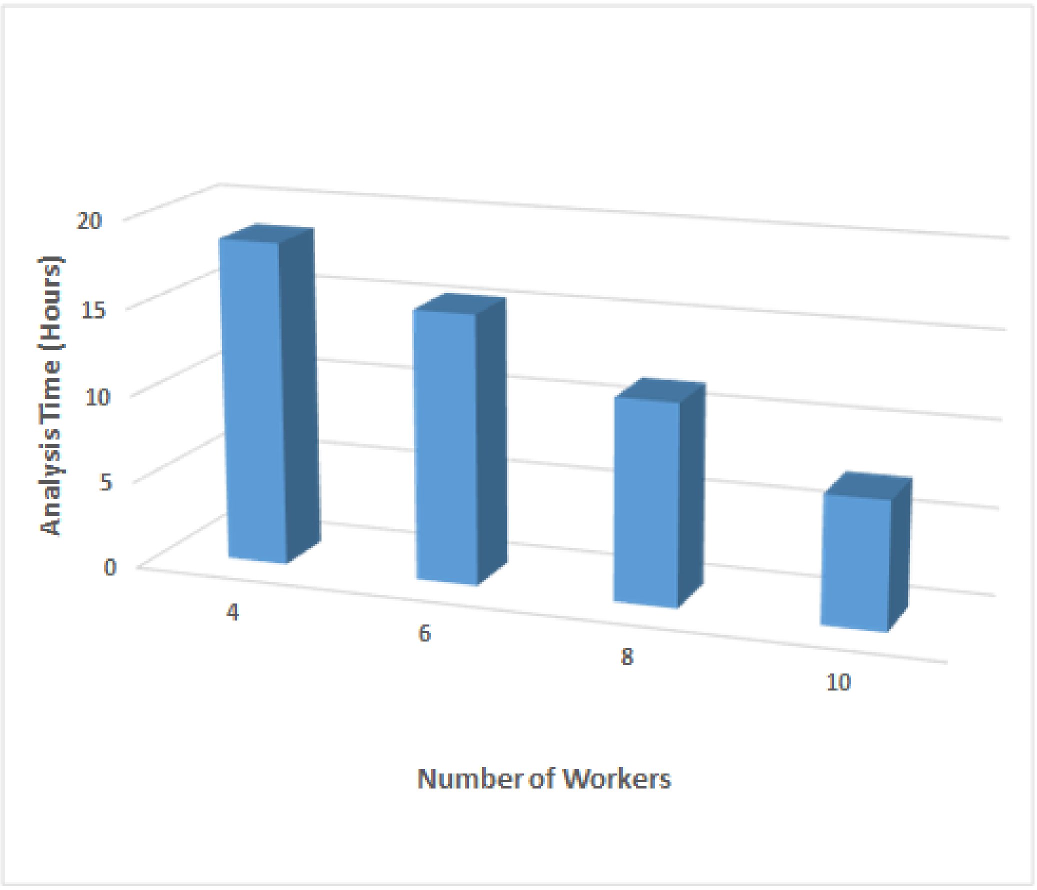


Figure 6.26: Analysis Time in the Cloud

6.7 Conclusion

In this Chapter, the results of the proposed system and its underlying algorithms are presented. It was demonstrated that the accuracy of the proposed deep learning based object detection and classification system can be improved by shifting it from spatial domain to spatial frequency domain. The challenges of illumination, expression and blur can be tackled by using a feature fusion strategy based on the orientation property of the intrinsic mode functions of each object in the dataset. The intrinsic mode functions can be generated by leveraging bi-dimensional empirical mode decomposition. The orientation fusion approach can significantly improve the performance of the proposed system when compared with state-of-the-art.

It is also demonstrated that the performance of the proposed basic video object detection and classification system can be improved by employing deep learning based Convolutional Neural Networks. We demonstrate the improvements in the results by discussing each component of the deep learning pipeline. It is shown that a highly accurate video analytics system can be obtained by an optimal selection of learning rate, regularization, normalization scheme and the design of the number of layers and their parameters.

We also showed that a distributed cloud infrastructure is an efficient way to cope with the challenges of increasing volume of data. The processing time of the video streams can be reduced by increasing the number of nodes in the cloud. The processing time can further be improved by optimizing the resource utilization and employing the efficient and optimal data transfer techniques. However, the processing time is also dependent on the amount of data being analysed. The increase in the amount of data also increases the processing time of the video streams.

Chapter 7

Conclusions and Future Directions

7.1 Outline

This chapter presents the conclusions and summarizes the thesis with a focus on the major contributions of this work. The main conclusions drawn from this research work are presented in Section 7.2. The advancements made by this thesis in the state-of-the-art are presented in Section 7.3. Section 7.4 presents the future directions through which the proposed system could be further extended. We also highlight some potential limitations of the proposed work in this section. Finally, a closing statement is provided in Section 7.5 which concludes the thesis.

7.2 Conclusions

This thesis presents a video analytics system which performs object detection and classification to classify objects from images and video data. The system addresses the problem of processing large-scale video data, that is being generated due to the increasing availability and deployment of video cameras.

The hypothesis of this research work asserts that:

"An optimally tuned spatial-frequency based video analytics system can be developed to automatically identify objects from large number of video streams in a cloud data centre."

In order to support the research hypothesis, a video analytics workflow for automated object detection and classification has been proposed. The orientation, phase and amplitude properties derived from IMFs through BEMD has been studied using Reisz transform. A feature fusion strategy based on the orientation property has been adopted. The proposed system has been optimally tuned to achieve maximum accuracy and performance. A number of experiments have been conducted and their results were discussed in detail. This research work helped to make a good progress in the fulfilment of the research ambition related to research hypothesis.

The main conclusions drawn from this research are the following:

1. It has been demonstrated that machine learning based object detection and classification can be orchestrated in a workflow to minimize the human intervention during classification process. It was also demonstrated that adaptive frame sampling (in which the frames which do not contain any object in them are discarded) can reduce the amount of video frames to be processed and improves the performance and execution time. *This addresses objective 1 of this thesis, as outlined in Chapter 1 which aims to propose a video analytics workflow for the automated detection and classification of objects from large number of video streams.*

2. It was also concluded that a feature fusion strategy based on the orientation components of the video frames can be used to achieve high accuracy, recall and precision for blur and illumination invariant object classification. The orientation, phase and amplitude properties derived from each intrinsic mode function were studied and their performance was shown in terms of accuracy. It has been concluded that the orientation component of the object leads to a higher accuracy rate. It was shown that feature fusion strategy based on the orientation components can significantly improve the accuracy of the video analytics system under uncontrolled conditions. *This fulfils the objective 2 of this thesis which targets to improve the accuracy of the video analytics system using feature fusion strategy in spatial-frequency domain under uncontrolled conditions.*

3. It was concluded that the tuning of the hyper-parameters associated with the deep learning algorithm can be modelled mathematically and the effects of hyper-parameter tuning on the system's performance can be observed. It was concluded that the hyper-parameters including learning rate, momentum, number of epochs and mini-batches have a major contribution in improving the training performance of the system. *This validates the objective 3 of this thesis which asserts that a mathematical model can be designed for the hyper-parameter tuning of the deep learning pipeline to improve the performance of training and inference for video analytics.*

4. It was concluded that the Cloud-based parallel distributed training and inference is an efficient way to speed up the training process. It was demonstrated that the training of the deep learning pipeline can be parallelized by dividing the dataset into small subsets and then passing over these subsets of data to separate Neural Network models for training. It was shown that the models can be trained in parallel and the resultant parameters for each model can be iteratively averaged to form a single trained model. *This addresses objective 4 of this thesis which states that the video analytics system can be scaled using Cloud infrastructure based on the number and size of the video streams.*

5. It was shown that a video object classification case-study can be used to validate the proposed video analytics system with the data that has been collected from real life scenarios. It was also shown that the proposed video analytics system can perform object classification with an accuracy and precision of 97 and 98 percent respectively. *This addresses objective 5 of this thesis which states that the video analytics system can be validated using a video object classification case-study on the data that has been collected from real life scenarios.*

These conclusions endorse the research aim and objectives of this research work as described in Chapter 1. The video analytics system proposed in this work tends to resolve a real life problem of classifying objects from large number of video streams. An operator using this system can specify an object of interest either for surveillance or monitoring. The operator can then identify or track the object to perform analytics automatically using the proposed system.

7.3 Advancements in State-of-the-Art

We advance the state-of-the-art by introducing automation, reducing human intervention, increasing accuracy and by improving training time and performance of large scale video analytics systems. The use of spatial frequency domain features and their fusion in existing literature is scarce. They have not been fully exploited for the blur and illumination invariant object classification which are the most common challenges for object classification. We improved the accuracy of blur and illumination invariant object classification using orientation based features fusion.

The deep learning based video analytics systems are difficult to tune as they involve a number of hyper-parameters. The main parameters include learning rate, momentum, activation function, optimization algorithm and weight parameter initialization. The tuning of these hyper-parameters is still a very challenging task in existing literature. We have advanced the deep learning hyper-parameter research by highlighting the parameters that have a major contribution in improving the performance of the system and formulating a mathematical model to observe the effects of hyper-parameter tuning on the system's performance.

The parallelization of the training of deep learning network plays an important role in reducing the training time of the system. Deep Networks are compute intensive in nature and perform slowly and inefficiently on a single machine, especially, if they are operating on large datasets. These systems are scaled and configured on a Cloud-based distributed infrastructure in this research work for rapid processing of video streams at a reasonable computational cost.

This research work encompasses three major areas of Computer Science. First and foremost Artificial Intelligence and Machine Learning as we have improved the state-of-the-art algorithms with the aim of video stream analytics. To the best of our knowledge, this was the first effort to quantify and demonstrate the shallow and deep learning algorithms for Cloud-based video analytics. We have proposed improvements in both Machine Learning based shallow networks and deep networks for efficient classification of objects from video streams. The algorithms have been applied in a novel setting based on the most optimal parameters for efficient performance and high accuracy.

The second area of Computer Science to benefit from this research is the image and signal processing analysis. We have proposed to improve the accuracy and precision of the proposed system using image and signal analysis. We have proposed the feature selection mechanism in the signal processing domain which is scarce in the current literature.

To improve the performance of the proposed system in terms of training and inference time, we have proposed improvements in the Distributed Computing which is the third beneficiary from this research work. We have advanced the state-of-the-art in improving the execution time of Machine Learning algorithms using a novel and optimized allocation of hyper-parameters for distributed training and inference.

7.4 Future Directions and Potential Limitations

The proposed video analytics system could potentially be improved in a number of ways. There could be many aspects in the proposed system in which potential improvements can be made to enhance the capabilities and performance of the system. Mainly two aspects can be the focus of improvements: (i) Algorithms (ii) Systems

Algorithms: In the proposed object detection and recognition system, the major focus has been made in recognising the humans by considering them as a general object. Different subjects are referred to as potential objects for classification. The capabilities of the proposed system could be further extended by making the system more generic by detecting and classifying objects from different object classes such as cars, bikes and pedestrians.

The integration of the proposed system with other deep learning based approaches can also increase the performance and capabilities of the system. Currently, it is only able to perform analytics on the basis of Convolutional Neural Networks due to the limited availability of labelled data. The system can also be able to leverage the benefits of Reinforcement Learning based models to further improve the performance of the system. This will also help to extend the functionality of the system by classifying other objects such as vehicles or pedestrians without necessitating any metric learning stage.

In order to tackle more realistic surveillance scenarios, such as beyond head-and-shoulder datasets, the work proposed in this thesis can be extended to incorporate more features such as Fourier features or Wavelet features before performing the classification process. The Fourier and Wavelet features can be fused with the EMD features which were proposed in this thesis to further increase the accuracy rates. In order to tackle the challenges on occlusion, more data containing the samples of occlusion can be added into the training dataset. The inclusion of more training data into the dataset will expose the classifier to more information and will help to capture more variations present in the dataset.

To further improve the performance of the proposed system, mechanisms from the transfer learning domain can be adapted into the classifier. As mentioned earlier, that only Convolutional Neural Network has been used in this work due to the limited availability of training data. The problem of limited availability of training data can be resolved by using a pre-trained classifier which was trained on a different but related task. This pre-trained classifier can act as a starting point and can improve the generalization capability to the object classification task.

Another improvement which can be made in the proposed system is the development of an automated mechanism for hyper-parameter optimization of deep learning models. The development of a tool-kit which could automate the process of hyper-parameter optimization can be included as part of the improvements in the system. One way to automate the hyper-parameter optimization is to use the graph learning-based methods. The nodes and edges of a graph-based model can be used to represent the hyper-parameters and their associated weights. Graph pruning can then be performed to select the most optimal values.

A meta-learning model can be developed for Cloud based video analytics which may be able to improve the hyper-parameter tuning on the basis of input datasets and its characteristics. The meta-learning model can also take into account the configurations of underlying in-memory compute cluster and can suggest appropriate tuning parameters for both deep learning model and in-memory cluster. The meta-learning model having the ability to acquire new knowledge versatility can help the systen to focus on the right dataset characteristics for hyper-parameter tuning.

The capabilities of the proposed video analytics system can be further enhanced by incorporating other useful features from the frequency domain so that it can cope with other challenges including rotation and translation variance. It can also help to cope with occlusion and illumination challenges.

Systems: The execution of the proposed deep learning based video analytics system can also be performed on multiple nodes of a GPU-based Cloud infrastructure. The GPU-based cloud infrastructure will help to increase the complexity of the model and will facilitate to experiment on a much bigger dataset. It can also help to use the proposed system for live streaming analytics. Mechanisms need to be developed where GPUs can be utilized on the in-memory processing cluster.

With the utilization of an in-memory cluster coupled with the computation power of GPUs, we can anticipate a higher performance of the video processing platform for live video streams analysis. This can also help to overcome the delays which occur due to various I/O operations and will improve the performance and training time. More innovation can be added on the infrastructure side by incorporating memory models to enhance the scalability and throughput of the proposed system.

The deployment of the proposed system on an Edge enhanced Cloud infrastructure can also play a major role in increasing the performance of the system. The detection and extraction of the objects can be performed on the Edge devices. This will help to filter out the redundant and unwanted frames and will reduce the processing cost.

The use of Field-programmable Gate Arrays (FPGAs) [166] at the edge devices can also help to increase the performance of the system. The FPGAs can be designed to be configured for the proposed system for object detection and extraction. A Software Defined Network (SDN) can then be used to route the extracted data efficiently to the Cloud. This can help the system to perform real-time video stream analytics.

The proposed system can also be enhanced to be used in many other useful applications related to different domains such as Smart Cities and Big Data. It can be enhanced to be used in medical imaging systems, traffic reinforcement systems, satellite imagery systems and many other application domains. The system can be used to identify unknown patterns or to make detection and classification from large amount of data.

7.5 Closing Statement

Due to the recent advances in cameras, cell phones, laptops and other digital devices, particularly the resolution at which they can record an image/video, large amounts of data is being generated daily. This video data requires automated and intelligent analysis to extract useful insights and metadata. Existing video analysis systems are time consuming, lack automation and are unable to harness the power of distributed processing for training and inference.

In this thesis, we presented a Cloud-based, optimally tuned video analytics system to process large numbers of video streams, where the underlying infrastructure was able to scale based on the number and size of the stream(s) being considered. The system automated the video analysis process and reduced manual intervention.

An operator using this system only specifies which object of interest is to be located from the video streams. The object classification is then performed by comparing the object of interest with the pre-stored trained patterns, generating a set of matching scores. The matching scores greater than an empirically-determined threshold reveal the classification of the object. The proposed system proved to be robust to classification errors and can be used as a general-purpose video analytics system.

Bibliography

[1] S. Samad, "The picture in not clear: How many surveillance cameras are there in the uk?," *British security industry association*, vol. 1, no. 2, p. 51, 2013.

[2] D. Murakami Wood, "A report on the surveillance society. for the information commissioner by the surveillance studies network," p. 98, 01 2006.

[3] N. Ohta, "A statistical approach to background subtraction for surveillance systems," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 481–486, IEEE, 2001.

[4] J. S. Bae and T. L. Song, "Image tracking algorithm using template matching and psnf-m," *International Journal of Control, Automation, and Systems*, vol. 6, no. 3, pp. 413–423, 2008.

[5] S. Mantri and D. Bullock, "Analysis of feedforward-backpropagation neural networks used in vehicle detection," *Transportation Research Part C: Emerging Technologies*, vol. 3, no. 3, pp. 161–174, 1995.

[6] C. C. Lin, S. Pankanti, G. Ashour, D. Porat, and J. R. Smith, "Moving camera analytics: Emerging scenarios, challenges, and applications," *IBM Journal of Research and Development*, vol. 59, pp. 5:1–5:10, March 2015.

[7] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 265–272, Omnipress, 2011.

[8] D. Yu and L. Deng, “Deep learning and its applications to signal and information processing [exploratory dsp],” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[9] J. Liu, J. Luo, and M. Shah, “Recognizing realistic actions from videos in the wild,” in *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pp. 1996–2003, IEEE, 2009.

[10] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” in *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

[11] T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[12] S. Murala and Q. J. Wu, “Local ternary co-occurrence patterns: a new feature descriptor for mri and ct image retrieval,” *Neurocomputing*, vol. 119, pp. 399–412, 2013.

[13] S. U. Hussain, T. Napoléon, and F. Jurie, “Face recognition using local quantized patterns,” in *British machive vision conference*, pp. 11–pages, 2012.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: Analysis and implementation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[15] L. Shen and L. Bai, “A review on gabor wavelets for face recognition,” *Pattern analysis and applications*, vol. 9, no. 2-3, pp. 273–292, 2006.

[16] J. Luo, Y. Ma, E. Takikawa, S. Lao, M. Kawade, and B.-L. Lu, “Person-specific sift features for face recognition,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 2, pp. II–593, IEEE, 2007.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[18] V. Ojansivu and J. Heikkilä, “Blur insensitive texture classification using local phase quantization,” in *International conference on image and signal processing*, pp. 236–243, Springer, 2008.

[19] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila, “Recognition of blurred faces using local phase quantization,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[20] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001.

[21] B. Wu, H. Ai, C. Huang, and S. Lao, “Fast rotation invariant multi-view face detection based on real adaboost,” in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 79–84, IEEE, 2004.

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

[23] C. H. Chan, M. A. Tahir, J. Kittler, and M. Pietikäinen, “Multiscale local phase quantization for robust component-based face recognition using kernel fusion of multiple descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1164–1177, 2013.

[24] C. H. Chan, J. Kittler, N. Poh, T. Ahonen, and M. Pietikäinen, “(multiscale) local phase quantisation histogram discriminant analysis with score normalisation for robust face recognition,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 633–640, IEEE, 2009.

[25] Z. Lei, T. Ahonen, M. Pietikäinen, and S. Z. Li, "Local frequency descriptor for low-resolution face recognition," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 161–166, IEEE, 2011.

[26] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[27] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[28] A. J. Izenman, "Linear discriminant analysis," in *Modern multivariate statistical techniques*, pp. 237–280, Springer, 2013.

[29] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski, "Face recognition by independent component analysis," *IEEE Transactions on neural networks*, vol. 13, no. 6, pp. 1450–1464, 2002.

[30] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media, 2005.

[31] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

[32] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using lda-based algorithms," *IEEE Transactions on Neural networks*, vol. 14, no. 1, pp. 195–200, 2003.

[33] S. Jameel, "Face recognition system using pca and dct in hmm," *Int. J. Adv. Res. Comput. Commun. Eng*, vol. 4, pp. 13–18, 2015.

[34] M. Chihaoui, W. Bellil, A. Elkefi, and C. B. Amar, "Face recognition using hmm-lbp," in *International Conference on Hybrid Intelligent Systems*, pp. 249–258, Springer, 2016.

[35] H. Schmid, "Probabilistic part-ofispeech tagging using decision trees," in *New methods in language processing*, p. 154, 2013.

[36] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using rao-blackwellized particle filters.,” in *Robotics: Science and Systems*, vol. 2, pp. 65–72, 2005.

[37] W. Du and Z. Zhan, “Building decision tree classifier on private data,” in *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*, pp. 1–8, Australian Computer Society, Inc., 2002.

[38] M. M. Adankon and M. Cheriet, “Support vector machine,” in *Encyclopedia of biometrics*, pp. 1303–1308, Springer, 2009.

[39] T. Joachims, “Transductive support vector machines,” *Chapelle et al.(2006)*, pp. 105–118, 2006.

[40] N. Vasconcelos and M. J. Saberian, “Boosting classifier cascades,” in *Advances in Neural Information Processing Systems*, pp. 2047–2055, 2010.

[41] P. Cunningham and S. J. Delany, “k-nearest neighbour classifiers,” *Multiple Classifier Systems*, vol. 34, pp. 1–17, 2007.

[42] J. Chen, H. Huang, S. Tian, and Y. Qu, “Feature selection for text classification with naïve bayes,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.

[43] P. Latha, L. Ganesan, and S. Annadurai, “Face recognition using neural networks,” *Signal Processing: An International Journal (SPIJ)*, vol. 3, no. 5, pp. 153–160, 2009.

[44] M. J. Er, S. Wu, J. Lu, and H. L. Toh, “Face recognition with radial basis function (rbf) neural networks,” *IEEE transactions on neural networks*, vol. 13, no. 3, pp. 697–710, 2002.

[45] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, “Going deeper with convolutions,” Cvpr, 2015.

[47] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 437–446, 2015.

[48] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

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

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[51] H. Wang, X. Shi, and D.-Y. Yeung, "Relational stacked denoising autoencoder for tag recommendation.," in *AAAI*, pp. 3052–3058, 2015.

[52] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 643–650, ACM, 2015.

[53] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 4694–4702, IEEE, 2015.

[54] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, pp. 568–576, 2014.

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

[56] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[57] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 34–42, 2015.

[58] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*, pp. 3642–3649, IEEE, 2012.

[59] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[60] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[61] F. J. Huang and Y. LeCun, "Large-scale learning with svm and convolutional for generic object categorization," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 284–291, IEEE, 2006.

[62] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

[63] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[64] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, 2015.

[65] G. H. Nguyen, S. L. Phung, and A. Bouzerdoum, “Reduced training of convolutional neural networks for pedestrian detection,” in *International Conference on Information Technology and Applications*, 2009.

[66] K. Kang, W. Ouyang, H. Li, and X. Wang, “Object detection from video tubelets with convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 817–825, 2016.

[67] Z. Xu, Y. Yang, and A. G. Hauptmann, “A discriminative cnn video representation for event detection,” in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 1798–1807, IEEE, 2015.

[68] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[69] J. Y.-H. Ng, F. Yang, and L. S. Davis, “Exploiting local features from deep networks for image retrieval,” *arXiv preprint arXiv:1504.05133*, 2015.

[70] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, “Exploiting image-trained cnn architectures for unconstrained video classification,” *arXiv preprint arXiv:1503.04144*, 2015.

[71] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[72] H. Liu, Y. Wu, and F. Sun, “Extreme trust region policy optimization for active object recognition,” *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2253–2258, 2018.

[73] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable object detection using deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154, 2014.

[74] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, “Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1728–1740, 2008.

[75] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas, “Joint embeddings of shapes and images via cnn image purification.,” *ACM Trans. Graph.*, vol. 34, no. 6, pp. 234–1, 2015.

[76] W. Li, R. Zhao, T. Xiao, and X. Wang, “Deepreid: Deep filter pairing neural network for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 152–159, 2014.

[77] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Deep metric learning for person re-identification,” in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 34–39, IEEE, 2014.

[78] E. Ahmed, M. Jones, and T. K. Marks, “An improved deep learning architecture for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3908–3916, 2015.

[79] Z. Waheed, M. U. Akram, A. Waheed, M. A. Khan, A. Shaukat, and M. Ishaq, “Person identification using vascular and non-vascular retinal features,” *Computers & Electrical Engineering*, vol. 53, pp. 359–371, 2016.

[80] C. Wang, M. Pelillo, and K. Siddiqi, “Dominant set clustering and pooling for multi-view 3d object recognition,” in *Proceedings of British Machine Vision Conference (BMVC)*, vol. 12, 2017.

[81] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection.,” in *CVPR*, vol. 1, p. 4, 2017.

[82] J. T. Geiger, M. Kneißl, B. W. Schuller, and G. Rigoll, “Acoustic gait-based person identification using hidden markov models,” in *Proceedings of the 2014 Workshop on Mapping Personality Traits Challenge and Workshop*, pp. 25–30, ACM, 2014.

[83] D. Reynolds, “Gaussian mixture models,” *Encyclopedia of biometrics*, pp. 827–832, 2015.

[84] B. Egger, D. Kaufmann, S. Schönborn, V. Roth, and T. Vetter, “Copula eigenfaces,” in *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: Volume 1: GRAPP*, pp. 50–58, SCITEPRESS-Science and Technology Publications, Lda, 2016.

[85] J. Wang and L. Jia, “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,” 2016.

[86] M. Carrasco, L. Pizarro, and D. Mery, “Bimodal biometric person identification system under perturbations,” in *Advances in Image and Video Technology* (D. Mery and L. Rueda, eds.), (Berlin, Heidelberg), pp. 114–127, Springer Berlin Heidelberg, 2007.

[87] N. Fox, R. Gross, P. de Chazal, J. Cohn, and R. Reilly, “Person identification using multimodal features: Speech, lip, and face,” in *ACM Multimedia Workshop in Biometrics Methods and Applications (WBMA 2003)*, pp. 25 – 32, January 2003.

[88] M. Bassiouni, W. Khaleefa, E. El-Dahshan, and A.-B. M. Salem, “A machine learning technique for person identification using ecg signals,” *Int. J. Appl. Phys*, vol. 1, pp. 37–41, 2016.

[89] N. Radha, A. Shahina, and A. N. Khan, “A person identification system combining recognition of face and lip-read passwords,” in *2015 International Conference on Computing and Network Communications (CoCoNet)*, pp. 882–885, Dec 2015.

[90] A. A. M. Abushariah, T. S. Gunawan, J. Chebil, and M. A. M. Abushariah, “Voice based automatic person identification system using vector quantization,” in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, pp. 549–554, July 2012.

[91] Z. Lai, Y. Xu, Q. Chen, J. Yang, and D. Zhang, “Multilinear sparse principal component analysis,” *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1942–1950, 2014.

[92] C. S. F. Smowton, R. Chaiken, W. Cui, O. H. Foehr, J. R. Lorch, D. Molnar, B. J. Parno, S. Saroiu, and A. Wolman, "Personal identification combining proximity sensing with biometrics," Jan. 28 2016. US Patent App. 14/846,172.

[93] L. Nanni, M. Munaro, S. Ghidoni, E. Menegatti, and S. Brahnam, "Ensemble of different approaches for a reliable person re-identification system," *Applied Computing and Informatics*, vol. 12, no. 2, pp. 142 – 153, 2016.

[94] S.-U. Lee, Y.-S. Cho, S.-C. Kee, and S. R. Kim, "Real-time facial feature detection for person identification system.," in *MVA*, pp. 148–151, 2000.

[95] A. Dhanalakshmi and B. Srinivasan, "Improved person identification system using face biometric detection," *International Journal of Advanced Networking and Applications*, vol. 4, no. 5, p. 1731, 2013.

[96] Y. Lu, A. Fleury, J. Boonaert, S. Lecoeuche, and S. Ambellouis, "Online person identification and new person discovery using appearance features," in *2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pp. 1–8, Dec 2015.

[97] G. Sasidharan, "Retina based personal identification system using skeletonization and similarity transformation," *Int. J. Comput. Trends Technol*, vol. 17, no. 3, pp. 144–147, 2014.

[98] M. Munaro, S. Ghidoni, D. T. Dizmen, and E. Menegatti, "A feature-based approach to people re-identification using skeleton keypoints," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5644–5651, IEEE, 2014.

[99] M. Yang, L. Zhang, L. Zhang, and D. Zhang, "Monogenic binary pattern (mbp): A novel feature extraction and representation model for face recognition," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2680–2683, IEEE, 2010.

[100] M. Yang, L. Zhang, S. C.-K. Shiu, and D. Zhang, "Monogenic binary coding: An efficient local feature extraction approach to face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1738–1751, 2012.

[101] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, “Local gabor binary pattern histogram sequence (lgbphs): a novel non-statistical model for face representation and recognition,” in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 786–791, IEEE, 2005.

[102] B. Zhang, S. Shan, X. Chen, and W. Gao, “Histogram of gabor phase patterns (hgpp): A novel object representation approach for face recognition,” *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 57–68, 2007.

[103] S. Shan, W. Zhang, Y. Su, X. Chen, and W. Gao, “Ensemble of piecewise fda based on spatial histograms of local (gabor) binary patterns for face recognition,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, IEEE, 2006.

[104] X. Tan and B. Triggs, “Fusing gabor and lbp feature sets for kernel-based face recognition,” in *International Workshop on Analysis and Modeling of Faces and Gestures*, pp. 235–249, Springer, 2007.

[105] J. Van De Weijer and C. Schmid, “Coloring local feature extraction,” in *European conference on computer vision*, pp. 334–348, Springer, 2006.

[106] M. J. Swain and D. H. Ballard, “Indexing via color histograms,” in *Active Perception and Robot Vision*, pp. 261–273, Springer, 1992.

[107] B. V. Funt and G. D. Finlayson, “Color constant color indexing,” *IEEE transactions on Pattern analysis and Machine Intelligence*, vol. 17, no. 5, pp. 522–529, 1995.

[108] J. H. Lai, P. C. Yuen, and G. C. Feng, “Face recognition using holistic fourier invariant features,” *Pattern Recognition*, vol. 34, no. 1, pp. 95 – 109, 2001.

[109] K. Yan, Y. Chen, and D. Zhang, “Gabor surface feature for face recognition,” in *Pattern Recognition (ACPR), 2011 First Asian Conference on*, pp. 288–292, IEEE, 2011.

[110] J. Flusser and T. Suk, “Rotation moment invariants for recognition of symmetric objects,” *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3784–3790, 2006.

[111] S. Pereira and T. Pun, “Robust template matching for affine resistant image watermarks,” *IEEE transactions on image Processing*, vol. 9, no. 6, pp. 1123–1129, 2000.

[112] J. Cogdell and P. Michel, “On the complex moments of symmetric power l-functions at s= 1,” *International Mathematics Research Notices*, vol. 2004, no. 31, pp. 1561–1617, 2004.

[113] V. Ojansivu, E. Rahtu, and J. Heikkila, “Rotation invariant local phase quantization for blur insensitive texture analysis,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[114] Z. Wu and N. E. Huang, “Ensemble empirical mode decomposition: a noise-assisted data analysis method,” *Advances in adaptive data analysis*, vol. 1, no. 01, pp. 1–41, 2009.

[115] S. Ehsan, S. M. U. Abdullah, M. J. Akhtar, D. P. Mandic, K. D. McDonald-Maier, *et al.*, “Multi-scale pixel-based image fusion using multivariate empirical mode decomposition,” *Sensors*, vol. 15, no. 5, pp. 10923–10947, 2015.

[116] A. Linderhed, “Image empirical mode decomposition: A new tool for image processing,” *Advances in Adaptive Data Analysis*, vol. 1, no. 02, pp. 265–294, 2009.

[117] Z. Liu and S. Peng, “Directional emd and its application to texture segmentation,” *Science in China Series F: Information Sciences*, vol. 48, no. 3, p. 354, 2005.

[118] M. U. Yaseen, A. Anjum, and N. Antonopoulos, “Spatial frequency based video stream analysis for object classification and recognition in clouds,” in *2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pp. 18–26, Dec 2016.

[119] A. Sharma, “Apache kafka: Next generation distributed messaging system,” 2015.

[120] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache flink: Stream and batch processing in a single engine,” *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[121] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[122] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[123] J. Halgaš and A. Janota, “Technical devices cooperation to obtain data for 3d environment modelling,” in *International Conference on Transport Systems Telematics*, pp. 330–337, Springer, 2011.

[124] I. Haritaoglu, D. Harwood, and L. S. Davis, “W/sup 4: real-time surveillance of people and their activities,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[125] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[126] Y. Matsuda, H. Hoashi, and K. Yanai, “Recognition of multiple-food images by detecting candidate regions,” in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pp. 25–30, IEEE, 2012.

[127] L. Jing, M. K. Ng, and J. Z. Huang, “An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data,” *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 8, 2007.

[128] G. B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, “The application of cloud computing to astronomy: A study of cost and performance,” in *e-Science Workshops, 2010 Sixth IEEE International Conference on*, pp. 1–7, IEEE, 2010.

[129] H. Tan and L. Chen, “An approach for fast and parallel video processing on apache hadoop clusters,” in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pp. 1–6, IEEE, 2014.

[130] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, ACM, 2009.

[131] H. Lee, A. Battle, R. Raina, and A. Y. Ng, “Efficient sparse coding algorithms,” in *Advances in neural information processing systems*, pp. 801–808, 2007.

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

[133] O. C. L. Au and M. C. Kung, “Block parallel and fast motion estimation in video coding,” Oct. 29 2009. US Patent App. 12/111,322.

[134] J. Kong and Y. Deng, “Gpu accelerated face detection,” in *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pp. 584–588, IEEE, 2010.

[135] C. H. Messom and A. L. Barczak, “Stream processing for fast and efficient rotated haar-like features using rotated integral images,” *International journal of intelligent systems technologies and applications*, vol. 7, no. 1, pp. 40–57, 2009.

[136] G. Lacey, G. W. Taylor, and S. Areibi, “Deep learning on fpgas: Past, present, and future,” *arXiv preprint arXiv:1602.04283*, 2016.

[137] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, “Dlau: A scalable deep learning accelerator unit on fpga,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 513–517, 2017.

[138] Q. Yu, C. Wang, X. Ma, X. Li, and X. Zhou, “A deep learning prediction process accelerator based fpga,” in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pp. 1159–1162, IEEE, 2015.

[139] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h. 264/avc video coding standard,” *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[140] X. Tang, Z. Ou, T. Su, and P. Zhao, “Cascade adaboost classifiers with stage features optimization for cellular phone embedded face detection system,” in *International Conference on Natural Computation*, pp. 688–697, Springer, 2005.

[141] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, “Surftrac: Efficient tracking and continuous object recognition using local feature descriptors,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2937–2944, IEEE, 2009.

[142] T. Wu and A. Toet, “Speed-up template matching through integral image based weak classifiers,” *Journal of Pattern Recognition Research*, vol. 1, pp. 1–12, 2014.

[143] A. Suruliandi, K. Meena, and R. R. Rose, “Local binary pattern and its derivatives for face recognition,” *IET Computer Vision*, vol. 6, no. 5, pp. 480–488, 2012.

[144] W.-H. Liao, “Region description using extended local ternary patterns,” in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 1003–1006, IEEE, 2010.

[145] J. Henseler, C. M. Ringle, and R. R. Sinkovics, “The use of partial least squares path modeling in international marketing,” in *New challenges to international marketing*, pp. 277–319, Emerald Group Publishing Limited, 2009.

[146] S. Huh and S. E. Fienberg, “Discriminative topic modeling based on manifold learning,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 4, p. 20, 2012.

[147] M. Y. Park and T. Hastie, “L1-regularization path algorithm for generalized linear models,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 4, pp. 659–677, 2007.

[148] S. Das, “Filters, wrappers and a boosting-based hybrid for feature selection,” in *Icml*, vol. 1, pp. 74–81, 2001.

[149] Y. Xiao, E. E. Moodie, and M. Abrahamowicz, "Comparison of approaches to weight truncation for marginal structural cox models," *Epidemiologic Methods*, vol. 2, no. 1, pp. 1–20, 2013.

[150] E. Keogh and A. Mueen, "Curse of dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, pp. 314–315, Springer, 2017.

[151] S. Samad, A. Haq, *et al.*, "Orientation invariant object recognitions using geometric moments invariants and color histograms," *International Journal of Computer and Electrical Engineering*, vol. 7, no. 2, p. 101, 2015.

[152] J. Van de Weijer and C. Schmid, "Blur robust and color constant image description," in *Image Processing, 2006 IEEE International Conference on*, pp. 993–996, IEEE, 2006.

[153] J.-W. Wang, N. T. Le, J.-S. Lee, and C.-C. Wang, "Color face image enhancement using adaptive singular value decomposition in fourier domain for face recognition," *Pattern Recognition*, vol. 57, pp. 31–49, 2016.

[154] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, "Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic acids research*, vol. 30, no. 14, pp. 3059–3066, 2002.

[155] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury, "The dual-tree complex wavelet transform," *IEEE signal processing magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[156] N. Baydar and A. Ball, "A comparative study of acoustic and vibration signals in detection of gear failures using wigner–ville distribution," *Mechanical systems and signal processing*, vol. 15, no. 6, pp. 1091–1107, 2001.

[157] Z. Wu, N. E. Huang, and X. Chen, "The multi-dimensional ensemble empirical mode decomposition method," *Advances in Adaptive Data Analysis*, vol. 1, no. 03, pp. 339–372, 2009.

[158] Z. Peng, W. T. Peter, and F. Chu, “A comparison study of improved hilbert–huang transform and wavelet transform: application to fault diagnosis for rolling bearing,” *Mechanical systems and signal processing*, vol. 19, no. 5, pp. 974–988, 2005.

[159] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015.

[160] “Nd4j.” Accessed: 2018-01-19.

[161] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[162] Y. Sun, X. Wang, and X. Tang, “Deep learning face representation from predicting 10,000 classes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1891–1898, 2014.

[163] A. BioID, “Bioid face ddatabase,” 2001.

[164] M.-H. Yang, N. Ahuja, and D. Kriegman, “Face recognition using kernel eigenfaces,” in *Image processing, 2000. proceedings. 2000 international conference on*, vol. 1, pp. 37–40, IEEE, 2000.

[165] Y. LeCun *et al.*, “Lenet-5, convolutional neural networks,” *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[166] D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami, “Real-time face detection and lip feature extraction using field-programmable gate arrays,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 4, pp. 902–912, 2006.