

SCALABLE AND DISTRIBUTED METHODS FOR LARGE-SCALE VISUAL COMPUTING

A THESIS

submitted by

DINESH SINGH

for the award of the degree

of

DOCTOR OF PHILOSOPHY



भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

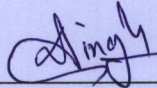
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY HYDERABAD

DECEMBER 2018

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.



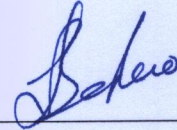
(Dinesh Singh)

CS14RESCH11003

(Roll No.)

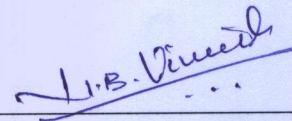
Approval Sheet

This thesis entitled "Scalable and Distributed Methods For Large-Scale Visual Computing" by Mr. Dinesh Singh is approved for the degree of Doctor of Philosophy from IIT Hyderabad.

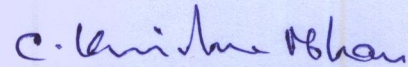


Prof. S. V. Rao
IIT Guwahati
Examiner 1

Prof. K. Sreenivasa Rao
IIT Kharagpur
Examiner 2

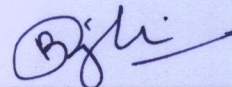


Dr. Vineeth N Balasubramanian
IIT Hyderabad
Internal Examiner



Dr. C. Krishna Mohan
IIT Hyderabad
Adviser/Guide

-Name and affiliation-
Co-Adviser



Dr. Bharat B Panigrahi
IIT Hyderabad
Chairman

Acknowledgements

This dissertation is the result of a long and arduous journey through the last four and a half years. Like all journeys, mine was filled with successes and failures both small and big. Holding my hand through the failures and buoying me on during my successes, my parents have been my source of perseverance and determination. My mother has been especially encouraging in times of crisis with hopes of a better tomorrow while my father has been constant in his unwavering support in my endeavors. I owe it all to my parents who have made me capable of charting my own course and making my own choices.

I consider myself fortunate to have Prof. C Krishna Mohan as my adviser as he has been a constant source of wisdom and encouragement through all these years. He has been a guide in the truest sense of the word and brought out the best in me. Apart from giving me a platform for showcasing my work at different avenues and allowing me to mentor students, he has shown immense trust in my abilities which have given me confidence and the zeal to become better every day. As I leave IIT Hyderabad, I will definitely miss our daily conversations. I hope that one day I will be able to emulate a fraction of his humility and maturity.

Finally, my time at IIT Hyderabad has flown by due to my friends who have made it a memorable journey. I would like to thank Dr. Mettu Srinivas and Dr. Debaditya Roy for all their support during the initial days of my research. A special thanks to my batch-mates with whom I continue to share my thoughts to this day. Also, my colleagues at VIGIL, Nazil, Vishnu, Kunal, Abhijeet, Sumit, and Bedanta, who have contributed to my growth as a researcher. A heartfelt thanks to all my lab-mates for maintaining a vibrant environment in our lab which has encouraged creativity and positive thinking.

Dedication

To my parents

Abstract

The objective of this research work is to develop efficient, scalable, and distributed methods to meet the challenges associated with the processing of immense growth in visual data like images, videos, etc. The motivation stems from the fact that the existing computer vision approaches are computation intensive and cannot scale-up to carry out analysis on the large collection of data as well as to perform the real-time inference on the resource-constrained devices. Some of the issues encountered are: 1) increased computation time for high-level representation from low-level features, 2) increased training time for classification methods, and 3) carry out analysis in real-time on the live video streams in a city-scale surveillance network. The issue of scalability can be addressed by model approximation and distributed implementation of computer vision algorithms. But existing scalable approaches suffer from the high loss in model approximation and communication overhead. In this thesis, our aim is to address some of the issues by proposing efficient methods for reducing the training time over large datasets in a distributed environment, and for real-time inference on resource-constrained devices by scaling-up computation-intensive methods using the model approximation.

A scalable method *Fast-BoW* is presented for reducing the computation time of bag-of-visual-words (BoW) feature generation for both hard and soft vector-quantization with time complexities $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$, respectively, where $|\mathbf{h}|$ is the size of the hash table used in the proposed approach and k is the vocabulary size. We replace the process of finding the closest cluster center with a softmax classifier which improves the cluster boundaries over k -means and can also be used for both hard and soft BoW encoding. To make the model compact and faster, the real weights are quantized into integer weights which can be represented using few bits (2 – 8) only. Also, on the quantized weights, the hashing is applied to reduce the number of multiplications which accelerate the entire process. Further the effectiveness of the video representation is improved by exploiting the structural information among the various entities or same entity over the time which is generally ignored by BoW representation. The interactions of the entities in a video are formulated as a graph of geometric relations among space-time interest points. The activities represented as graphs are recognized using a SVM with low complexity graph kernels, namely, random walk kernel ($O(n^3)$) and Weisfeiler-Lehman kernel ($O(n)$). The use of graph kernel provides robustness to slight topological deformations, which may occur due to the presence of noise and viewpoint variation in data. The further issues such as computation and storage of the large kernel matrix are addressed using the Nystrom method for kernel linearization.

The second major contribution is in reducing the time taken in learning of kernel sup-

port vector machine (SVM) from large datasets using distributed implementation while sustaining classification performance. We propose *Genetic-SVM* which makes use of the distributed genetic algorithm to reduce the time taken in solving the SVM objective function. Further, the data partitioning approaches achieve better speed-up than distributed algorithm approaches but invariably leads to the loss in classification accuracy as global support vectors may not have been chosen as local support vectors in their respective partitions. Hence, we propose *DiP-SVM*, a distribution preserving kernel SVM where the first and second order statistics of the entire dataset are retained in each of the partitions. This helps in obtaining local decision boundaries which are in agreement with the global decision boundary thereby reducing the chance of missing important global support vectors. Further, the task of combining the local SVMs hinder the training speed. To address this issue, we propose *Projection-SVM*, using subspace partitioning where a decision tree is constructed on a projection of data along the direction of maximum variance to obtain smaller partitions of the dataset. On each of these partitions, a kernel SVM is trained independently, thereby reducing the overall training time. Also, it results in reducing the prediction time significantly.

Another issue addressed is the recognition of traffic violations and incidents in real-time in a city-scale surveillance scenario. The major issues are accurate detection and real-time inference. The central computing infrastructures are unable to perform in real-time due to large network delay from video sensor to the central computing server. We propose an efficient framework using edge computing for deploying large-scale visual computing applications which reduces the latency and the communication overhead in a camera network. This framework is implemented for two surveillance applications, namely, motorcyclists without a helmet and accident incident detection. An efficient cascade of convolutional neural networks (CNNs) is proposed for incrementally detecting motorcyclists and their helmets in both sparse and dense traffic. This cascade of CNNs shares common representation in order to avoid extra computation and over-fitting. The accidents of the vehicles are modeled as an unusual incident. The deep representation is extracted using denoising stacked auto-encoders trained from the spatio-temporal video volumes of normal traffic videos. The possibility of an accident is determined based on the reconstruction error and the likelihood of the deep representation. For the likelihood of the deep representation, an unsupervised model is trained using one class SVM. Also, the intersection points of the vehicle's trajectories are used to reduce the false alarm rate and increase the reliability of the overall system. Both the approaches are evaluated on the real traffic videos collected from the video surveillance network of Hyderabad city in India. The experiments on the real traffic videos demonstrate the efficacy of the proposed approaches.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	vi
List of tables	xi
List of figures	xii
1 Introduction to large-scale visual computing	1
1.1 Visual computing	1
1.1.1 Feature representation techniques	2
1.1.2 Modeling Techniques	2
1.1.3 Challenges in large-scale visual computing	3
1.2 Issues addressed in this thesis	4
1.3 Organization of the thesis	5
2 Review of scalable and distributed methods for visual computing	7
2.1 Large-scale feature representation	7
2.2 Large-scale modeling	8
2.2.1 Support vector machines (SVM)	8
2.2.2 K-nearest neighbours (k-NNs)	11
2.2.3 Neural networks (NNs)	11
2.3 System architecture for large-scale surveillance network	12
2.4 Large-scale visual computing applications	13
2.4.1 Abnormal activity recognition	14
2.4.2 Helmetless motorcyclists detection	14
2.4.3 Accident detection	15
2.5 Observations from the review	16

2.6	Summary	17
3	Fast-BoW: Scaling bag-of-visual-words (BoW) generation	19
3.1	Fast cluster prediction using softmax with quantized weights	21
3.1.1	Learning probability distribution of the clusters	21
3.1.2	Weight quantization and hashing	22
3.1.3	Hierarchical tree for hard BoW generation	24
3.2	Experiments and results	24
3.3	Summary	27
4	Graph representation for abnormal activity recognition	29
4.1	Graph formulation of activities in a video	30
4.1.1	Detection of space-time interest points	30
4.1.2	Graph formulation of a video	32
4.1.3	Activity recognition	32
4.2	Experiments and results	35
4.3	Summary	38
5	Distributed QP solver for kernel SVM using genetic algorithm	39
5.1	Operations of Genetic-SVM	41
5.1.1	Random key encoding	42
5.1.2	Initial population generation	43
5.1.3	Fitness evaluation	44
5.1.4	Selection operator	44
5.1.5	Reproduction operator	44
5.1.6	Elitism	45
5.2	Distributed execution of Genetic-SVM	45
5.2.1	Distributed Genetic-SVM	46
5.2.2	Distributed Genetic-SVM for large dataset	47
5.3	Experiments and results	48
5.4	Summary	51
6	DiP-SVM : Distribution preserving distributed kernel SVM	53
6.1	Distribution preserving partitioning	54
6.2	Distributed execution of DiP-SVM	56
6.2.1	Empirical evaluation	57
6.3	Experiments and results	59

6.4	Summary	63
7	Distributed kernel SVM using subspace partitioning	65
7.1	Subspace partitioning using decision tree and dominant eigenvector	66
7.2	Training and prediction in distributed environment	68
7.2.1	Prediction using proposed distributed SVM	68
7.2.2	Time complexity analysis	70
7.3	Experiments and results	72
7.3.1	Sketches of correctness	72
7.3.2	Comparison with state-of-the-art methods	74
7.4	Summary	77
8	Edge computing-based framework for city-scale traffic incident detection	79
8.1	Edge computing framework for traffic monitoring	80
8.2	Real-time detection of motorcyclists without helmet	81
8.2.1	Detection of motorcyclist using CNN based object detector	81
8.2.2	Localization of the rider's head	82
8.2.3	Classification of head and helmet using CNN	82
8.2.4	Temporal consolidation of the alerts:	84
8.2.5	Experiments and results	85
8.3	Deep spatio-temporal representation for detection of road accident	90
8.3.1	Spatio-temporal volume generation	91
8.3.2	Stacked denoising autoencoder (SDAE)	92
8.3.3	Detection of intersection points in trajectories	93
8.3.4	Accident score generation	94
8.3.5	Experiments and results	96
8.4	Summary	101
9	Summary and future work	103
9.1	Contributions of the thesis	104
9.2	Directions for Further Research	104
	References	105

List of Tables

3.1	Details of the datasets used	24
3.2	Comparison of the classification performance (%) of the proposed Fast-BoW approach with the existing Seq-BoW and Tree-BoW approaches . . .	25
3.3	Comparison of time (milliseconds) taken in BoW generation per test video.	26
4.1	Comparison of classification performance (%) of proposed approach with existing bag-of-words (BoW) approaches using STIP, SIFT and dense-trajectories (DT)	37
4.2	Performance comparison (%) of proposed approach with existing methods .	38
5.1	Details of datasets used to evaluate the performance of Genetic-SVM	49
5.2	Performance of classification (%) of the Genetic-SVM and comparison with SMO using LIBSVM	50
5.3	Training time (seconds) of the Genetic-SVM and comparison with sequential SVM	50
6.1	Distortion in the distributions of partitions for random partitioning vs. proposed distribution preserving partitioning on kddcup99 dataset	56
6.2	Details of the datasets used	60
6.3	Comparison of classification performance (%) and average runtime (sec.) .	61
7.1	Performance of classification (%) of proposed distributed SVM and comparison with LIBSVM, DC-SVM, CA-SVM and DT-SVM.	75
7.2	Various evaluation metrics for effectiveness & efficiency of the proposed distributed SVM.	75
8.1	Comparison of classification performance of ‘helmet’ vs. ‘without helmet’ .	89
8.2	Space & time requirements of the proposed models.	89
8.3	Area under curve (AUC) for various modalities and methods	99

List of Figures

2.1	Schematic of cloud SVM architecture [14].	9
2.2	Training flow of cascade SVM [32].	10
2.3	The sample video frames, showing various difficulties in the collected video dataset for accident detection.	16
3.1	Scaling the process of BoW generation.	21
3.2	Effect of the vocabulary size on classification accuracy and BoW generation time.	25
3.3	Comparison of the classification loss in existing tree based approach and the proposed Fast-BoW approach with respect to sequential BoW.	26
3.4	Scaling factor of Fast-BoW with respect to sequential BoW approach and the existing tree based approach.	27
4.1	Block diagram of the proposed framework for abnormal activity recognition in surveillance videos.	31
4.2	Illustration of normal and abnormal sample and corresponding graphs from all datasets.	36
5.1	Genetic-SVM operations. (A) The flow diagram of the steps in genetic algorithm. (B) The process of the solutions representation using random key encoding. (C) The process of crossover operation for reproduction of new candidate solutions.	42
5.2	Proposed architecture of distributed Genetic-SVM	47
5.3	Proposed architecture of distributed Genetic-SVM for large dataset	48
5.4	Performance of classification for Genetic-SVM on the GISETTE dataset after 2400 epoch.	49
5.5	Performance of genetic algorithm based optimization of QP problem for 10 runs using population size 1000 and pool size 2000 at each slave process and using population size 100 and pool size 1000 at master process.	51

5.6	Time taken by each process for 10 generations, each for population size 1000 and pool size 2000 at work VMs.	51
6.1	(A) MNIST [120] dataset containing 60,000 samples. (B) A sample partition generated using Algorithm 6.1. Colors represent various classes.	56
6.2	Comparison of DiP-SVM with the existing clustering based methods in [40] [132] for local and global solutions on well separable clusters having points from both the classes. (Best viewed in colors)	59
6.3	Comparison of DiP-SVM with the existing clustering based methods in [40] [132] for local and global solutions using non-linear kernel. (Best viewed in colors)	60
6.4	A comparison of the loss in classification accuracy (%) of DC-SVM, CP-SVM and proposed distributed SVM with respect to LIBSVM on publicly available datasets.	62
6.5	A comparison of the training time (seconds) of LIBSVM, DC-SVM, CP-SVM and proposed distributed SVM on publicly available datasets.	62
6.6	Computational and communication efficiency of the proposed approach on the dataset kddcup99 consisting of $\approx 5M$ records.	63
6.7	Performance of classification for mini-batch training of DiP-SVM.	64
7.1	Proposed approach for data partitioning using decision tree along the direction of maximum variability.	67
7.2	Block diagram of the Projection-SVM training over the cluster. Master node contains a sample tree model. The job scheduler evenly distribute the task of training SVMs to compute nodes	69
7.3	Illustration of the working of proposed distributed SVM for well separable classes.	73
7.4	A comparison of the sequential SVM and the proposed distributed SVM on a sample 2D-data which is a mixture of the 20-Gaussian distributions.	74
7.5	A comparison of the classification performance (%) for sequential SVM and proposed distributed SVM. (A) Sample dataset. (B) Performance at various mixture of K -Gaussian distributions. Class labels are assigned randomly. $K = 10, 20, 30, 40, 50, 60$, Number of data points $n = 2400$	74
7.6	A comparison of the loss in classification accuracy (%) of DCSVM, CASVM, DTSVM, and proposed distributed SVM with respect to LIBSVM on various datasets.	76

7.7	A comparison of the training time (seconds) for LIBSVM, DCSVM, CASVM, DTSVM, and proposed distributed SVM on various datasets.	77
7.8	Comparison of the test time of sequential SVM and the proposed approach .	77
8.1	The proposed edge computing-based framework for traffic monitoring . . .	80
8.2	Block diagram of proposed framework for the detection of motorcyclists without helmet.	81
8.3	The sample images of the located motorcycle riders with and without a helmet of various style in different viewpoints.	83
8.4	Architecture of the proposed network HH-Net for head vs. helmet classification.	83
8.5	The activations produced by the various layers of the proposed CNN classifier after training for helmet (top) and head (bottom).	84
8.6	Sample frames from <i>IITH_Helmet_1</i> dataset showing the various difficulties.	86
8.7	Sample frames from <i>IITH_Helmet_2</i> dataset showing the various difficulties.	86
8.8	Sample images and their respective activation maps for the two classes, namely, helmet (followers) and head (violators). [Best viewed in color] . .	88
8.9	Scatter plots <i>IITH_Helmet_1</i> dataset showing distributions of the two classes, namely, helmet (green dots) and head (red dots) in train and test datasets before and after the training. [Best viewed in color]	88
8.10	The architecture of the proposed framework for accident detection. (A) Overview of the framework. It consists of two streams, one for the generation of the collision score using the trajectories of the moving vehicles and the other one for generation of abnormality score using deep representation. (B) A detailed diagram of abnormality detection using deep stacked autoencoder on three modalities, namely, appearance, motion, and joint representation.	91
8.11	The generation of the spatio-temporal video volumes (STVVs). The STVVs are the pixels in the immediate vicinity of a point $p(x, y, z)$ covered by a 3D sliding window of size (w, h, t)	92
8.12	The network topology of the proposed stacked autoencoder used to model the baseline for the normal traffic. The network consists three decoder layers followed by three decoder layers. The reconstruction error is the Euclidean distance of the input and output layers. The output of the middle layers is the latent intermediate representation.	93

8.13	The intersection of two trajectories during an accident. The trajectories of the motorcycle and car intersect each other also there is no further progress in the trajectories of the motorcycle and car, thus considered as a collision. .	94
8.14	Sample frames from the video dataset used to evaluate the performance of proposed approach. The dataset contains videos of accidents during the various environmental conditions such as high sunlight, night, early morning as well as from different cameras and view angles.	96
8.15	The 2-D visualization of the distribution of the STVVs data generated from a sample video. The STVVs during normal and accident are shown using the green cross and red dot, respectively. [Best viewed in color]	97
8.16	ROC curve for accident detection using reconstruction error at various Layers.	98
8.17	ROC curve for accident detection using reconstruction error (RE), one class SVM, and their combination.	99
8.18	ROC curve for accident detection using reconstruction error, one class SVM, and their combination for the appearance, motion, and their joints representations.	99
8.19	Accident detected using proposed approach for different videos. The red region is the predicted accident regions using a single score either by appearance, motion, or intersection of tracks while the green box shows the region decided using final score.	100

Chapter 1

Introduction to large-scale visual computing

In today's digital world, visual information plays a crucial role in various fields such as video surveillance systems, multimedia analysis, virtual reality, robotics, scientific visualization, communication systems, oceanography, analysis of natural life, etc [54]. The application such as video surveillance in a smart city consists thousands of video cameras and needs to analyze a large amount of video footage in order to locate various incidents such as violation of traffic rules, accident incidents, abnormal activities, etc. In such a large video surveillance network, maintaining surveillance facilities using conventional techniques is a tedious and time-consuming task. The existing automated systems for the detection of potential security problems are unable to be used in such a large-scale because of their high computational complexities [19]. Also, conventional standalone machines and sequential modeling algorithms are highly incompetent to harness the benefits of the large visual data for learning a model in reasonable time as visual computing methods are both data and computation intensive. Thus, there is a need for scalable and distributed methods to perform such tasks in real-time at such a large-scale.

1.1 Visual computing

Visual computing involves processing and analyzing of visual data such as images and videos to find semantic patterns that are useful for interpretation. The visual computing methods provide solutions to variety of applications but are extremely complicated and computationally complex. A visual computing framework generally consists of two major tasks, namely, feature representation and modeling. A brief introduction of them is provided here:

1.1.1 Feature representation techniques

An appropriate representation of visual data lead to more accurate decision making by subsequent modeling processes. The basic representation of visual information is pixel. Pixel representation is for human to perceive things in similar manner he/she perceive them in real world. In order to make decision on such information, we need to extract appropriate features (or properties) which have some correlation to the target problem. Data representation has a direct impact on the performance of machine learning techniques. There have been variety of representation techniques proposed till now, which can be categorized as hand engineered feature representation and data driven feature representation techniques as discussed below.

Hand engineered feature representation

Hand engineered feature representation techniques are based on the perception of human experts to extract relevant information. They are based on well-established theories. The widely used techniques such as histogram of oriented gradients (HOG) [26], histogram of optical flow (HOF) [17], scale-invariant feature transform (SIFT) [70], and space-time interest points (STIP) [59] along with a bag-of-visual-words (BoW) approach had achieved a good performance for variety of vision tasks. The hand engineered feature representation techniques can be categorized into flow-based features (like optical flow, particle flow, streak flow), local spatio-temporal features (like spatio-temporal gradients, motion histogram), and trajectory/tracklet.

Data driven feature representation

Data driven feature representation techniques are based on deep nets and automatically learn abstract and complicated representation of the data in a hierarchical manner by passing the data through multiple layers of nonlinear transformations [81]. Some existing automatic feature representation techniques are auto-encoder [36], convolution neural network (CNN) [18], and restricted Boltzmann machine (RBM) [18].

The performance of the representation techniques changes drastically with the domain and nature of application. Finding a better representation of visual data which is also computationally efficient is a big challenging task.

1.1.2 Modeling Techniques

Modeling techniques learn different parameters from the data generated by representation techniques. These parameters are later used for making the decision on the unknown data.

Various machine learning techniques [109, 6, 47, 11, 30] are already existing in the literature for modeling. Generally, they are classified into three categories, namely, 1) supervised learning techniques (e.g., support vector machine, neural network, decision tree, Bayesian network, Fisher's linear discriminant analysis etc.) which require knowledge of ground truth, 2) unsupervised learning techniques (e.g., k-means, k-medoids, etc.) which do not require any ground truth, and 3) semi-supervised learning techniques which require ground truth for few samples. The widespread applications of machine learning are subjected to overcome major issues such as the need of large data sets, the need of ground truth, concept drift in a dynamic environment, and computational complexity.

1.1.3 Challenges in large-scale visual computing

The visual computing presented above suffer from the issues such as computation intensiveness, increased training time of modeling techniques, real-time inference on the resource-constrained devices, high loss in model approximation, high communication overhead, and handling city-scale surveillance network.

Computation intensiveness

The high space and time complexities of the feature representation and modeling techniques make the visual computing applications a computation intensive task because of involvement of large amount of data. The use of such algorithm further makes the situation worse for the applications involving large datasets.

Sequential modeling techniques

Researchers in the past mainly focused only on improving the performance of the vision task and used the sequential algorithms with high space and time complexities. Some existing approaches cannot be implemented in parallel/distributed manner. Those easy to be implemented in parallel/distributed manner are computation intensive and require a pool of computing resources or take longer time to compute thus unable to be used in real-time.

Real-time inference on the resource-constrained devices

The best performing visual computing models are extremely large in size (millions of parameters) and thus need significant computing resources which make them unfit for real-time usage on the resources-constrained devices such as smart phones and embedded cards. In order to fit them on limited memory devices for real-time speed, it is important to compress and accelerate the model.

High loss in model approximation

Model approximation is applied to scale the sequential and computation intensive algorithms using some kind of surrogate functions that mimics the functionality of the original function but are comparatively less computationally complex. However, existing model approximation approaches suffer from a high loss in the effectiveness. Thus it is highly important to reduce the loss occurred because of model approximation in order to retain the performance.

High communication overhead

The existing distributed learning algorithms suffer from the large flow of the data on the communication network thus making the entire process a time-consuming task. Also, the high congestion near the central computing nodes may lead to a single point of failure. Thus it is expected to design such algorithms which transfer minimal data over the communication link.

Handling city-scale surveillance network

The city-scale video surveillance networks involve thousands of cameras generating gigabytes of data per second. Using a centrally computing facility are not a good choice for performing vision task because of the two reasons. First, the transmission of the data from the camera to central facility will increase the latency. Second, the congestion near the central facility will increase the frame dropping rate. Also, both the cases increase the overall response time.

1.2 Issues addressed in this thesis

The existing computer vision approaches suffered from one or more of the above issues. In this thesis, we explore various ways to build scalable and distributed methods for large-scale visual computing that address one or more of the these issues using model approximation and distributed implementation of computer vision algorithms. A scalable method *Fast-BoW* is propose for reducing the computation time of bag-of-visual-words (BoW) feature generation for both hard and soft vector-quantization with time complexities $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$, respectively. The process of finding the closest cluster center is approximated using a softmax classifier accelerated by reducing the number of multiplications through weight sharing which can be used for both hard and soft BoW encoding. Further, we increase the effectiveness and efficiency of video activity representation by exploiting the structural information in a video activity among the various entities or same entity over the

time which is generally ignored by BoW representation. The interactions of the entities in a video are formulated as a graph of geometric relations among space-time interest points.

The long training time of kernel SVM from large datasets is reduced by using the distributed genetic algorithm for the fast optimization of the SVM objective function on a GPU enabled cluster. The effect of the distribution preserving data partitioning approach is investigated to obtain local decision boundaries in close agreement with the global decision boundary. It is found that retaining first and second statistics reduces the chance of missing important global support vectors. Further, to reduce both training and prediction time, we propose a subspace partitioning, where a decision tree is constructed on a projection of data along the direction of maximum variance to obtain smaller partitions of the dataset. On each of these partitions, a kernel SVM is trained independently thereby reducing the overall training time. Also, it results in reducing the prediction time significantly.

Also, we explore edge computing based framework for deploying large-scale visual computing applications to reduce the latency and the communication overhead in the camera network. Two most desired surveillance applications, namely, detection of motorcyclists without a helmet and accident incident detection are designed and implemented using this framework. Efficient model are designed for detecting accidents and motorcyclists without helmet in real-time on edge devices. The overall computation is distributed to the edge device and the central servers to minimize the data transmission over the network while strengthened the overall performance.

1.3 Organization of the thesis

The thesis is organized as follows. The Chapter 2 presents a review of existing large-scale methods for visual computing and highlight the relevant research issues. The first approach for scaling bag-of-visual-words (BoW) generation is discussed in Chapter 3. In Chapter 4 a novel graph representation of local feature descriptors is presented for abnormal activity recognition. A distributed genetic algorithm is presented in Chapter 5 for fast training of the kernel SVM on GPUs. A distribution preserving partitioning based approach is investigated in Chapter 6 for distributed training of the kernel SVM over a cluster. Also, a subspace partitioning based approach for distributed kernel SVM from large-scale dataset is presented in Chapter 7. An edge computing based framework is presented in Chapter 8 for city-scale traffic monitoring along with robust and computationally efficient methods for detection of helmet-less motorcyclists and accident detection. Finally, in Chapter 9, a summary of the research work carried out as part of this thesis is presented with directions for future work.

Chapter 2

Review of scalable and distributed methods for visual computing

The challenges in the large-scale visual computing presented in previous chapter have been studied in the vision and big data communities. The main issues encountered are computation intensiveness, increased training time of modeling techniques, real-time inference on the resource-constrained devices, high loss in model approximation, high communication overhead, and handling city-scale surveillance network. The challenges can be broadly split into three major tasks: 1) challenges associated with large-scale feature representation, 2) challenges associated with the large-scale modeling and prediction, 3) challenges associated with the system architecture for large-scale surveillance network.

The rest of the chapter is organized as follows. Various feature representation techniques are discussed in Section 2.1. We describe the various modeling techniques used in visual computing in Section 2.2. The various system architectures are discussed in the Section 2.3. Section 2.4 presents the existing work on the various large-scale visual computing applications. Finally, the observations arising from the literature review are presented in Section 2.5 and the summary is presented in Section 2.6.

2.1 Large-scale feature representation

In computer vision, the success of a task highly depends on the feature representation technique used. On the bases of technique used, the existing feature representation methods are generally classified into two categories, namely, traditional feature representation and deep feature representations. On the basis of level of granularity, the feature representations are classified into two categories, namely, low-level and high-level feature representations. The high-level features derived from the whole chunk of visual data and thus can be used directly by the modeling techniques. While the low-level features are generally

extracted from a small region of the entire image/videos. For example, scale invariant feature transform (SIFT), space-time interest points (STIPs), and improved dense trajectories. All these features extract 2D or 3D interest points in image and videos, respectively, and then in the close vicinity of the interest points, they extract appearance and/or motion based histogram called local feature descriptors. The number of interest points extracted from different chunk of visual data are different thus leads to variable length code. Before applying any modeling technique, first we need to learn an aggregate representation from the low-level feature descriptors. Some of the techniques for aggregate feature representation are bag-of-visual-words (BoW), Fischer vector (FV), vector of locally aggregated descriptors (VLAD), sequential models, dictionary learning and sparse coding, etc. Bag-of-visual-words (BoW) approach is one of the widely used aggregate feature representation technique because of its simplicity. The performance of the BoW features increase when increasing the vocabulary size. However, large vocabularies increase the time taken for BoW feature generation and thus make them unfit for real-time usage specially on the resource-constrained devices.

2.2 Large-scale modeling

Training a model from large collection of data is not a trivial task as iterative computations in most of the machine learning algorithms are computation intensive and often extremely difficult to be parallelize or distribute. Here, we highlight current research efforts and related challenges in scalable machine learning models widely used for large-scale visual computing, namely, support vector machine (SVM), k-nearest neighbour (kNN) and neural networks (NNs).

2.2.1 Support vector machines (SVM)

Support vector machines are well-known for their excellent generalization capability and widely used in variety of pattern recognition applications. Learning a SVM corresponds to solving a quadratic programming optimization whose space and computational requirements increase with an increase in number of training samples. Thus, in order to train SVM on large scale data, it requires a distributed version with trade-off between computational accuracy and communication overhead. Many parallel and distributed implementations of SVM have been proposed in the literature [53, 124]. Broadly, we can group them into four categories, namely, parallel [137], distributed [12, 72], heterogeneous (MapReduce based) [14, 130, 111, 6], and GPU based [39]. In [82], Vazquez *et al.* proposed a distributed

support vector machine in which local support vectors (LSVs) are calculated on each subset. The set of global support vectors (GSVs) is the union of all the LSVs. Then the GSVs are merged with each training subset and the process is repeated until convergence (i.e. no change in the empirical risk). However, the size of the subsets increases with the number of iterations which contributes to increased learning time. Also, at each time, LSVs are collected from each node to form the GSVs and then these GSVs are broadcasted to all the nodes which further increases the communication overhead. This also results in high redundancy among LSVs across all the nodes. Similar approaches have been proposed by Lu *et al.* [72] for strongly connected networks (SCNs). Catak *et al.* [14] proposed a MapReduce-based implementation of the same methodology in the cloud environment in order to improve scalability and parallelism of training phase by splitting training dataset into smaller subsets as shown in Fig. 2.1.

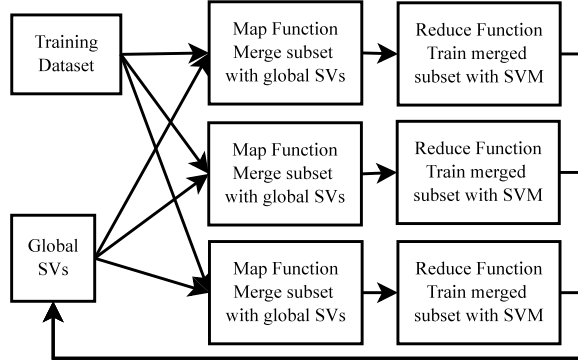


Figure 2.1: Schematic of cloud SVM architecture [14].

In [32], Graf *et al.* proposed cascade SVM, where the training samples are divided hierarchically into subsets and the training starts at the top nodes where each subset is then trained a sequential SVM (referred to as a subSVM). The support vectors of two smaller subSVMs are then passed down as input to next level subSVM and this process is repeated until we reach the final SVM at the bottom where the global SVM model is obtained as shown in Fig. 2.2. Sun *et al.* [111] implemented the same cascade SVM architecture with the help of MapReduce and Twister. A similar approach is taken by Vazquez *et al.*'s [82], where the size of training data at each node increases with each subsequent iteration causing high communication overhead. The communication overhead is the amount of data transfer over the communication network from one node to another during the training process.

Hsieh *et al.* [40] instead propose a divide-and-conquer (DC-SVM) approach to solve the kernel SVM problem. DC-SVM is similar to cascade SVM with two major differences: 1) It uses K -means clustering to partition the dataset instead of sequential or random partition. 2) It passes all of the training vectors and solutions from one level to next level,

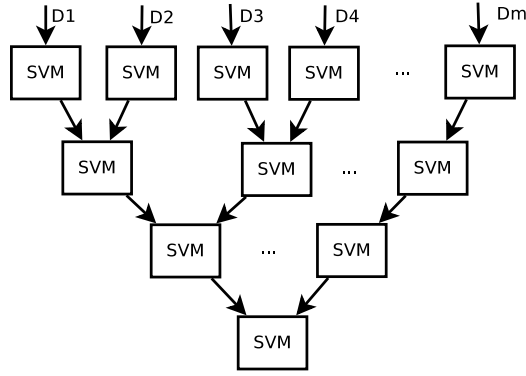


Figure 2.2: Training flow of cascade SVM [32].

instead of only SVs. However, the disadvantage of DC-SVM is that at the last level, it operates a single SVM on the whole training dataset which is essentially quite slower and non-scalable for larger datasets. A similar approach is also proposed by Alham *et al.* [6] for large scale image annotations using MapReduce. The entire training data is partitioned into smaller subsets and each of these subsets are allocated to separate Map tasks. Each *Map* task optimizes the partition in parallel. The outputs, Lagrangian multiplier α arrays and bias b values, obtained from each *Map* task are then joined in the *Reduce* task in order to produce the global Lagrangian multiplier α array and the average bias b . Even though this model reduces training time, the blind partitioning of the sample dataset results in different classification performance with different initialization for the partitions.

To reduce the communication involved in the methods discussed above, You *et al.* [132] proposed a communication avoiding SVM (CA-SVM) for shared memory architecture by combining several approaches like the cascade SVM, DC-SVM etc. Basically, a divide and conquer filter selects the LSVs and their corresponding Lagrange multipliers α values from one level in order to initialize next level Lagrange multipliers α in cascade SVM which results in faster convergence. The partitions are obtained as the clusters resulting from a balanced k -means clustering algorithm. This approach performs better when the dataset contains well-separated clusters and each cluster contains points from the corresponding classes. However, its performance degrades in the case of overlapping clusters and also if a cluster contains examples belonging to only one class.

Yu *et al.* [134] make an attempt to train the SVM on a large dataset which is suitable only for well separated low dimensional data. This method uses a hierarchical clustering based tree called CF-tree where data at the root of the tree is first used to train an initial model which is refined successively. This refinement is carried out at each subsequent lower level by considering the clusters near to the decision boundaries in the upper level to train an SVM from the beginning. This is in stark contrast to our proposed approach where

the Lagrangian multipliers of the SVM in a lower level are initialized from the upper level Lagrangian multipliers that is empirically shown to help in the faster convergence of the SVM.

2.2.2 K-nearest neighbours (k-NNs)

In computer vision, k-nearest neighbors (k-NN) method is used widely for classification and retrieval tasks because of their simplicity and interpretability. For a large number of samples, the k-NN classifier asymptotically converges to Bayes optimal classifier [29]. Especially, the k-NNs achieve the best classification accuracy and retrieval performance for a large number of labels. Since k-nearest neighbors method needs to store all samples and the prediction involves searching into these samples, making it unsuitable for large databases. To date, several approximate nearest neighbor search techniques have been proposed to reduce the space and time complexity of the nearest neighbor search. The existing techniques can be grouped into three categories, namely, tree-based approaches [10, 131], hashing/binary embedding approach [61, 101], and vector quantization approach [49]. The approaches in each category have their advantages and limitation and generally applied in an unsupervised setting.

2.2.3 Neural networks (NNs)

The neural network is also a popular and widely used classification technique. In the past few decades, neural networks have emerged to be powerful computational models to address complex pattern classification, medical imaging, and speech recognition [140]. The deep neural network plays an important role in visual computing. In deep learning, neural networks are trained with multiple layers where each layer represents some kind of low-level representation. Neural networks are well-known for their adaptation capability and give good accuracy when they are trained on large scale data. Since neural networks require huge computational resources, the conventional uni-processor or high-speed computers are failed to meet the required speed and storage capacity. Distributed implementation can be a solution directive to overcome time and space limit of neural networks. Several attempts have been made in this direction. Z. Liu *et al.* [69] have trained a back-propagation neural network with Ada-boosting using the map-reduce framework in order to increase the efficiency and precision of mobile data classification having natural properties like large-scale, inter-class similarity and noisiness. A similar approach is also developed by H. Zhang *et al.* [140] for parallel implementation of multilayer neural networks based on map-reduce over the cloud.

The above discussed implementations show that scaling machine learning algorithms become an important constituent to strengthen modeling. The distributed implementation of machine learning is one of the possible solution to achieve learning on large-scale visual data. In the early stage, researchers casually distributed tasks over multiple machines in order to increase the performance. However, next step towards the distributed implementation require trade-off between computational accuracy and communication overhead in the computing cluster.

2.3 System architecture for large-scale surveillance network

Designing an effective system architecture for deploying large-scale visual computing applications in a city-wide video surveillance network is a challenging task. Here, we investigate existing visual computing framework proposed for deploying a variety of visual computing applications and highlight their challenges.

Park *et al.* [84] presented Ubiquitous-city, a GIS image processing platform using cloud computing in order to find and select optimal computing resources for applications and run virtual machines to execute the applications. Ubiquitous-city enables citizens to access the converged information anywhere and anytime using ubiquitous IT. Ubiquitous-city requires a lot of computing power because large amount of data need to be processed in real-time. It consists of OpenNebula, a cloud computing framework [95] and Haizea, a virtual machine job manager (VMJM). The massive amount of data generated by parallel air pollution map generation is processed efficiently using this platform.

Abdullah *et al.* [3] presented a framework for stream processing in cloud that is capable of detecting vehicles from the recorded video streams for large-scale vehicular traffic monitoring. This framework provides an end-to-end solution for video stream capture, storage, and analysis using a cloud based graphics processor unit (GPU) cluster. It empowers traffic control room operators by automating the process of vehicle identification and finding events of interest from the recorded video streams. An operator only specifies the analysis criteria and the duration of video streams to analyze. These video streams are then automatically fetched from the cloud storage, decoded, and analyzed on a Hadoop based GPU cluster without operator intervention. It reduces the latency in video analysis process by porting its compute intensive parts to the GPU cluster.

An automatic license plate recognition system is presented by Chen *et al.* [19] using cloud computing in order to realize massive data analysis, which enables the detection and tracking of a target vehicle in a city with a given license plate number. It realizes a

fully integrated system with a surveillance network of city scale, automatic large scale data retrieval and analysis, and combination of pattern recognition in order to achieve contextual information analysis.

Ryu *et al.* [96] have presented an extensible video processing framework over apache Hadoop framework in order to do parallel video processing in a cloud environment. This framework employs ffmpeg for a video coder, and opencv for an image processing engine. To optimize the performance, it exploits map-reduce implementation which helps to minimize video image copy. The ffmpeg source code is then modified and extended to access and exchange essential data and information with hadoop effectively. For demonstration, face tracking system is then implemented on top of this framework which traces the continuous face movements in a sequence of video frames.

Zhang *et al.* [139] have proposed a hybrid cloud model for video surveillance system with mixed-sensitivity video streams. The hybrid cloud is used to address the security issues by keeping sensitive data in the private cloud, while relieving seasonal workload by pushing computation to the public cloud. To enhance usability and reduce the cost, a middle-ware is used that seamlessly integrates the private cloud with public cloud and scheduled the tasks effectively. A stream processing model in this hybrid cloud optimizes overall monetary cost to be incurred on the public cloud with the constraints of resources, security and Quality-of-Service (QoS).

The visual computing framework mentioned above make use of centralized or cloud systems in order to overcome the problem of storage and computing resource intensiveness of visual computing applications. However, the challenges such as high communication overhead, congestion near the centralized nodes, and high latency due to low bandwidth need to be overcome.

2.4 Large-scale visual computing applications

Digital video surveillance systems are ubiquitously deployed in public places to ensure security and safety. According to the British security industry association, approximately 4 million to 5.9 million cameras are deployed in UK [3]. This widespread use of surveillance systems in roads, stations, airports or malls has led to a huge amount of data that needs to be analyzed for safety, retrieval or even commercial reasons [52]. Automation of various task for a city surveillance system is very important for security applications, where it is difficult even for trained personnel to reliably monitor scenes with dense crowd or videos of long duration [52]. Here, we describe few large-scale visual computing applications.

2.4.1 Abnormal activity recognition

An anomalous event in a crowd is an event which do not confirm the normal appearance or dynamics of crowd. An appearance-related anomaly would be, e.g. a bicycle passing through a crowd. Moreover, sudden changes in velocity, like an abrupt increase of its magnitude and the dispersion of individuals in the crowd indicates that something unusual and potentially dangerous may have occurred [52]. In the past decade, a considerable amount of literature is focused on the abnormal activity recognition in surveillance videos [9, 13, 50, 92, 127, 98]. The detailed surveys in [63, 86] enlighten the progress on this topic in last decades.

The most recent methods focus on both appearance and motion anomalies at local and global scale. Space-time interest points have been explored recently for abnormal activity recognition in surveillance videos [20]. In [20], Cheng *et al.* detect local and global anomalies via hierarchical feature representation using bag-of-visual-words (BoVW) and Gaussian process regression. The extraction of normal interactions from training videos is formulated as the problem of efficiently finding the frequent geometric relations of the nearby sparse space-time interest points (STIPs). In [121, 122], Wang *et al.* use a standard bag-of-features approach to construct separate vocabularies of 4000 visual-words for each type of low level descriptors. The low level descriptors encode information of trajectory shape, appearance using HoG, local motion using HoF, and gradient of horizontal, and vertical components of optical flow using motion boundary histograms (MBH). However, above methods use bag-of-words based approach that do not consider the geometric relationships among salient points.

2.4.2 Helmetless motorcyclists detection

Since, motorcycles are an affordable and daily mode of transport, there has been a rapid increase in motorcycle casualties due to the fact that most of the motorcyclists do not wear the helmet which makes it an ever-present danger every day to travel by motorcycle [25, 107, 116]. In the last couple of years alone, most of the deaths in accidents are due to damage in the head [8]. Because of this fact, wearing a helmet is mandatory according to the traffic rules, violation of which attracts hefty fines. In spite of this fact, a large number of motorcyclists do not follow the traffic rules. The manual strategies to identify these violators have several drawbacks such as interrupted traffic flow, unpleasant weather conditions for police personnel, etc. Existing video surveillance based methods are passive and require significant human assistance. In general, such systems are infeasible due to involvement of humans, whose efficiency decreases over long duration. Automation of this

process is highly desirable for reliable and robust monitoring of these violations as well as it also significantly reduces the amount of human resources needed. Presently, all major cities across the world already deployed extensive video surveillance network at public places to keep a vigil on a wide variety of threats. Thus the solution for detecting violators using the existing infrastructure is also cost-effective.

To date, several researchers [21, 22, 102, 89, 103, 25, 107, 116] have tried to tackle the problem of detection of motorcyclists without helmet by using different methods from computer vision. But they have not been able to accurately identify motorcyclists without helmets under challenging conditions such as occlusion, illumination, poor quality of video, varying weather conditions, etc. Some of the reasons for the poor performance of existing approaches are: (i) the use of not so efficient handcrafted features for object classification, (ii) the consideration of irrelevant objects against the objective for the detection of motorcyclists without helmet, and (iv) most of the existing methods are computationally complex and thus not suitable to be used in real-time.

2.4.3 Accident detection

The growing size of cities and increasing population mobility have determined a rapid increase in the number of vehicles on the roads, which has resulted in many challenges for road traffic management authorities. Among them, road accidents require immediate attention to reduce the loss of life and properties. Traffic accidents caused an estimated 1.2 million deaths in 2004, with 50 million people injured [28]. Over the recent years, researchers from both industry and academia have been working to develop automatic detection methods using computer vision and pattern recognition techniques, but the level of current technology is still limited to apply them in the real world. Devising vision-based algorithm for this task is very challenging. In practice, the performance of computer vision based traffic accident detection algorithms can be challenged by many factors [135, 128, 91]. These factors include imaging conditions (varying illumination and changing weather conditions), environments (urban, highways), as shown in Fig. 2.3.

As also pointed out by Yun *et al.* [136], the existing methods for traffic accident detection developed till date can be categorized into three approaches:

- **Modeling of traffic flow patterns:** In this category, the typical law-full traffic patterns (namely, go-straight, U-turn, right-turn, etc.) are modeled as baseline [113, 41] and any deviation from this model is considered as an abnormal traffic event. This approach will work only when the normal traffic pattern appears at a fixed region repeatedly, thus unable to detect collisions which are essential to accident detection.

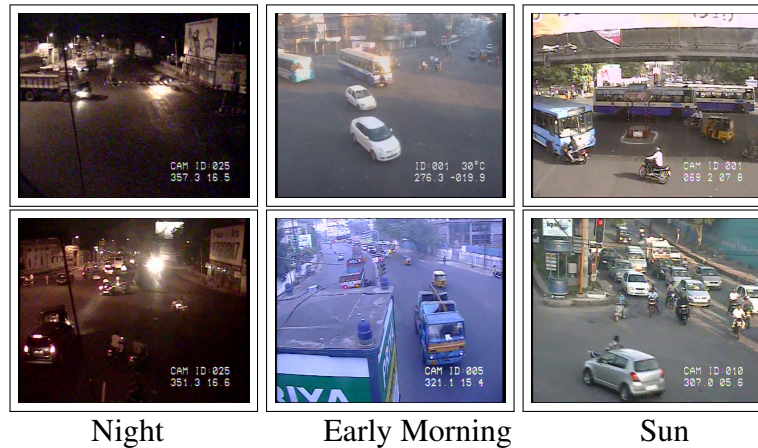


Figure 2.3: The sample video frames, showing various difficulties in the collected video dataset for accident detection.

- **Analysis of vehicle activities:** The methods in this categories first detect the moving vehicles and then extract motion features such as the distance between two vehicles, acceleration, direction, etc. of a vehicle from moving vehicles' tracks [42, 60, 97, 5, 44, 55, 41]. However, unsatisfactory tracking performance in crowded traffic scenes becomes their bottleneck and limits their usage.
- **Modeling of vehicle interactions:** These methods have been inspired by sociological concepts and model the interaction among vehicles and detect accidents [75, 110]. However, a large number of training data and use of speed change information alone limit the performance of these methods.

These existing methods make use of motion or track of moving objects and simply try to define a normal baseline (many time using pre-decided threshold only) and any unknown event not obeying this baseline is simply declared as an accident. Although, the deviations in motion parameters give useful pre-collision information but do not sufficient for accident detection.

2.5 Observations from the review

The entire review of large-scale visual computing for various surveillance applications presents three aspects of the visual computing which are large-scale feature representation, large-scale modeling techniques, and system architecture for large-scale surveillance network. Within feature representation approaches, aggregated descriptors using bag-of-visual-words (BoW) representations with large-vocabularies are shown to be effective for representing image and videos but computationally time consuming because of high-

dimensional local-descriptors and large number of visual words in the vocabulary. For fast BoW generation for large vocabularies, existing approaches use tree hierarchy of the vocabulary elements which results into a significant fall in the effectiveness of the BoW features thus hypothesize to predict the cluster index efficiently instead of a searching the closest cluster hence propose for fast generation of BoW features while sustaining the effectiveness of the generated features. Also, we hypothesize for a novel feature representation which is not only effective in recognition but also helps in better localization of the action/activities in a video.

Many of the modeling techniques presented in the review like kernel SVM are hard to parallelize or distribute and cannot leverage the benefit of distributed computing systems while learning from large-scale datasets. Existing approaches for distributed kernel SVM training suffer from high loss in accuracy and the communication overhead. The core issues listed in the distributed learning of SVM are the use of sequential minimal optimization (SMO) algorithm, loss of global support vectors because of random partitioning, increased training time because of non-separable and complex distribution of the class data, and increased prediction time because of large number of support vectors.

Apart from scaling individual methods used in visual computing, an efficient distributed system architecture is also equally important to deploy visual computing methods in a city-wide surveillance camera network because a centralized system suffers from the high communication overhead and hinders their real-time performance.

2.6 Summary

In this chapter, the existing literature on the large-scale visual computing is reviewed covering large-scale feature representation, large-scale modeling, and system architecture for large-scale surveillance network. Most traditional representations are based on aggregated descriptors of local features that holistically describe actions. However, such representations are using large vocabularies which increase their computation time. The deep representations are supervised and computation intensive thus require a large number of labeled examples and a pool of computing resources enabled with GPUs. Also, many classification techniques such as kernel support vector machine are hard to compute in a distributed environment. Visual computing brings automation in various areas of applications but always suffers from data intensiveness and lack of computing resources. This review provides an overview of existing methods for large-scale visual computing, investigates their challenges and identifies opportunities to enhance existing visual computing methods for large-scale applications.

Chapter 3

Fast-BoW: Scaling bag-of-visual-words (BoW) generation

As discussed in the last chapter, the bag-of-visual-words (BoW) is a widely used method for unsupervised representation of visual data based on the local feature descriptors [43, 46, 141, 142, 74, 66, 105] but it is not suitable for large vocabularies because of its high computational complexities. In this chapter, we present *Fast-BoW*, a scalable method for BoW generation for both hard and soft vector-quantization with time complexities $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$, respectively, where $|\mathbf{h}|$ is the size of the hash table used in the proposed approach and k is the vocabulary size. The process of finding the closest cluster center is replaced with a softmax classifier which improves the cluster boundaries over k -means and also can be used for both hard and soft BoW encoding. Further, to make the model compact and faster, the real weights of the softmax classifier are quantized to integer weights which can be represented using few bits (2–8) only. Then a hash table of the quantized weights is maintained to reduce the number of multiplications which makes the process further faster. The proposed approach is empirically evaluated on several public benchmark datasets. The experimental results outperform the existing hierarchical clustering tree-based approach by ≈ 12 times.

The BoW generation process involves two phases, 1) *vector quantization for vocabulary generation* that is typically performed by k -means clustering algorithm, and 2) *frequency histogram generation* using nearest neighbour search. During the vector quantization, the goal is to construct a vocabulary \mathcal{V} with a small average quantization error. Let $\mathcal{L} = \{\mathcal{F}_1, \dots, \mathcal{F}_n\}$ be the set of local feature descriptors $\mathcal{F}_i = \{\mathbf{f}_1, \dots, \mathbf{f}_{m_i}\}$, $\mathbf{f}_i \in \mathbb{R}^d$ for n training videos, where d is the dimension of the local feature descriptor. The input to the vector quantization algorithm is a set of m vectors $\mathcal{Q} = \{\mathbf{f}_1, \dots, \mathbf{f}_m\}$, $\mathbf{f}_i \in \mathbb{R}^d$ where $\mathcal{Q} \subseteq \{\mathcal{F}_1 \cup \dots \cup \mathcal{F}_N\}$. The output of the algorithm is a set of k vectors $\mathcal{V} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}$, $\boldsymbol{\mu}_i \in \mathbb{R}^d$, where $k \ll m$. The set \mathcal{V} is called the vocabulary. We say that \mathcal{V} is a good vocabulary for \mathcal{Q}

if for most $f \in \mathcal{Q}$ there is a representative $\mu \in \mathcal{V}$ such that the Euclidean distance between f and μ is small. The average quantization error of \mathcal{V} with respect to \mathcal{Q} is defined as

$$J(\mathcal{V}, \mathcal{Q}) = \mathbb{E} \left[\min_{1 \leq j \leq k} \|F - \boldsymbol{\mu}_j\|^2 \right] = \frac{1}{m} \sum_{i=1}^n \min_{1 \leq j \leq k} \|\mathbf{f}_i - \boldsymbol{\mu}_j\|^2, \quad (3.1)$$

where $\|\cdot\|$ denotes Euclidean norm and the expectation is over F drawn uniformly at random from \mathcal{Q} . The k -optimal set of visual words is defined to be the vocabulary \mathcal{V} of size k for which $J(\mathcal{V}, \mathcal{Q})$ is minimized. Typically, k -means algorithm is used for this whose per epoch computational complexity is $O(mdk)$. The average quantization error for a general sets of unit diameter in \mathbb{R}^d , is roughly $k^{-2/d}$ for large k [33]. For high dimensional local feature vectors where d is high, it becomes too time consuming. For instance, if $d = 100$, and A is the average quantization error for k_1 visual words, then to guarantee a quantization error of $A/2$, one needs a vocabulary of size $k_2 \approx 2^{d/2}k_1$: that is, 2^{50} times as many visual words just to halve the error. In other words, vector quantization is susceptible to the same curse of dimensionality that has been the bane of other non-parametric statistical methods.

However, increasing the vocabulary size k increases the time to generate the frequency histograms which requires $O(dk)$ real valued vector-vector multiplications per local feature descriptor in a video. As in real-world application, this computation is performed on the fly and thus it effects their real-time performance. The typical solution to this problem is to maintain tree hierarchy of the clusters. The use of a tree leads to $O(d \log_2 k)$ but invariably results in significant fall in the effectiveness of the generated BoW features. Dasgupta *et al.* [27] proposed a set of hierarchical random projection trees for vector quantization and subsequent search of the right subspace by traversing each tree and finally making a consensus. This approach reduces the time complexity in tree construction in comparison to other tree methods. But the use of several trees increases the time of BoW generation. Also, due to hard splitting criteria, the projection tree-based algorithms suffer from the high loss in classification accuracy. The proposed *Fast-BoW* addresses both issues, namely, loss in accuracy and computational time by learning probability distribution of the clusters at each level of the tree and then applying quantization and hashing to reduce the vector operations.

The remainder of the chapter is organized as follows. Section 3.1 presents the proposed *Fast-BoW* for fast generation of the BoW. Section 3.2 discusses the experimental setup, dataset, and performance of *Fast-BoW*. Finally, we conclude in section 3.3.

3.1 Fast cluster prediction using softmax with quantized weights

As shown in the Fig. 3.1, the proposed approach first uses the clustering algorithm (like k -means) for vector quantization of the training vector space into the pre-defined number of clusters. Then to learn the probability distribution of each cluster, we train a soft-max classifier. The class labels for the soft-max are the cluster index of each point received from the clustering algorithm. Also, the large weight matrices have a significant redundancy that can be avoided by applying model compression techniques. We exploit this fact to reduce the memory and computation time by applying weight quantization and hashing. The weight quantization reduces the number of levels (i.e. unique floating point numbers) in a weight matrix and the hashing reduces the number of floating point multiplications needed for a *matrix-vector* or *vector-vector* multiplication.

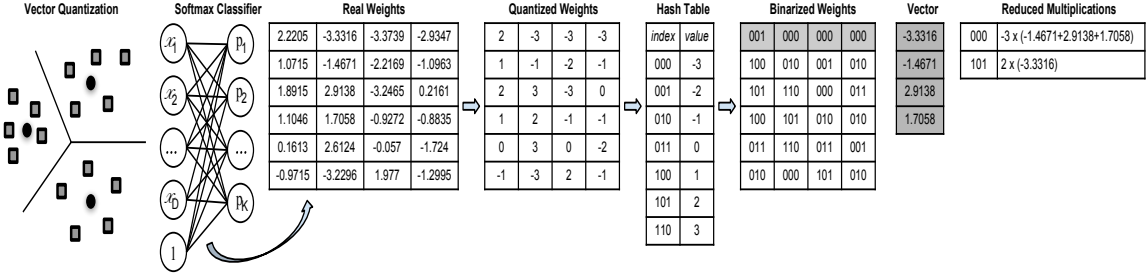


Figure 3.1: Scaling the process of BoW generation.

3.1.1 Learning probability distribution of the clusters

First, we learn k clusters using k -means on the local feature descriptors. As the cluster learning from a large number of local feature descriptors is time-consuming, we first use a small sample to learn the initial clusters $\{\mu_1, \dots, \mu_k\}$ and later refinement the centers using the remaining points only once as suggested by Raghunathan *et al.* [88]. Let, for i^{th} local feature descriptor \mathbf{f}_i , μ_j^i be the closest center then it updates the μ_j^i as

$$\mu_j^i = (1 - \eta)\mu_j^i - 1 + \eta\mathbf{f}_i, \quad (3.2)$$

where, $\eta = \frac{3k \log_3 m}{m}$ gives the best results. After successfully learning the means of the k clusters, the next task is to learn the probability distribution for soft/hard cluster assignment. This can be achieved using a Gaussian mixture model (GMM). However, GMM's training and prediction are time-consuming and also speed-up of them is also not so easy. Thus, we

learn a softmax classifier on the given data where the labels of each vector are the cluster index previously given by the k -means. Doing so helps in learning the soft boundaries for the clusters that can be applied to both types of hard and soft BoW generations. The probability of a local feature descriptor \mathbf{f}_i belongs to j^{th} cluster is

$$p(y_i = j | \mathbf{f}_i; \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}_j^T \mathbf{f}_i}}{\sum_{j=1}^k e^{\boldsymbol{\theta}_j^T \mathbf{f}_i}}. \quad (3.3)$$

The $\boldsymbol{\theta}$ is learned by maximization of the class cross entropy. The object function for the softmax classifier along with regularization is

$$J(\boldsymbol{\theta}) = -\frac{1}{n} \left[\sum_{i=1}^n \sum_{j=1}^k I(j = y_i) \log \left(\frac{e^{\boldsymbol{\theta}_j^T \mathbf{f}_i}}{\sum_{j=1}^k e^{\boldsymbol{\theta}_j^T \mathbf{f}_i}} \right) \right] + \frac{\lambda}{2} \sum_{i=1}^k \sum_{j=1}^d \boldsymbol{\theta}(i, j)^2. \quad (3.4)$$

The loss function in Equation (3.4) is solved using scaled conjugate gradient[78].

3.1.2 Weight quantization and hashing

The parameter $\boldsymbol{\theta}$ contains $(k.d)$ real numbers and the computation of $p(y_i = j | \mathbf{f}_i; \boldsymbol{\theta})$ requires computation of k inner product of d -dimensional real-valued vectors. Although, its cheaper than the k -means and GMM, it still poses significant challenges. Also, a known fact is that the $\boldsymbol{\theta}$ contains significant redundancy and the operations on real numbers (generally represented using 32-64 bits) are much more complex than the integer-real numbers. In order to exploit these facts, first we apply quantization on the values of the parameter $\boldsymbol{\theta}$ and then maintain a hash table \mathbf{h} where the quantization and hash functions are defined as follows:

Definition 3.1. *The quantization function $q : \mathbb{R} \rightarrow \mathbb{Z}$ on a real scalar parameter $\theta \in \mathbb{R}$ is defined as*

$$z = q(\theta) = \left\lfloor \frac{\theta}{\max(\text{abs}(\boldsymbol{\theta}))} \times L \right\rfloor, \quad (3.5)$$

Definition 3.2. *The hash function $h : \mathbb{Z} \rightarrow \mathbb{Z}^*$ on a integer key $z \in \mathbb{Z}$ is defined as*

$$\mathbf{h}(z) = z + L, \quad (3.6)$$

where, $\lfloor \cdot \rfloor$ denotes the nearest integer and L is a free parameter such that $|\mathbf{h}| = 2L + 1$ is the size of the hash table. The Algorithm 3.1 provides the pseudocode of the procedure

of the learning softmax parameters, quantization, and generation of hash table.

Algorithm 3.1 Fast-BoW: Gen_Vocabulary

Input: \mathcal{Q} : Set of local feature descriptors for vocabulary generation, m : Cardinality of the \mathcal{Q} , i.e., $|\mathcal{Q}|$, k : Size of the vocabulary i.e., $|\mathcal{V}|$

Output: θ^* : Softmax parameters, \mathbf{h} : Hash table

gen_vocabulary

$[\mathcal{V}, idx] = k\text{-means}(\mathcal{Q}, k)$;

$model = softmax(X = \mathcal{Q}, Y = idx)$;

$\theta = model.\theta$;

$H = \text{Unique values in the range of } q(\theta)$;

$\theta^* = h(q(\theta))$; {Replace the values of the θ with its hash value (i.e. indexes in \mathbf{h})}

Once, we get \mathbf{h} and θ^* , then the BoW histogram is generated using Algorithm 3.2 where for each local feature descriptor, it first groups and then does summation of its values in a hash bucket 's' according to similar parameter indexes in respective θ_j^* 's then do the multiplication of the real values vector 's' and the quantized integer-valued vector \mathbf{h} which can be represented using less number of bits. Doing so reduces the number of multiplications to less than $|\mathbf{h}|$ which also contains one operand as short integer thus speeds up the entire process and the complexity reduces to $O(|\mathbf{h}|k)$ from the $O(dk)$.

Algorithm 3.2 Fast-BoW: Gen_Histogram

Input: $\mathcal{F} = \{f_1, \dots, f_m\}$:Set of m local feature descriptors of a video, \mathbf{h} :Hash table, θ^* :Hash values of parameters, k :#Clusters, d :#Dimensions

Output: \mathbf{x} :Global BoW feature descriptor

gen_BoW

for $i = 1 : m$ **do**

$max_idx = -1$; $max_sum \leftarrow -\infty$;

for $j = 1 : k$ **do**

$\mathbf{s} \leftarrow \mathbf{0}$; $sum \leftarrow 0$;

for $l = 1 : d$ **do**

$S_{\theta_{j_l}^*} += f_{il}$;

end for

for $l = 1 : |\mathbf{h}|$ **do**

$sum += \mathbf{s}_l \times \mathbf{h}_l$;

end for

if $sum > max_sum$ **then**

$max_sum \leftarrow sum$; $max_idx \leftarrow j$;

end if

end for

$\mathbf{x}_{max_idx} += 1$;

end for

3.1.3 Hierarchical tree for hard BoW generation

In the hard BoW generation, we need to find most probable cluster only unlike soft BoW where we need to compute the probabilities with all the clusters. Thus to further speed up and sustain the classification accuracy of the simple hierarchical trees, we apply similar techniques as discussed above. At each internal node, we learn and maintain a hash table of softmax classifier’s parameters with $k = 2$. Thus, the final complexity of finding a cluster reduces to $O(|h| \log_2 k)$ from $O(|h| k)$ for soft clustering and $O(d \log_2 k)$ for simple hierarchical trees.

3.2 Experiments and results

All the methods are implemented in C++. The experiments are conducted on a machine with Intel(R) Xeon(R) CPU E5620@2.40GHz processor and 32GB RAM. We conducted experiments on various large-scale and challenging video datasets for human action recognition, namely, *KTH* [100], *HMDB51* [58] and *UCF101* [108]. The space-time interest points (STIP) [59] are used as the local features descriptors for the generation of the BoW per video clip. Table 3.1 provide the details of the datasets used in the experiments.

Table 3.1: Details of the datasets used

Dataset	Classes	Train Videos	Test Videos	Avg.Length (Desc.)
KTH	6	383	216	4 sec. (849)
HMDB51	51	3,567	1,530	5 sec. (1456)
UCF101	101	9,535	3,782	7.21 sec. (1574)

The sequential BoW generation (Seq-BoW) and tree-based BoW generation (Tree-BoW) approaches are used as the baseline for both the effectiveness (classification accuracy) and the computational time. To measure the effectiveness of the generated BoW features, we use the linear SVM with default hyper-parameters for all the existing and proposed approaches because the objective of this research work is to scale the process of BoW generation while preserving the effectiveness of the generated features instead of improving the state-of-the-art on these datasets. However, one can use a scalable kernel SVMs [106, 104] with a suitable kernel to further improve the classification accuracy. Also, we keep the train and test split as described in the respective dataset. In case of multiple train-test splits, the performance is the averaged for the splits. The performance of both the existing and the proposed approach are compared for various sizes of the vocabularies (i.e. $k = \{128, 1024, 8192, 65536\}$). Fig. 3.2 illustrates the challenge of the rapid increase

in the computation time for the sequential BoW with an increase in the vocabulary size on the *KTH* action dataset. It shows that large vocabulary based BoW increases the classification performance but also increases the time taken in the BoW generation rapidly and after certain vocabulary size, it becomes impractical to be used in real-time. For example for the vocabulary sizes $k = 8192$ and $k = 65536$, the time taken by Seq-BoW is 3.295 and 31.076 seconds, respectively for the video of 4 second duration.

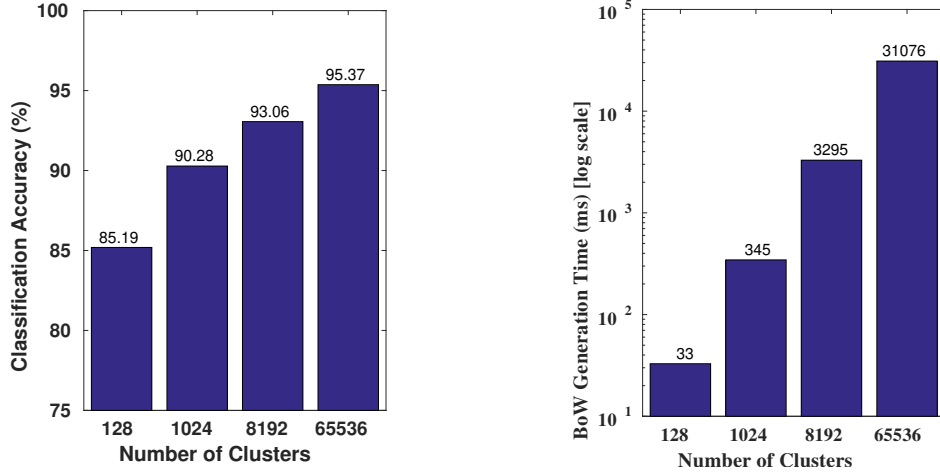


Figure 3.2: Effect of the vocabulary size on classification accuracy and BoW generation time.

Table 3.2: Comparison of the classification performance (%) of the proposed Fast-BoW approach with the existing Seq-BoW and Tree-BoW approaches

Dataset+Method	Number of Clusters (k)			
	128	1024	8192	65536
KTH+Seq-BoW	85.19	90.28	93.06	95.37
KTH+Tree-BoW	79.17	88.89	86.11	90.74
KTH+Fast-BoW	82.41	89.81	89.96	90.74
HMDB51+Seq-BoW	08.17	13.99	15.16	15.37
HMDB51+Tree-BoW	06.54	11.11	13.40	11.50
HMDB51+Fast-BoW	07.19	13.59	13.79	14.38
UCF101+Seq-BoW	16.13	30.88	36.20	39.66
UCF101+Tree-BoW	12.14	27.10	32.84	32.18
UCF101+Fast-BoW	14.99	27.47	35.69	37.82

Table 3.2 shows the performance of the classification for existing and proposed methods on various sizes of the vocabularies. The results demonstrate that the proposed approach *Fast-BoW* is significantly reduced the loss in the effectiveness of the generated BoW in comparison to the hierarchical clustering based tree approach. The results in Table 3.3

illustrate that the *Fast-BoW* outperform all the existing approaches with $\approx 4000\times$ and $\approx 12\times$, respectively.

Table 3.3: Comparison of time (milliseconds) taken in BoW generation per test video.

Dataset+Method	Number of Clusters (k)			
	128	1024	8192	65536
KTH+SeqBoW	33	345	3295	31076
KTH+TreeBoW	3	4	5	7
KTH+FastBoW	0.24	0.32	0.39	0.57
HMDB51+SeqBoW	40	585	5152	35805
HMDB51+TreeBoW	5	6	9	13
HMDB51+FastBoW	0.37	0.49	0.71	1
UCF101+SeqBoW	129	219	9265	18470
UCF101+TreeBoW	14	15	20	26
UCF101+FastBoW	1	1	2	2

Values greater than 1 ms are rounded to nearest integers.

Figure 3.3 gives the comparison of the loss in the classification performance of the existing Tree-Bow and the proposed Fast-BoW with respect to the sequential approach Seq-BoW. The figure clearly shows that the loss in the proposed Fast-BoW is significantly lower than the loss in the Tree-BoW for various vocabulary size across all three datasets. Thus, the proposed Fast-BoW preserves the effectiveness of the BoW features in comparison to the existing Tree-BoW approach.

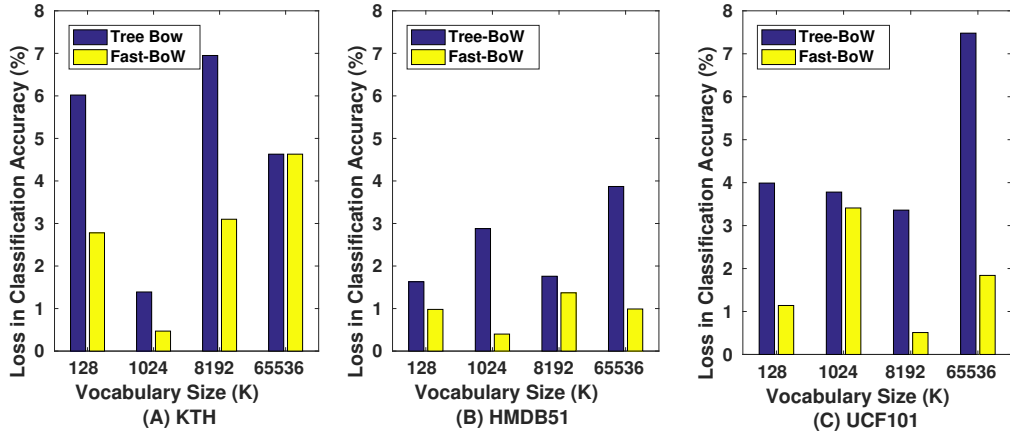


Figure 3.3: Comparison of the classification loss in existing tree based approach and the proposed Fast-BoW approach with respect to sequential BoW.

Figure 3.4 illustrate the speed-up gain by the proposed Fast-BoW with respect to the existing Seq-BoW and Tree-BoW approaches. It can be observed from the figure that the

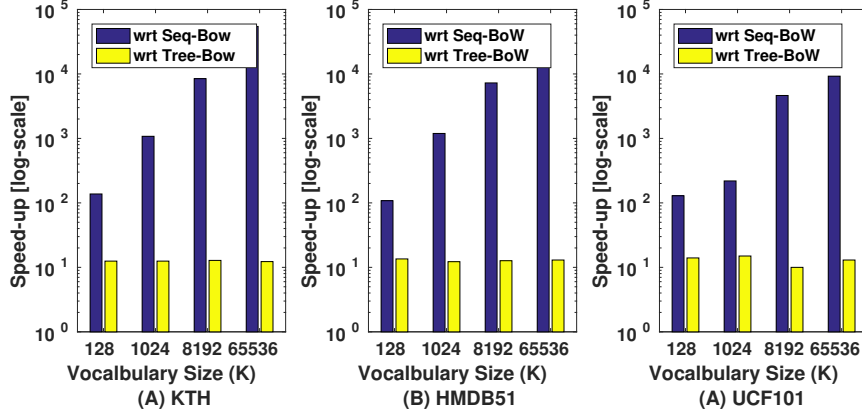


Figure 3.4: Scaling factor of Fast-BoW with respect to sequential BoW approach and the existing tree based approach.

proposed Fast-BoW gains several orders of speed-up over the existing approaches. For sequential approach, it is $\approx 100\times$ faster for the vocabulary size $k = 128$ and this scale increase with the increase in the vocabulary size. Also, it achieves $\approx 12\times$ speed up over the Tree-BoW approach when keeping the $|\mathbf{h}| = \sqrt{d}$, where $d = 162$ is the dimensions of the STIP descriptors. As the reducing the $|\mathbf{h}|$ (for example $|\mathbf{h}| = \log_2 d$) results into further speed-up but increases the loss in the classification performance. Also, the proposed approach takes ≤ 2 milliseconds only for each video of duration 4 – 7.21 seconds. Thus making the BoW approach practical to be used in real-time with increased vocabulary sizes and increased effectiveness.

3.3 Summary

The proposed *Fast-BoW* scales the computational complexity of hard and soft bag-of-words generation to $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$ from $O(d \log_2 k)$ and $O(d k)$, respectively. While it preserves better the effectiveness of the generated features than the existing hierarchical clustering tree-based approach. The use of softmax for predicting the cluster probabilities for a local feature vector not only improves the clustering performance but also provides the flexibility to further scaling by applying quantization and hashing. The experimental results show that the choosing $|\mathbf{h}| = \sqrt{d}$ gives almost no loss to the final classification accuracy while choosing $|\mathbf{h}| = \log_2 d$ results into a mild loss. Thus the proposed *Fast-BoW* shows the efficacy to be used in real-time applications with large vocabularies.

Chapter 4

Graph representation for abnormal activity recognition

Abnormal activity recognition is a challenging task in surveillance videos. In order to detect abnormal activities in surveillance videos or crowd behavior analysis, various kinds of activity modeling are proposed in the literature [9, 13, 50, 63, 86, 67, 68]. The widely used bag-of-words (BOW) approaches [20, 121, 125] show excellent performance in action and activity recognition. A bag-of-words (BOW) approach computes an unordered histogram of visual words occurrences that encodes only the global distribution of low level descriptors, while it ignores the local structural organization (i.e. geometry) of the salient points and corresponding low level descriptors. However, use of such local structure of salient points and corresponding low level descriptors may lead to discriminative video representation which further leads to better recognition of video activities. In this chapter, we propose an approach for abnormal activity recognition based on graph formulation of video activities and graph kernel support vector machine.

The interactions of entities in a video are formulated as a graph of local actions which includes appearance and motion information along with geometric relationships among various interactions of the entities in a video activity. The vertices of the graph are spatio-temporal interest points and an edge represents the relation between appearance and dynamics around the interest points. The edges of the graph are determined using a fuzzy membership function on the basis of closeness and the similarity of the entities associated with the interest points. If two points are close to each other, then there is a high probability that some interactions take place between the corresponding entities. In order to keep track of the objects, we also incorporate the appearance and motion of the entities using histogram of oriented gradients (HOG) and histogram of oriented optical flow (HOF). Once the activity is represented using a graph, then for classification of the activities into normal or abnormal classes, we use a support vector machine with graph kernel. These graph ker-

nels provide robustness to slight topological deformations in comparing two graphs, which may occur due to the presence of noise in data. We demonstrate the efficacy of the proposed method on the publicly available standard datasets viz; UCSDped1, UCSDped2 and UMN. The experiments demonstrate the superiority of the proposed work over the existing methods which are based on dense trajectories and bag-of-words with various feature descriptors.

The rest of the chapter is organized as follows: Section 4.1 describes the proposed approach for abnormal activity recognition. Section 4.2 discusses the experimental setup, datasets and results. The summary of the chapter is provided in section 4.3.

4.1 Graph formulation of activities in a video

The proposed framework for abnormal activity recognition in surveillance videos is presented in this section. The block diagram of the proposed framework is shown in Fig. 4.1. The proposed framework consists of three steps. In the first step, the incoming video feed is split into video clips of size T and space-time interest points in each video clip are extracted. In the second step, a set of undirected graphs of local activities is generated. The vertices of the graphs are space-time interest points and an edge represents a possible interaction. In the third step, each activity is classified into normal or abnormal categories. Which is further classified into local abnormal activity recognition and global abnormal activity recognition. For local activity classification a max-margin classifier is trained using graph kernel support vector machine from training videos. For global activity recognition, bag-of-graphs (BoG) feature vectors are generated for a set of local activity graphs and a support vector machine model trained from BoG feature vectors from training videos is used to declare the global behavior. Each of these steps are discussed in detail below:

4.1.1 Detection of space-time interest points

The space-time interest points [59] are salient points, which are the regions in $f : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$ having significant eigenvalues λ_1, λ_2 , and λ_3 of a spatio-temporal second-moment matrix μ , which is a 3-by-3 matrix composed of first order spatial and temporal derivatives averaged using a Gaussian weighting function $g(\cdot; \sigma_i^2, \tau_i^2)$ with integration scales σ_i^2 (spatial variance) and τ_i^2 (temporal variance). The value of μ is computed as

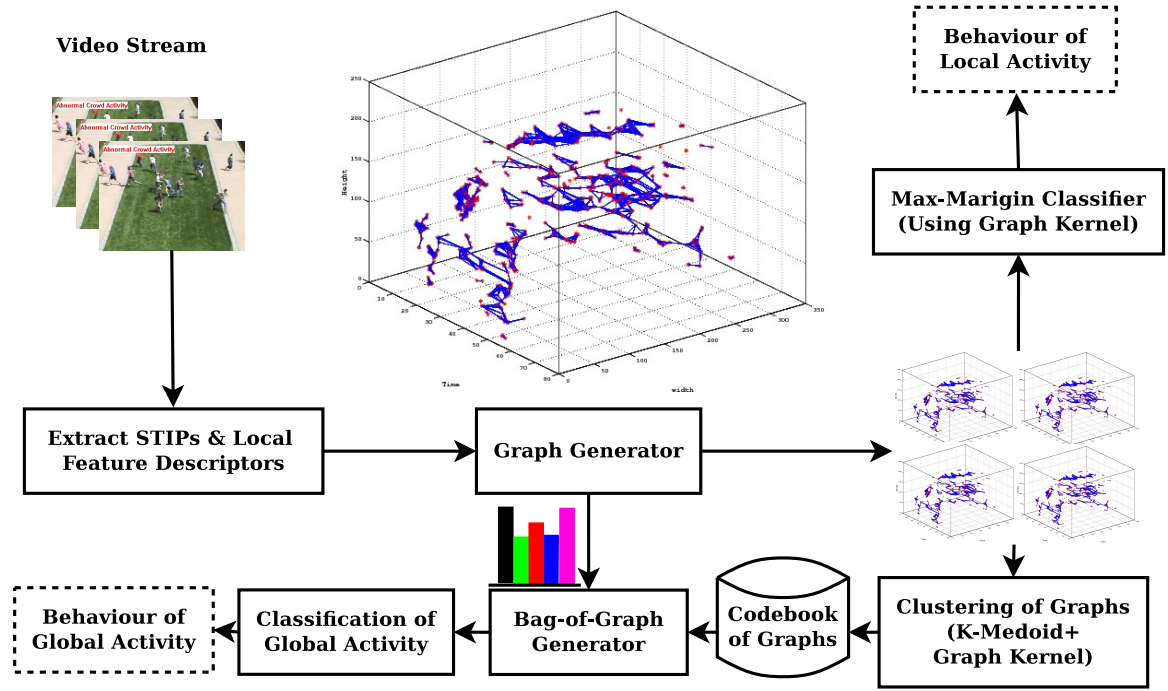


Figure 4.1: Block diagram of the proposed framework for abnormal activity recognition in surveillance videos.

$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix}, \quad (4.1)$$

where L_x , L_y , and L_t are first-order derivatives with respect to x , y , and t of the linear scale-space representation $L : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}_+^2 \rightarrow \mathbb{R}$ of f constructed by convolution of f with an anisotropic Gaussian kernel $g(\cdot; \sigma_i^2, \tau_i^2)$ with local scales σ_i^2 (spatial variance) and τ_i^2 (temporal variance). The value of the L is computed as

$$L(\cdot; \sigma_i^2, \tau_i^2) = g(\cdot; \sigma_i^2, \tau_i^2) * f(\cdot). \quad (4.2)$$

These interest points are detected using Harris3D corner function (H) for the spatio-temporal domain by combining the determinant (det) and the trace of μ ($trace$) as follows:

$$H = det(\mu) - k trace^3(\mu)$$

$$H = \lambda_1 \lambda_2 \lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3, \quad (4.3)$$

where k is a constant. Then around each salient point $p(w, h, t)$, 72-dimensional HOG [26] and 90-dimensional HOF [17] descriptors are extracted, which together represent an in-

terest point in the 3D space by a 162-dimensional feature vector $\mathbf{f} = \mathbb{R}^{162}$ called STIP descriptor. In this way, the STIP feature descriptors include the appearance information using HoG and motion information using HoF around the salient points. Next we present the process of graph generation.

4.1.2 Graph formulation of a video

In previous step, we obtain a set of space-time interest points $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ where $\mathbf{p}_i \in (x, y, t)$ represents a 3D point and their respective feature vectors $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^n$ for a given video. In this step, we represent the video as a graph $G(\mathbf{P}, \mathbf{E})$, where \mathbf{P} is the set of space-time interest points detected from previous step and \mathbf{E} is the set of edge. An edge between two points \mathbf{p}_i and \mathbf{p}_j is decided based on μ_{ij} , which is the edge existence score and is computed as

$$\mu_{ij} = \frac{K(\mathbf{f}_i, \mathbf{f}_j)}{\|\mathbf{p}_i - \mathbf{p}_j\|_2}, \quad (4.4)$$

where, $K(\mathbf{f}_i, \mathbf{f}_j)$ is the similarity measure between the feature vectors \mathbf{f}_i and \mathbf{f}_j extracted at points \mathbf{p}_i and \mathbf{p}_j , respectively. Any geometric kernel function can be used as a similarity measure like linear kernel, polynomial kernel, RBF kernel, or sigmoid kernel. This similarity is high if feature vectors \mathbf{f}_i and \mathbf{f}_j are belonging to similar events and/or similar object. This shows that these points are either too close to each other and share a significant information during feature extraction or object at point \mathbf{p}_i moved to point \mathbf{p}_j over the time. The later case is significant for modeling an activity, so geometric distance $\|\mathbf{p}_i - \mathbf{p}_j\|_2$ between points \mathbf{p}_i and \mathbf{p}_j is in the denominator. In this way, a high value of μ_{ij} shows significance towards the existence of an event between these points. The adjacency matrix \mathbf{A} of the graph G is computed as

$$A_{ij} = \begin{cases} 0, & \text{if } \mu_{ij} < \mu_T \text{ Threshold} \\ 1, & \text{otherwise} \end{cases} \quad (4.5)$$

4.1.3 Activity recognition

Once the video activities are represented using graphs, then the next task is to classify them as normal activity or abnormal activity. This section presents framework for detecting both local and global activities.

Recognition of local abnormal activities

A surveillance video may contain multiple local activities occurring simultaneously. Each local activity can be represented using a graph. A one-class SVM classifier is trained from the graphs of local activity from the training videos. Then this classifier is used to predict the behaviour of the local activities in the test videos. The anomaly score γ for a given graph of local activity G is computed from one-class SVM as below:

$$\gamma = f(G) = \sum_{i=1}^m \alpha_i K(G_i, G) - \rho, \quad (4.6)$$

where, $\{G_i, \dots, G_m\}$ are the m support graphs with their respective Lagrange multipliers α_i , ρ is the threshold value. If the weighted density of the test graph with support graphs is above a threshold ρ then the activity associate with the graph is classified as normal and abnormal otherwise. The values of these parameters are computed by solving below dual problem for n training graphs $\{G_i, \dots, G_n\}$:

$$\max_{\bar{\alpha}} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(G_i, G_j) \text{ s.t. } \sum_{i=1}^n \alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq \frac{1}{vn}, \quad (4.7)$$

where $v \in (0, 1)$ control the penalty imposed on the nonzero slack variables, C is the box constraint parameter, and $K(G_i, G_j)$ is a graph kernel function used for computation of the similarity between two graphs. A wide range of graph kernels are proposed in the literature like shortest path kernel and random walk kernel. We used random walk kernel because it is computationally efficient than other graph kernels. The random walk kernel [31] compares two graphs by counting number of common random walks between them. The number of common random walks of length k are calculated by taking direct product graphs because random walk on direct product graph is equivalent to simultaneous random walk in the two graphs [31]. The k^{th} power of adjacency matrix of the resultant graph after direct product gives the number of common walks. The direct product graph of two graphs is defined as given below:

Definition 4.1 (Product Graph). *Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are two graphs, then $G_{\times}(V_{\times}, E_{\times})$ is the direct product graph where the node and edge set of the direct product graph are defined as*

$$V_{\times} = (v_1^i, v_2^r) : v_1^i \in V, v_2^r \in V'$$

$$E_{\times} = ((v_i, v_r'), (v_j, v_s')) : (v_i, v_j) \in E \wedge (v_r', v_s') \in E'$$

Using the definition of direct product graph, Gartner *et al.* in [31] defined random walk

kernel as follows:

Definition 4.2 (Random Walk Kernel). *Let G_1 and G_2 be two graphs. Then for product graph G_\times , let V_\times be the node set of G_\times and \mathbf{A}_\times be the adjacency matrix for the graph product. With start probability \mathbf{p}_\times , end probability \mathbf{q}_\times , and a sequence of weights (decaying factor) $\lambda = \lambda_1, \lambda_2, \dots, (\lambda_i \in \mathbb{R}, \lambda_i \geq 0 \forall i \in \mathbb{N})$, the random walk kernel is defined as*

$$K(G_1, G_2) = \sum_{k=1}^{\infty} \lambda_k \mathbf{q}_\times^T \tilde{\mathbf{A}}_\times^k \mathbf{p}_\times, \quad (4.8)$$

where $\tilde{\mathbf{A}} = \mathbf{A}^T [\mathbf{I} \cdot (\mathbf{A}^T \cdot \mathbf{e})]^{-1}$ is the normalized matrix.

The kernel in Equation (4.8) is a valid positive semi-definite (p.s.d.) kernel. This can be proved with the help of following technical lemma:

Lemma 4.3. $\forall k \in \mathbb{N} : \tilde{\mathbf{A}}_\times^k \mathbf{p}_\times = \text{vec}[(\tilde{A}_2^k \mathbf{p}')(\tilde{A}_1^k \mathbf{p})^T]$.

Lemma 4.4. *If $X \in \chi^{n \times m}$, $Y \in \mathbb{R}^{m \times p}$, and $Z \in \chi^{p \times q}$, then*

$$\text{vec}[\tilde{X}Y\tilde{Z}] = [\tilde{Z}^T \otimes \tilde{X}] \text{vec}(Y) \in \mathbb{R}^{nq \times 1}$$

where \otimes represents Kronecker product and vec represent the vectorization. The proofs of Lemma 4.3 and Lemma 4.4 can be found in [118]. Using Lemma 4.3 and Lemma 4.4, we can write

$$\begin{aligned} \mathbf{q}_\times^T \tilde{\mathbf{A}}_\times^k \mathbf{p}_\times &= \mathbf{q}_\times^T \text{vec}[(\tilde{A}_2^k \mathbf{p}_2)(\tilde{A}_1^k \mathbf{p}_1)^T] && \text{Using Lemma 4.3} \\ &= (\mathbf{q}_1 \otimes \mathbf{q}_2)^T \text{vec}[(\tilde{A}_2^k \mathbf{p}_2)(\tilde{A}_1^k \mathbf{p}_1)^T] && \text{Because } \mathbf{q}_\times = \mathbf{q}_1 \otimes \mathbf{q}_2 \\ &= \text{vec}[\mathbf{q}_2^T \tilde{A}_2^k \mathbf{p}_2 (\tilde{A}_1^k \mathbf{p}_1)^T \mathbf{q}_1] && \text{Using Lemma 4.4} \\ &= (\mathbf{q}_1^T \tilde{A}_1^k \mathbf{p}_1)^T (\mathbf{q}_2^T \tilde{A}_2^k \mathbf{p}_2) \end{aligned} \quad (4.9)$$

Each individual term of Equation (4.9) equals $\Phi^k(G_1)^T \Phi^k(G_2)$ for some function Φ , and is therefore a valid positive semi-definite kernel. The time complexity of computation of Equation (4.8) is $O(n^6)$. A fast random walk kernel is proposed by Vishwanathan *et al.* in [117] which reduces the time complexity to $O(n^3)$ with the help of Sylvester equation and conjugate gradient (CG) methods to solve the system of equations,

$$K(G, G') = \mathbf{q}_\times^T (\mathbf{I} - \lambda \mathbf{A}_\times)^{-1} \mathbf{p}_\times. \quad (4.10)$$

Recognition of global abnormal activity

The global activities are the set of multiple local activities. The local activities in a global abnormal activity need not be abnormal. The co-occurrence of several normal local activity can lead to an abnormal behaviour. After formulating all the local activities as graphs representing geometric relations of interactions of entities, we build a high level vocabulary $\mathbf{V} = \{G_j\}_{j=1}^n$ of graphs using k -median clustering over the set of all graphs $\mathbf{G} = \{G_i\}_{i=1}^n$ by solving the objective function:

$$\arg, \min_{G_j \in \mathbf{V}} \sum_{G_i \in \mathbf{G}} K^{-1}(G_i, G_j). \quad (4.11)$$

Then vocabulary \mathbf{V} of graphs of local activities is used to generate, a high level bag-of-graphs (BoG) representation for global activities. After this, a one-class SVM with a standard vector kernel is used to classify the global activities into normal or abnormal categories.

4.2 Experiments and results

The experimental setup, benchmark datasets, and the outcomes of the experiments are discussed in this section. The value of the λ in Equation (4.10) is set to $1/d^2$, where d is the maximum degree in the entire graph dataset. The value of box constraint C in SVM is set to 1. Three datasets, namely, UCSDped1 [73], UCSDped2 [73], and UMN are used to validate the proposed approach. Fig 4.2 shows samples of one normal and one abnormal activities from each of the three datasets and their corresponding graph formulation. We compare the proposed approach with other existing state-of-the-art methods like bag-of-visual-words (BoW) using STIP/SIFT and dense trajectory based approaches.

The UCSD anomaly detection dataset is a well-known publically available dataset for video anomaly detection. The videos of pedestrian are captured using a camera mounted at an elevation where the abnormal events include the circulation of non-pedestrian entities in the walkways or anomalous pedestrian motion patterns such as fast motion, zig-zag motion, etc. The crowd density in the walkways ranges from sparse to very crowded. This dataset have two subsets, namely, UCSDped1 and UCSDped2. UCSDped1 dataset contains videos captured in vertical view i.e. groups of people walking towards and away from the camera, and there is some amount of perspective distortion. It contains 34 training and 36 testing videos. The performance of classification on UCSDped1 dataset using the proposed approach is 97.14%, where as the performance of existing bag-of-words approach using STIP

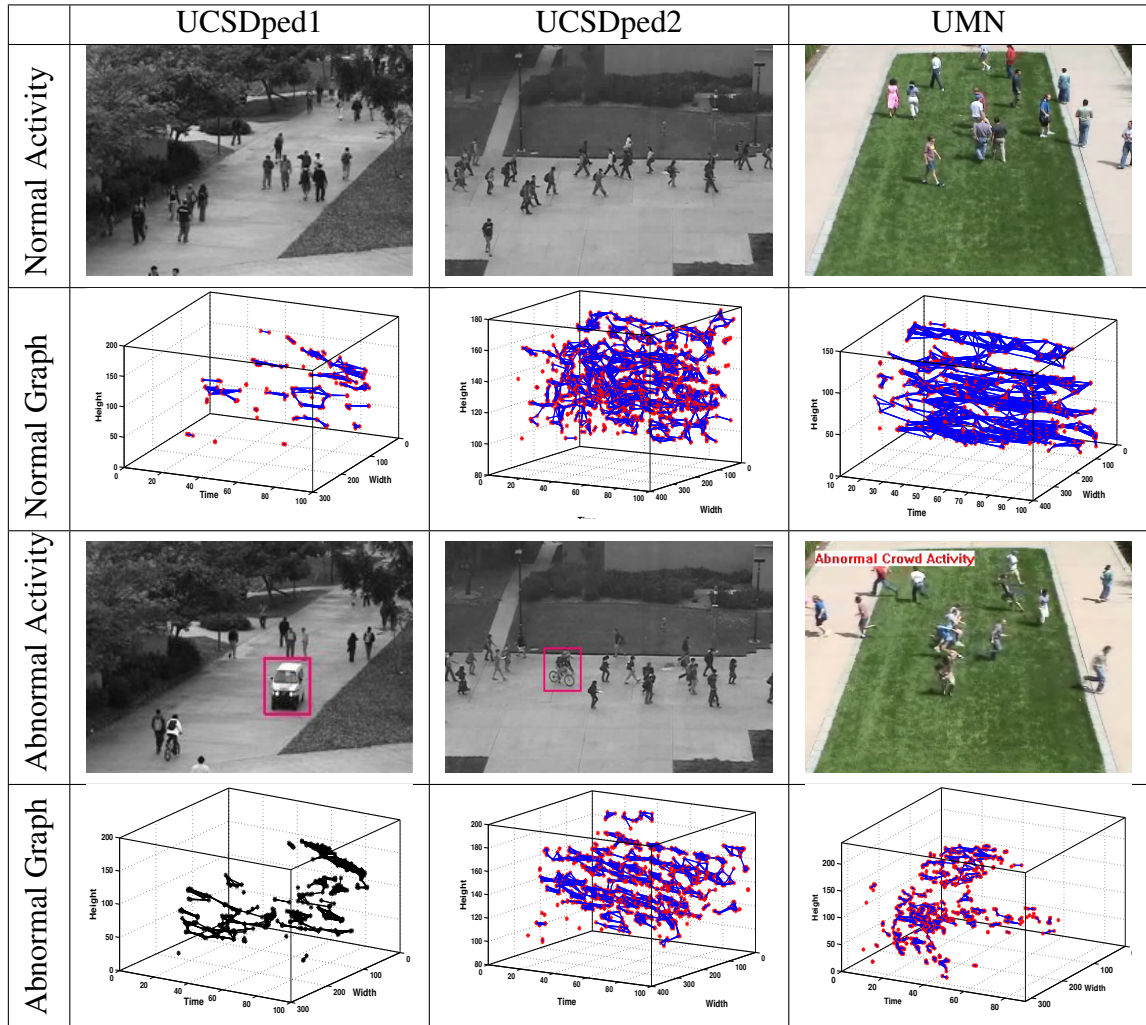


Figure 4.2: Illustration of normal and abnormal sample and corresponding graphs from all datasets.

features is 82.00% on the same dataset. The performance of existing other bag-of-words approaches using SIFT and dense trajectories give 80.00% and 85.71%, respectively, on the same dataset. UCSDped2 dataset contains the scenes of pedestrian movement parallel to the camera plane. It contains 16 training video samples and 12 testing video samples. The performance of classification on UCSDped2 dataset using the proposed approach is 90.13%, where as the performance of existing bag-of-words approach using STIP features is 75.82% on the same dataset. The performance of existing other bag-of-words approaches using SIFT and dense trajectories give 77.62% and 88.86%, respectively, on the same dataset.

The proposed approach is able to extract significant evidence with discriminative ability in order to detect abnormal activities efficiently because of incorporation of geometric structure along with motion and appearance information. There is a significant improve-

ment in the performance of the proposed approach on both the datasets due to the deviation in the geometrical structure of the graphs generated during normal walking and the graphs corresponds to the fast motion, zig-zag motion, and appearance of vehicles as can be shown in fig 4.2.

UMN dataset is also a publically available benchmark dataset from the University of Minnesota for video anomaly detection containing normal and abnormal crowd videos. The initial part of each video consist normal behavior while ends with sequences of the abnormal behavior. The dataset contains 11 training and 11 testing video scenes in different environments where a crowd of people walking normally and after some time, they suddenly start running. Fig. 4.2 shows that the geometrical structure of the graphs generated during normal walking (dense and bigger graph) are deviating from the graphs generated during running (sparse and small graph). In this way, the evidences obtained using the proposed approach contains significant information in order to detect abnormal activities efficiently. The performance of classification on UMN dataset using the proposed approach is 95.24%, where as the performance of existing bag-of-words approach using STIP features is 85.00% on the same dataset. The performance of existing other bag-of-words using SIFT and dense trajectories give 85.00% and 81.00%, respectively, on the same dataset.

It is observed that the proposed approach achieves better performance when compared to other bag-of-words approached using various descriptors like STIP (HoG+HoF), SIFT, and dense trajectories on UCSDped1, UCSDped2, and UMN datasets. Table 4.1 gives the performance comparison of the proposed approach with existing methods.

Table 4.1: Comparison of classification performance (%) of proposed approach with existing bag-of-words (BoW) approaches using STIP, SIFT and dense-trajectories (DT)

Dataset	SIFT+BoW	STIP+BoW	DT+BoW	Proposed
UCSDped1	80.00	82.00	85.71	97.14
UCSDped2	77.62	75.82	88.86	90.13
UMN	85.00	85.00	81.00	95.24

Table 4.2 presents the performance comparison of proposed approach with the existing state-of-the art methods. It can be observed from the Table 4.2 that the proposed method achieves consistent performance on all the three datasets used. Also, the proposed approach on UCSDped1 and UCSDped2 datasets outperforms the state-of-the-art methods and achieves a comparable performance on UMN datasets. This may be due to the fact that the performance of abnormal activity recognition depends on the nature/type of anomaly present in the dataset. Overall, the proposed method achieves better performance across datasets as it is able to detect a wide variety of abnormal activities in videos.

Table 4.2: Performance comparison (%) of proposed approach with existing methods

Reference	Method	UCSDped1	UCSDped2	UMN
Adam <i>et al.</i> 2008 [4]	Adam	61.10	54.20	-
Mehran <i>et al.</i> 2009 [75]	SF	63.50	65.00	87.40
Kim <i>et al.</i> 2009 [56]	MPPCA	64.40	64.20	-
Mahadevan <i>et al.</i> 2010 [73]	MDT	75.00	75.00	96.30
Wu <i>et al.</i> 2010 [127]	Chaotic Invar.	-	-	94.70
Cong <i>et al.</i> 2011 [23]	Sparse	81.00	-	97.20
Raghavendra <i>et al.</i> 2011 [87]	PSO	79.00	-	-
Antic <i>et al.</i> 2011 [7]	BVP	82.00	-	-
Saligrama <i>et al.</i> 2012 [98]	LSA	84.00	-	96.60
Roshtkhari <i>et al.</i> 2013 [93]	Roshtkhari	85.00	-	-
Lu <i>et al.</i> 2013 [71]	150fps	85.00	-	-
Li <i>et al.</i> 2014 [65]	H-MDT	82.20	81.50	96.30
Kaltsa <i>et al.</i> 2015 [52]	Swarm	72.98	73.08	97.01
Chang <i>et al.</i> 2015 [20]	GPR	76.30	-	-
Proposed	Graph	97.14	90.13	95.24

4.3 Summary

In this chapter, we present a novel framework for abnormal activity recognition in surveillance videos. The graph formulation of activities captured in surveillance videos contain significant discriminative ability to determine the behaviour of activities. The motion of the objects/entities, their co-relation, and interactions to each others is subsequently represented by graphs. Finally, the graph formulation of the video activities convert the problem of anomaly detection into a graph classification problem for this, we exploit support vector machine together with graph kernel. The use of graph kernel for measuring similarity between two graphs provides robustness to slight deformations to the topological structures due to presence of noise in data. The experimental results outperform the existing widely used methods like dense trajectories, bag-of-visual-words etc., which proves the efficacy of the proposed approach.

Chapter 5

Distributed QP solver for kernel SVM using genetic algorithm

Support vector machine (SVM) is a powerful tool for classification and regression problems due to its generalization capabilities. However, the time and space complexities of the existing SVM solvers make it unsuitable for large datasets. In this chapter, we present Genetic-SVM, an evolutionary computing based distributed approach to find optimal solution of quadratic programming (QP) for kernel support vector machine. In Genetic-SVM, novel encoding method and crossover operation help in obtaining the better solution. In order to train a SVM from large datasets, we distribute the training task over the graphics processing units (GPUs) enabled cluster. It leverages the benefit of the GPUs for large matrix multiplication. The experiments show better performance in terms of classification accuracy as well as computational time on standard datasets like GISETTE, ADULT, etc.

The core of SVM is in solving a quadratic programming (QP), a computationally complex problem which separates support vectors from the rest of the training data. The time complexity for a standard SVM training is $O(n^3)$ and the space complexity is $O(n^2)$, where n is the size of training dataset [114]. It is thus computationally infeasible on very large data sets. Sequential minimal optimization (SMO) is the state of the art QP solver which is used in LIBSVM, an implementation of SVM. But this method is sequential, so we can not leverage the benefit of high performance distributed computing environments like high performance cluster (HPC), cloud cluster, GPU cluster etc. Stochastic gradients decent (SGD) method can be distributed in order to train on large scale datasets. But this method works only for linear kernels. There is no existing true parallel or distributed algorithm to solve the constrained quadratic programming problem used to separate the support vectors from the training data for kernel SVM. In order to improve the training speed of SVM, many approaches have been proposed in the literature. These approaches can be categorized into *decomposition based approaches* and *partitioning based approaches*. The decomposition

based approaches efficiently address the space complexity, however, time complexity remains a challenge. The partitioning based parallel and distributed SVM methods partition the data into smaller partitions and train SVM over them independently and later combine them to produce final support vectors. But, the partitioning based distributed SVM approaches [111, 6, 40, 132] are prone to loss of accuracy and high communication overhead.

In [38], Herrero-Lopez *et al.* accelerate SVM training by integrating graphics processing unit (GPU) into MapReduce clusters. It distributes the matrix multiplication tasks during the sequential update of the Lagrangian multipliers, however, it does not allow the desired level of acceleration due to the sequential nature of the SVM. The evolutionary computing shows success in order to find a solution near to the optimal solution quickly for NP hard problems and the computations are easy to perform independently in a distributed environment. Also, the execution of genetic algorithms can be accelerated by utilizing the massive parallelization power of the GPU cluster for training over large datasets. GPU-based parallel genetic algorithm are also proposed by [80, 83, 119, 123] for various applications. Several researchers also use genetic algorithm for parameter tuning, i. e. selecting the best performing parameters for SVM training [127].

However, in this work, we aim to exploit the evolutionary computing based optimization ability of the genetic algorithm to perform distributed computation in finding the optimal solution for the SVM i. e. support vectors and their respective α coefficients. Merz *et al.* [76] use genetic algorithm for binary quadratic programming (BQP) problem, but this is not applicable for the real valued QP problem in SVM. Herrera *et al.* [37] implement genetic algorithm based support vector regression. It represents the real numbers into binary strings and apply traditional genetic algorithm. Also, it does not explore the automatic tuning of the various parameters used in kernel SVM like regularization parameter C , what is considered an open research area. In [102], Silva *et al.* implement least square SVM (LS-SVM) using genetic algorithm. The disadvantages of these methods are: 1) Sparsity is not incorporated, due to which all vectors in the training dataset become support vectors (SVs), and 2) Generation of large number of invalid solutions reduce the computational efficiency. Apart from these limitations, all the above discussed methods use sequential computation only. We propose an evolutionary computing based quadratic programming (QP) solver for distributed training of kernel SVM known as Genetic-SVM.

The rest of the chapter is organized as follows: The proposed Genetic-SVM is discussed in section 5.1. Section 5.3 describes the experimental setup, evaluation method and results. We summarize in section 5.4.

5.1 Operations of Genetic-SVM

This section presents the proposed Genetic-SVM for the optimization of quadratic programming (QP) problem for support vector machine (SVM).

Let $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ be the dataset with n feature vectors, $\mathbf{x}_i \in \mathbb{R}^d$ be the d dimensional feature vector and $y_i \in \{-1, +1\}$ be the class label. Then the QP problem for SVM is to maximize:

$$J(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{e} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}, \quad (5.1)$$

where $q_{ij} = y_i y_j K(\mathbf{x}_i^T, \mathbf{x})$ and $\alpha_i \in \boldsymbol{\alpha}$ are the Lagrangian multipliers. A valid solution must satisfy following constraints:

$$\boldsymbol{\alpha}^T \mathbf{y} = 0, \quad (5.2)$$

$$0 \leq \alpha_i \leq C, \quad \forall \alpha_i \in \boldsymbol{\alpha}. \quad (5.3)$$

Here, C is a regularization parameter. Solving Equation (5.1) gives $\boldsymbol{\alpha}$ and the value of bias b . All non-zero $\alpha_i \in \boldsymbol{\alpha}$ are called support vectors. Let m be the number of support vectors. Then the decision of a vector \mathbf{x} is predicted using support vectors and their corresponding α_i values using the following decision function:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^m \alpha_i y_i K(\mathbf{x}_i^T, \mathbf{x}) + b \right). \quad (5.4)$$

As discussed earlier, existing sequential minimal optimization (SMO) solves Equation (5.1) sequentially and also result in a solution that is not necessarily optimal. In the subsequent section, we propose a solver for Equation (5.1) using genetic algorithm in order to obtain the better solution. Also, we propose a distributed framework which performs distributed computation over GPU enabled cluster in order to reduce the time for SVM training.

Here, we propose a genetic algorithm based solver for QP in Equation (5.1). The solution for Equation (5.1) is the optimal set of Lagrangian multipliers $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^n, \alpha_i \in \mathbb{R}$. As shown in Fig. 5.1, it generates random solutions i.e. $\boldsymbol{\alpha}_j$ and represents each solution using its n values of α_i , called random key representation. For evaluating the fitness of each solution, we use objective function in Equation (5.1) as fitness function. Reproduction operations are performed directly on the random keys of two candidate solution in order to generate new solutions. The details of the operations performed in proposed genetic algorithm based QP solver for searching the best solution is given here. As shown in Fig. 5.1-(A), the steps in a genetic algorithm include:

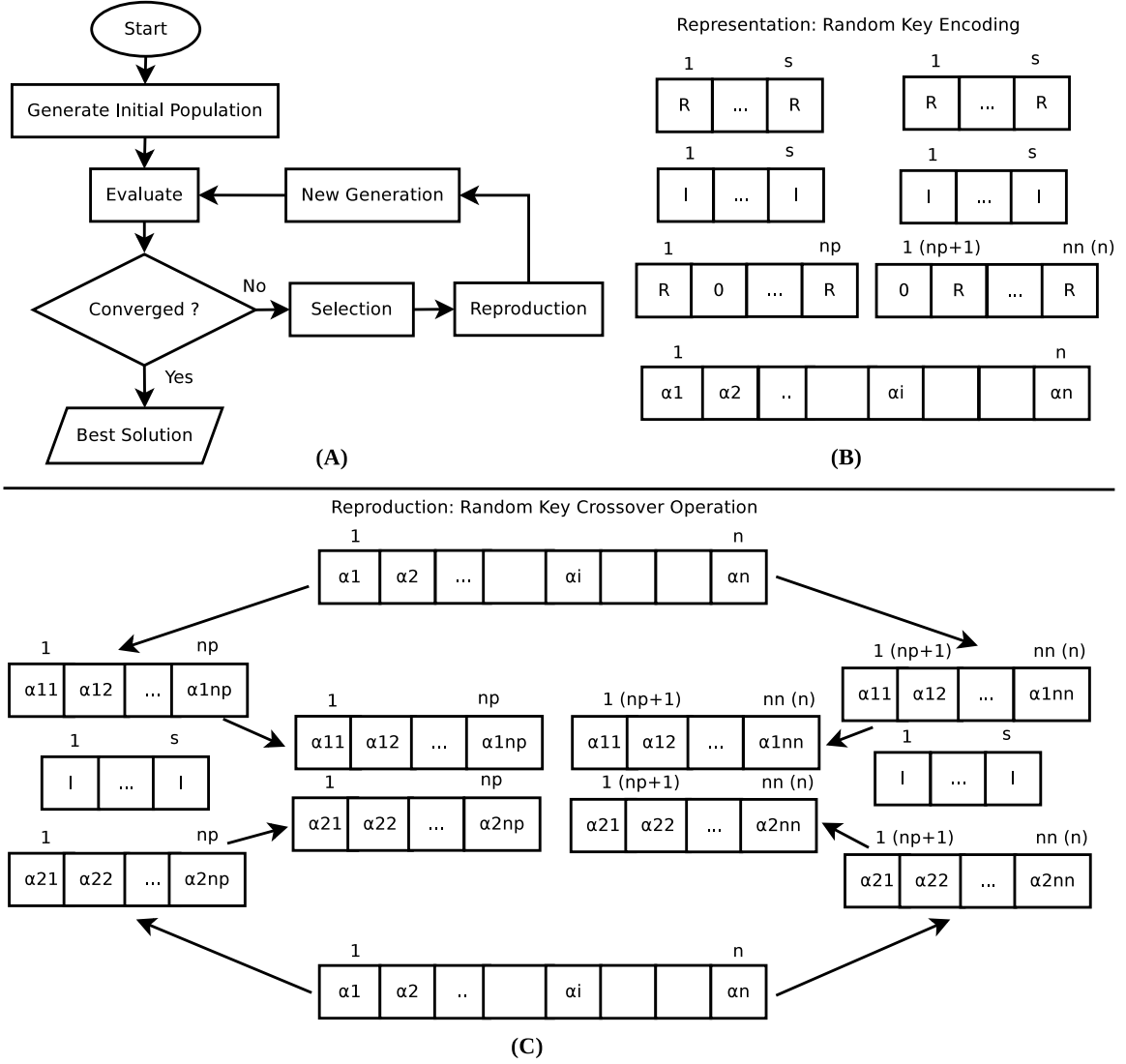


Figure 5.1: Genetic-SVM operations. (A) The flow diagram of the steps in genetic algorithm. (B) The process of the solutions representation using random key encoding. (C) The process of crossover operation for reproduction of new candidate solutions.

5.1.1 Random key encoding

The proposed approach uses random key encoding in order to represent the candidate solutions. The candidate solutions are the positive real valued $\alpha \in \mathbb{R}^n$, where n is the number of vectors in the training set. The encoding should satisfy the constraints given in Equation (5.2) & (5.3). The Algorithm 5.1 generates $\alpha \in \mathbb{R}^n$ which satisfies both the constraints. Let n_p be the number of positive class vectors and n_n be the negative class vectors. As shown in Fig. 5.1-(B), it generates two random vectors α_p and α_n of size n_p and n_n with sparsity s_p and s_n , respectively. Hence, output vectors α_p and α_n have only s_p and

s_n non zero values, respectively. In order to satisfy constraints given in Equation (5.2) & (5.3), the vector α_p and α_n are normalized with factor f_p and f_n , respectively, as follows:

$$\alpha^p \leftarrow \alpha^p \times f_p; \text{ where } f_p \leftarrow \frac{n \times C}{4 \times \mathbf{e}^T \alpha^p}, \quad (5.5)$$

$$\alpha^n \leftarrow \alpha^n \times f_n; \text{ where } f_n \leftarrow \frac{n \times C}{4 \times \mathbf{e}^T \alpha^n}. \quad (5.6)$$

Then the final solution is represented by α as follows:

$$\alpha \leftarrow [\alpha^p, \alpha^n]. \quad (5.7)$$

Algorithm 5.1 Random Key Encoding *genAlpha*(n_p, n_n, d)

Require: n_p :Number of positive class samples in training dataset.

n_n :Number of negative class samples in training dataset.

d :Number of dimensions of sample vector.

- 1: $n \leftarrow n_p + n_n$;
 - 2: $C \leftarrow \text{rand_int}(d, 1)$; {random integer in the range 1 to d }
 - 3: $s_p \leftarrow \text{rand_int}(n_p, 1)$;
 - 4: $\mathbf{r}^p \leftarrow \text{rand_int}(n_p, s_p)$; { s_p random integers in the range 1 to n_p }
 - 5: $\alpha_r^p \leftarrow |\mathcal{N}(s_p \times 1)|$;
 - 6: $f_p \leftarrow \frac{n \times C}{4 \times \mathbf{e}^T \alpha_r^p}$;
 - 7: $\alpha^p \leftarrow \alpha_r^p \times f_p$;
 - 8: $s_n \leftarrow \text{rand_int}(n_n, 1)$;
 - 9: $\mathbf{r}^n \leftarrow \text{rand_int}(n_n, s_n)$;
 - 10: $\alpha_r^n \leftarrow |\mathcal{N}(s_n \times 1)|$;
 - 11: $f_n \leftarrow \frac{n \times C}{4 \times \mathbf{e}^T \alpha_r^n}$;
 - 12: $\alpha^n \leftarrow \alpha_r^n \times f_n$;
 - 13: $\alpha \leftarrow [\alpha^p, \alpha^n]$;
 - 14: **return** α ;
-

5.1.2 Initial population generation

We generate initial population \mathbf{A} of size m using Algorithm 5.2.

$$\mathbf{A} \leftarrow \{\alpha_j\}_{j=1}^m \quad (5.8)$$

Algorithm 5.2 Initial Population Generation

Require: m :Size of the initial population, n_p :Number of positive class samples in training dataset, n_n :Number of negative class samples in training dataset, d : Number of dimensions of sample vector.

- 1: Initialize $A[m]$;
 - 2: **for** $j = 1 \rightarrow m$ { in parallel } **do**
 - 3: $\alpha_j \leftarrow \text{genAlpha}(n_p, n_n, d)$; {using Algorithm-5.1}
 - 4: $A[j] \leftarrow \alpha_j$;
 - 5: **end for**
 - 6: **return** A ;
-

5.1.3 Fitness evaluation

In order to evaluate the fitness of a solution α , the objective function $J(\alpha)$ in Equation (5.1) is used as the fitness function. The fitness value f_j for j^{th} solution α_j is given by

$$f_j = \alpha_j^T \mathbf{e} - \frac{1}{2} \alpha_j^T \mathbf{Q} \alpha_j. \quad (5.9)$$

Equation (5.9) gives the fitness of single candidate solution only. In order to utilize the GPUs efficiently, we can calculate the fitness of all m , $\alpha_j \in \mathbf{A}$ as:

$$\mathbf{f} \leftarrow \mathbf{A} \times \mathbf{e} - \frac{1}{2} ((\mathbf{A} \times \mathbf{Q}) \cdot \mathbf{A}) \times \mathbf{e}. \quad (5.10)$$

5.1.4 Selection operator

For selection, *roulette wheel* selection is used, however other methods such as rank selection can also be used. The fitness value of each $\alpha_j \in \mathbf{A}$ is used to associate a probability of selection. Let f_j be the fitness of α_j , then its probability of being selected is given by

$$p_j = \frac{f_j}{\sum_{k=1}^m f_k}. \quad (5.11)$$

5.1.5 Reproduction operator

For reproduction, we use only crossover. The crossover operation is a random r-site crossover in which two parents generate four children. As shown in Fig. 5.1-(B), it randomly selects two solutions α_1 and α_2 from mating pool and separates them into α_1^p , α_1^n , α_2^p , and α_2^n . Random key crossover is applied separately on pairs i.e. α_1^p , α_2^p and α_1^n , α_2^n . The random key crossover generates random integer indices k^p and k^n in the range 1 to n_p and 1 to n_n , respectively. And the values of α_1^p and α_1^n are exchanged with α_2^p and α_2^n at the respective

indices in k^p and k^n . However, $\alpha_1^p, \alpha_2^p, \alpha_1^n, \alpha_2^n$ may violate the constraint in Equation (5.2) due to exchange of values. So, in order to meet the constraint in Equation (5.2), the error i.e. the difference in the sum of values exchanged is calculated and adjusted. Then, we get the updated values of $\alpha_1^p, \alpha_2^p, \alpha_1^n, \alpha_2^n$ which will result into four new solutions:

$$\mathbf{c}_1 = [\alpha_1^p, \alpha_1^n], \quad (5.12)$$

$$\mathbf{c}_2 = [\alpha_1^p, \alpha_2^n], \quad (5.13)$$

$$\mathbf{c}_3 = [\alpha_2^p, \alpha_1^n], \quad (5.14)$$

$$\mathbf{c}_4 = [\alpha_2^p, \alpha_2^n]. \quad (5.15)$$

The complete procedure of the new solution generation using random r -site crossover is given in Algorithm 5.3.

5.1.6 Elitism

Lets us consider the initial population size $m = 100$. Then, the population at $(g + 1)^{th}$ generation retains 4-best solutions from g^{th} generation. And 92 new solutions are reproduced using 23(= 92/4) crossover operations using Equation (5.3) and the remaining 4 are the new solutions generated using Algorithm 5.1 as generated in the initial population.

The proposed Genetic-SVM solves the QP problem in Equation (5.1) with results comparable to SMO. However, time taken on a single processor is too high. In order to reduce the training time, we perform distributed computations in cloud environment as presented in the next section.

5.2 Distributed execution of Genetic-SVM

The proposed genetic algorithm based QP solver is able to get the best solution for Equation (5.1) but the time taken is too high. However, unlike sequential minimal optimization (SMO), the proposed solver is easy to distribute. For Genetic-SVM, we can utilize distributed environments like GPU enabled HPC or cloud clusters etc. Here, we present two distributed frameworks for Genetic-SVM over the GPU enabled cluster. The proposed frameworks work according to the available resources and size of the dataset. The first distributed Genetic-SVM framework, run multiple instances of the algorithm and share the best solution among each others in order to achieve fast convergence. The second framework further distribute the task of a single instance of the algorithm for a large dataset.

Algorithm 5.3 Random r -Site Crossover

Require: α_1 :First Parent, α_2 :Second Parent, n_p :Number of positive class samples in training dataset, n_n :Number of negative class samples in training dataset.

```
1:  $r \leftarrow \text{rand\_int}(n_p, 1)$ ;  
2:  $k^p \leftarrow \text{rand\_int}(n_p, r)$ ;  
3:  $\alpha_1^p \leftarrow \{\alpha_{1i}^p\}_{i=1}^{n_p}$ ;  $\alpha_2^p \leftarrow \{\alpha_{2i}^p\}_{i=1}^{n_p}$ ;  
4:  $\mathbf{t}_1^p \leftarrow \{\alpha_{1i}^p\}_{i=k_1^p}^{k_1^p}$ ;  $\mathbf{t}_2^p \leftarrow \{\alpha_{2i}^p\}_{i=k_1^p}^{k_1^p}$ ;  
5:  $\{\alpha_{1k_i^p}^p \leftarrow t_{2i}^p\}_{i=1}^r$ ;  $\{\alpha_{2k_i^p}^p \leftarrow t_{1i}^p\}_{i=1}^r$ ;  
6:  $\epsilon \leftarrow \frac{e^{\mathbf{t}_1^p} - e^{\mathbf{t}_2^p}}{2}$   
7: if  $\epsilon > 0$  then  
8:    $l \leftarrow \text{rand\_int}(n_p, 1)$ ;  
9:    $\alpha_{1l}^p \leftarrow \alpha_{1l}^p + \epsilon$ ;  
10:  while  $\epsilon \neq 0$  do  
11:     $l \leftarrow \text{rand\_int}(n_p, 1)$ ;  
12:    if  $\alpha_{2l}^p \geq \epsilon$  then  
13:       $\alpha_{2l}^p \leftarrow \alpha_{2l}^p - \epsilon$ ;  $\epsilon = 0$ ;  
14:    else  
15:       $\alpha_{2l}^p \leftarrow 0$ ;  $\epsilon \leftarrow \epsilon - \alpha_{2l}^p$   
16:    end if  
17:  end while  
18: else  
19:    $\epsilon \leftarrow |\epsilon|$ ;  
20:    $l \leftarrow \text{rand\_int}(n_p, 1)$ ;  
21:    $\alpha_{2l}^p \leftarrow \alpha_{2l}^p + \epsilon$ ;  
22:   while  $\epsilon \neq 0$  do  
23:      $l \leftarrow \text{rand\_int}(n_p, 1)$ ;  
24:     if  $\alpha_{1l}^p \geq \epsilon$  then  
25:        $\alpha_{1l}^p \leftarrow \alpha_{1l}^p - \epsilon$ ;  $\epsilon = 0$ ;  
26:     else  
27:        $\epsilon \leftarrow \epsilon - \alpha_{1l}^p$ ;  $\alpha_{1l}^p \leftarrow 0$ ;  
28:     end if  
29:   end while  
30: end if  
31: Similarly calculate  $\alpha_1^n$  and  $\alpha_2^n$ .  
32:  $\mathbf{c}_1 = [\alpha_1^p, \alpha_1^n]$ ;  $\mathbf{c}_2 = [\alpha_1^p, \alpha_2^n]$ ;  $\mathbf{c}_3 = [\alpha_2^p, \alpha_1^n]$ ;  $\mathbf{c}_4 = [\alpha_2^p, \alpha_2^n]$ ;  
33: return  $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}$ ;
```

5.2.1 Distributed Genetic-SVM

The first framework is applicable when one virtual machine (VM) is able to store the data in physical memory but training time is too high. Here, we are considering availability of virtual resources provisioned over cloud environment. As shown in Fig. 5.2, we launch

multiple instance of the GPU enabled virtual machines (VMs). One VM acts as master VM and all others act as worker VMs. Here, we maintain a global pool, $\mathbf{A}_G = \{\alpha_{(k)}\}_{k=1}^N$ at master VM and a local pool $\mathbf{A}_L^k = \{\alpha_j\}_{j=1}^m$ at k^{th} worker VM, $k = 1, 2, \dots, N$. The kernel matrix \mathbf{Q} is copied to all the worker VMs. Each worker VM generates the initial population, then do the fitness evaluation and send the best solution to the master VM. Master VM collects all the local solutions in the global pool \mathbf{A}_G , then it selects the global solution from the local solutions, and then broadcasts the best solution to all worker VMs. Further, each worker VM prepares the next generation which consists of global best solution, local best solution (if not winner worker VM), reproduced children solutions from previous generation solutions, and randomly generated solutions. This process is repeated until convergence. The fitness value (f) is calculated using Equation 5.10. Also, in this process, the N worker VMs send only best solution, thus, total N messages are passed over the network after each generation. This leads to very low communication which is of the order of $O(N)$. The sharing of best solutions leads to fast convergence.

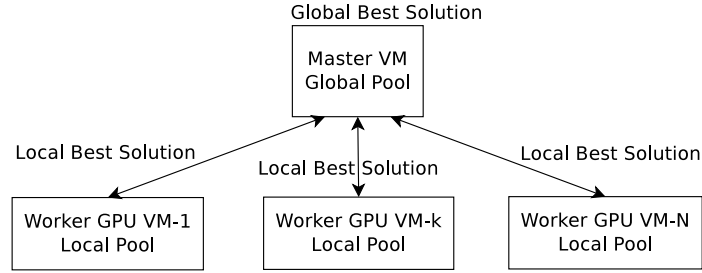


Figure 5.2: Proposed architecture of distributed Genetic-SVM

5.2.2 Distributed Genetic-SVM for large dataset

The first distributed framework i.e. *Distributed Genetic-SVM* is not applicable for large dataset. Because the size of kernel matrix increases quadratically $O(n^2)$ with an increase in dataset size n . Thus, for a large dataset it is not an efficient way to store kernel matrix in one worker VM and execute the task. Thus in this framework for distributed Genetic-SVM, we distribute the kernel matrix \mathbf{Q} into L sub-worker VMs with GPU support, while worker VMs do not require GPU support as shown in Fig. 5.3. Each sub-worker VM with identifier $l = 1, 2, \dots, L$ contains $\mathbf{Q}^l = \{\{q_{ij}\}_{i=1}^n\}_{j=\frac{(l-1)n}{L}}^{\frac{ln}{L}}$, a part of kernel matrix \mathbf{Q} , having size $n \times \frac{n}{L}$. The partial fitness f^l is calculated at each sub-worker machine as follows:

$$\mathbf{P} \leftarrow \mathbf{A} \times \mathbf{Q}^l, \quad (5.16)$$

$$\mathbf{A}^l \leftarrow \left\{ \left\{ a_{ij} \right\}_{i=1}^m \right\}_{j=\frac{(l-1)n}{L}}^{\frac{ln}{L}}, \quad (5.17)$$

$$\mathbf{P} \leftarrow \mathbf{P} \cdot \mathbf{A}^l, \quad (5.18)$$

$$\mathbf{f}^l \leftarrow (\mathbf{A}^l \times \mathbf{e} + \frac{1}{2} \mathbf{P} \times \mathbf{e}). \quad (5.19)$$

Finally, the fitness value \mathbf{f} is calculated as

$$\mathbf{f} \leftarrow \sum_{l=1}^L \mathbf{f}^l. \quad (5.20)$$

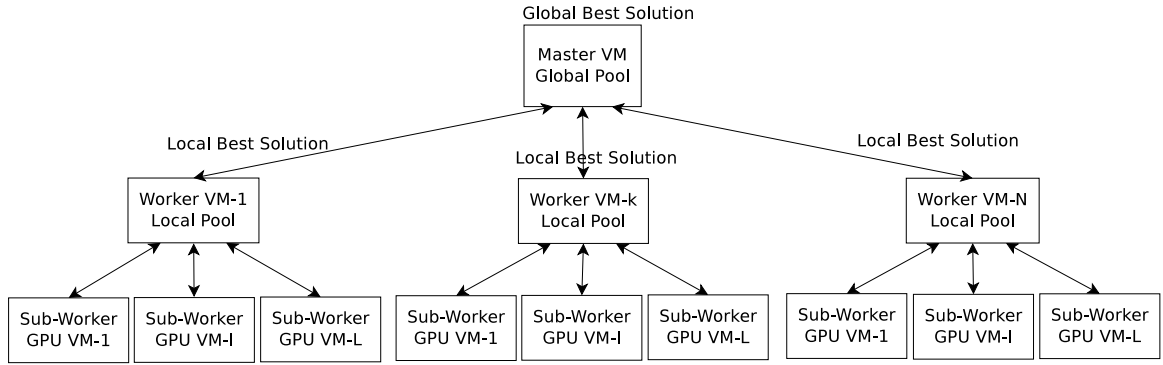


Figure 5.3: Proposed architecture of distributed Genetic-SVM for large dataset

5.3 Experiments and results

The genetic algorithm is implemented in C/C++, CUDA, and OpenMPI over a GPU cluster running Ubuntu 14.04. The cluster contains two machines with specifications: 1) First machine has 2 Intel Xeon processors with 12 core each, 64GB physical memory and 6 Nvidia Tesla K20Xm GPUs with 5GB device memory each. 2) Second machine has 2 Intel Xeon processors with 24 core each, 128GB physical memory, 2 Nvidia Tesla K20c GPUs with 6GB device memory each. The large matrix multiplications are accelerated using GPUs. We have also successfully tested Genetic-SVM on HPC with 512 nodes and on the Amazon Elastic Compute Cloud (EC2) using StarCluster [77]. StarCluster is a tool for dynamically creating, managing cluster on Amazon EC2 for testing MPI programs. Table 5.1 provides the details of the datasets used in the experiments.

The results in Fig. 5.4-(A) show the fitness values of candidates in the pool after 2400 epochs and the results in Fig. 5.4-(B) reflect the improvement to the fitness index over the epochs. The presented experiment is conducted on the GISETTE dataset of OCR published

Table 5.1: Details of datasets used to evaluate the performance of Genetic-SVM

Dataset	Dimensions	Training Size	Test Size
GISETTE	5000	6000	1000
ADULT (A1A)	123	1605	30956
ADULT (A2A)	123	2265	30296
ADULT (A3A)	123	3185	29376
ADULT (A4A)	123	4781	27780
ADULT (A5A)	123	6414	26147
ADULT (A6A)	123	11220	21341
ADULT (A7A)	123	16100	16461
ADULT (A8A)	123	22696	9865
ADULT (A9A)	123	32561	16281
MUSHROOMS	112	5000	3124
SVMGUIDE1	4	3089	4000

during a NIPS challenge. The results show that the classification performance of the proposed approach is very close to sequential SVM. The results in Table 5.2 show the good classification ability of the proposed algorithm with a negligible loss of accuracy which can be further reduced by running algorithm for more number of generations.

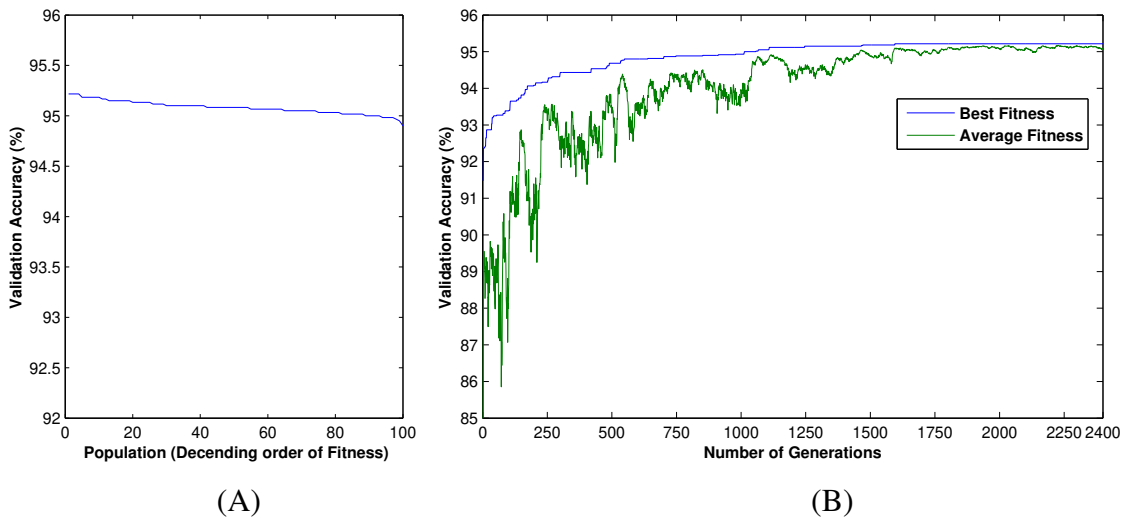


Figure 5.4: Performance of classification for Genetic-SVM on the GISETTE dataset after 2400 epoch.

Fig. 5.5 shows the performance of classification while running the Genetic-SVM algorithm multiple times. The results show very low standard deviations when running 10 times. Also, Fig. 5.5 shows that the proposed approach obtains the significant improvements in first few hundred iterations only, which shows the suitability of the encoding

method and crossover operations used for generating new solutions. Fig. 5.6 shows the time taken by 100 worker VMs. Finally, when running the complete pipeline of the algorithm on various datasets, the Genetic-SVM algorithm performs approximately 10-20 times faster than the LIBSVM as shown in Table 5.3.

Table 5.2: Performance of classification (%) of the Genetic-SVM and comparison with SMO using LIBSVM

DataSet Used	SMO Accuracy (%)	Genetic-SVM Loss (%)	Accuracy Accuracy (%)
GISSETTE	97.60	97.60	0.0
ADULT (A1A)	83.59	83.19	-0.4
ADULT (A2A)	83.98	83.28	-0.7
ADULT (A3A)	83.84	83.54	-0.3
ADULT (A4A)	83.96	83.26	-0.7
ADULT (A5A)	84.17	83.37	-0.8
ADULT (A6A)	84.17	83.27	-0.9
ADULT (A7A)	84.58	83.78	-0.8
ADULT (A8A)	85.01	84.31	-0.7
ADULT (A9A)	84.82	84.52	-0.3
MUSHROOMS	97.09	96.39	-0.7
SVMGUIDE1	66.93	66.33	-0.6

Table 5.3: Training time (seconds) of the Genetic-SVM and comparison with sequential SVM

Dataset Used	Sequential SVM (Seconds)	Genetic-SVM (Mean±Var.) (Seconds)	Scaling
GISSETTE	214	9.2091±1.3368	≈ 20×
ADULT (A7A)	11.84	0.8013 ± 0.1307	≈ 15×
ADULT (A8A)	22.97	1.4556 ± 0.2023	≈ 15×
ADULT (A9A)	45.85	2.5473 ± 0.7359	≈ 18×

The proposed Genetic-SVM performs better than existing partitioning based distributed SVMs approaches in terms of classification accuracy and time taken in training a SVM model. The proposed approach successfully achieves a comparable accuracy to sequential SVM for GISSETTE dataset. Along with improvement in accuracy, proposed approach also performs approximately 3 times faster than the approach by You *et al.* [132]. Also, it can be observed that the loss of accuracy is less than 0.9% on other datasets, which demonstrates the efficacy of the proposed approach.

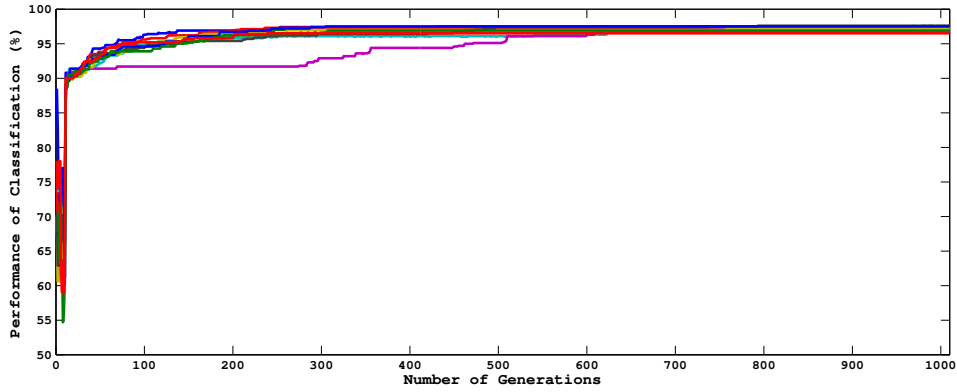


Figure 5.5: Performance of genetic algorithm based optimization of QP problem for 10 runs using population size 1000 and pool size 2000 at each slave process and using population size 100 and pool size 1000 at master process.

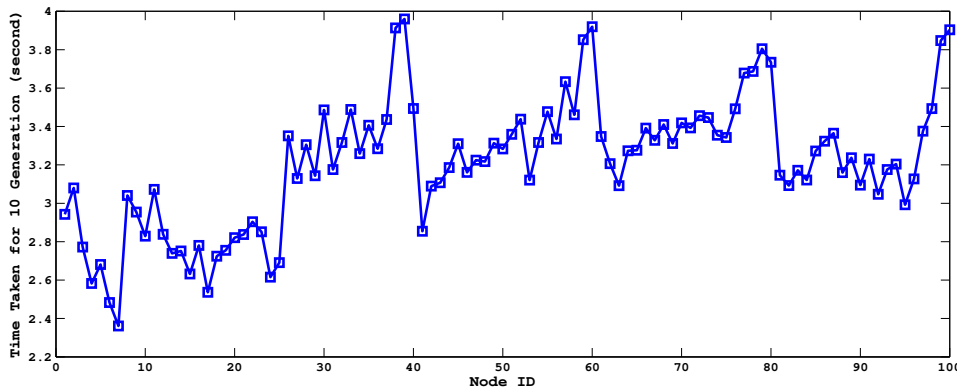


Figure 5.6: Time taken by each process for 10 generations, each for population size 1000 and pool size 2000 at work VMs.

5.4 Summary

The partitioning based distributed SVMs have generally proven to be faster than sequential SVMs on large datasets. However, classification performance still lags behind. In the proposed Genetic-SVM, we aimed at providing a distributed SVM approach which retains or improves the classification performance of sequential SVM while having the computational time gains as of distributed approaches on a large dataset. The Genetic-SVM shows success in order to find the better solution quickly and also the computations are efficiently distributed over GPU cloud cluster to leverage the benefit of the GPUs for large matrix multiplication. The experiments show better performance in terms of classification accuracy as well as computational time.

Chapter 6

DiP-SVM : Distribution preserving distributed kernel SVM

The Genetic-SVM proposed in the previous chapter relies on the availability of large pool of computational resources thus cannot scale much because of resource constraints. In this chapter, we propose a data partitioning based approach for scaling the training of kernel SVM. In data partitioning approaches, the task of learning a support vector machine for large datasets has been performed by splitting the dataset into manageable sized “partitions” and training a sequential support vector machine on each of these partitions separately to obtain local support vectors. However, this process invariably leads to the loss in classification accuracy as global support vectors may not have been chosen as local support vectors in their respective partitions. We hypothesize that retaining the original distribution of the dataset in each of the partitions can help solve this issue. Hence, we propose DiP-SVM, a distribution preserving kernel support vector machine where the first and second order statistics of the entire dataset are retained in each of the partitions. This helps in obtaining local decision boundaries which are in agreement with the global decision boundary, thereby reducing the chance of missing important global support vectors. We show that DiP-SVM achieves a minimal loss in classification accuracy among other distributed support vector machine techniques on several benchmark datasets. Also, few of the existing approaches [40] do not rely on local support vectors and instead transfer all the data points to next level which leads to very high communication overhead. We further demonstrate that our approach reduces communication overhead between partitions leading to faster execution on large datasets and making it suitable for implementation in distributed environments.

The rest of the chapter is organized as follows. The details of the distribution preserving partitioning are presented in section 6.1. Section 6.2 presents the process of distributed learning in a cluster. Section 6.3 discusses the experimental setup and results. We summa-

size in section 6.4 with future directions.

6.1 Distribution preserving partitioning

The proposed DiP-SVM approach operates in two distinct phases, namely, *distribution preserving partitioning (DPP)* phase and *distributed learning* phase. In this section, we present a detailed discussion of the distribution preserving partitioning phase which aims for the balanced partitioning of data points while preserving the statistical properties of the entire dataset. Let $\mathbf{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is the dataset consisting of N data points with mean $\boldsymbol{\mu}_{\mathbf{D}}$ and variance $\boldsymbol{\Sigma}_{\mathbf{D}}$. In this phase, we divide \mathbf{D} into P balanced partitions $\{\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_P\}$, each of size $N_p = \lceil \frac{N}{P} \rceil$ such that the p^{th} partition can be expressed as

$$\mathbf{D}_p = \{(\mathbf{x}_{\pi(n)}, y_{\pi(n)})\}_{n=1}^{N_p}, \quad (6.1)$$

with mean $\boldsymbol{\mu}_{\mathbf{D}_p}$ and variance $\boldsymbol{\Sigma}_{\mathbf{D}_p}$. The mapping function $\pi(n)$ gives the corresponding index in \mathbf{D} of a point at index n in \mathbf{D}_p . These partitions are created in such a manner that the first and second order statistics (mean and variance) of each partition is approximately close to the given dataset \mathbf{D} which can be expressed as the following objective functions:

$$\min \sum_{p=1}^P \|\boldsymbol{\mu}_{\mathbf{D}_p} - \boldsymbol{\mu}_{\mathbf{D}}\| \text{ and } \min \sum_{p=1}^P \|\boldsymbol{\Sigma}_{\mathbf{D}_p} - \boldsymbol{\Sigma}_{\mathbf{D}}\|. \quad (6.2)$$

Also, we want each partition to retain the ratio of the number of points in each class as in the complete dataset \mathbf{D} . So, we use the partitioning approach separately on each of the constituent classes \mathbf{D}_c instead of \mathbf{D} . To formalize this idea more clearly, we represent \mathbf{D} as a collection of \mathcal{C} classes:

$$\mathbf{D} = \{\mathbf{D}_c\}_{c=1}^{\mathcal{C}}, \quad (6.3)$$

where the c^{th} class contains N_c data points. The research work proposed in [138] attempts to achieve similar objective, however, this method is highly dependant on the initial sets chosen and thus may lead to partitions whose distribution may not be balanced. In order to solve the objective functions stated in Equation 6.2, we employ K -means clustering on each of the classes separately whose objective is defined as

$$\arg \min_{\mathbf{D}_c} \sum_{k=1}^K \sum_{\mathbf{x} \in D_{ck}} \|\mathbf{x} - \boldsymbol{\mu}_{ck}\|^2, \quad (6.4)$$

where K is the number of clusters, \mathbf{D}_{ck} is the k^{th} cluster of the c^{th} class and $\boldsymbol{\mu}_{ck}$ is the

mean of points in \mathbf{D}_{ck} . From each cluster \mathbf{D}_{ck} having N_{ck} points, P balanced partitions $\{\mathbf{D}_{ck}^p\}_{p=1}^P$ are created, each containing $N_{ck}^p = \lceil \frac{N_{ck}}{P} \rceil$ points selected according to uniform distribution. This is carried out for all the \mathcal{C} classes using Algorithm 6.1.

Algorithm 6.1 Distribution Preserving Partitioning (DPP)

Input: \mathbf{D} : $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$, $y_n \in \{c = 1, 2, \dots, \mathcal{C}\}$, N : #instance (vectors) in \mathbf{D} , d : #dimensions, \mathcal{C} : #classes in the dataset, P : #partitions, K : #clusters, $\mathcal{U}(N_{ck}^p, N_{ck})$: generate N_{ck}^p random indexes in range $[1 - N_{ck}]$.

Output: $\{\mathbf{D}_p\}_{p=1}^P$: partitions.

- 1: $\{\mathbf{D}_c\}_{c=1}^{\mathcal{C}} \leftarrow \text{groupClasswise}(\mathbf{D}, K)$; //where \mathbf{D}_c is the set of points belongs to c_{th} class.
 - 2: **for** $c = 1 \rightarrow \mathcal{C}$ **do**
 - 3: $\{\mathbf{D}_{ck}\}_{k=1}^K \leftarrow \text{kmeans}(\mathbf{D}_c, K)$; //where \mathbf{D}_{ck} is the k_{th} cluster of the c_{th} class.
 - 4: **for** $k = 1 \rightarrow K$ **do**
 - 5: **for** $p = 1 \rightarrow P$ **do**
 - 6: $N_{ck}^p \leftarrow \lceil \frac{N_{ck}}{P} \rceil$;
 - 7: $\mathbf{D}_{ck}^p \leftarrow \mathbf{D}_{ck}[\mathcal{U}(N_{ck}^p, N_{ck})]$
 - 8: $\mathbf{D}_p \leftarrow \mathbf{D}_p \cup \mathbf{D}_{ck}^p$;
 - 9: $\mathbf{D}_{ck} \leftarrow \mathbf{D}_{ck} - \mathbf{D}_{ck}^p$;
 - 10: $N_{ck} \leftarrow N_{ck} - N_{ck}^p$;
 - 11: **end for**
 - 12: **end for**
 - 13: **end for**
 - 14: **return** $\{\mathbf{D}_p\}_{p=1}^P$;
-

Fig. 6.1, illustrate an example of full dataset \mathbf{D} and one of its partition \mathbf{D}_p obtained by using Algorithm 6.1 for the MNIST dataset. It can be observed that partition \mathbf{D}_p is a sparse representation of the full training data set \mathbf{D} . Also, the statistical properties of the partitions $\{\mathbf{D}_p\}_{p=1}^P$ are approximately close to the statistical properties of entire dataset \mathbf{D} . From the maximum likelihood estimation (MLE), it is intuitive that mean and variance of the entire dataset \mathbf{D} containing N samples (N is very large) will be approximately close to mean and variance of its partition \mathbf{D}_p for sufficiently large size N_p , i.e. $\mu_{\mathbf{D}_p} \cong \mu_{\mathbf{D}}$ and $\Sigma_{\mathbf{D}_p} \cong \Sigma_{\mathbf{D}}$. In order to justify this statement, we empirically show that for K number of clusters (K is large), mean and variance of partitions obtained using Algorithm 6.1 are approximately close to the entire dataset. Table 6.1 presents a comparison between the mean and variance of the partitions formed using the proposed approach and random partitioning approach [111, 6] on the kddcup99 dataset ($K = 1000$ and $P = 100$). It can be seen that the partitions formed using DPP are up to $10^3 \times$ closer to the mean and variance of the entire dataset than the random partitions.

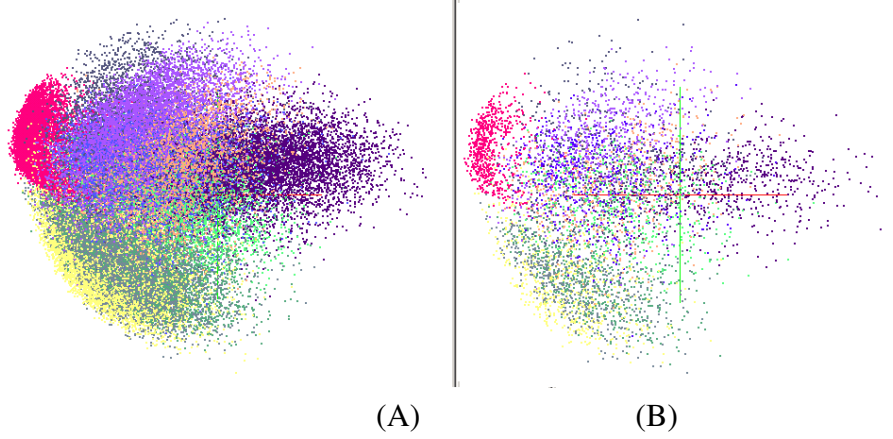


Figure 6.1: (A) MNIST [120] dataset containing 60,000 samples. (B) A sample partition generated using Algorithm 6.1. Colors represent various classes.

Table 6.1: Distortion in the distributions of partitions for random partitioning vs. proposed distribution preserving partitioning on kddcup99 dataset

	Min	Mean±Deviation	Max
Random $\ \mu_{D_p^*} - \mu_D\ $	1.76e-06	1.21e-05±9.15e-06	4.84e-05
Random $\ \Sigma_{D_p^*} - \Sigma_D\ $	1.14e-05	4.63e-05±2.78e-05	1.83e-04
Proposed $\ \mu_{D_p} - \mu_D\ $	7.53e-08	1.10e-07±1.99e-08	1.60e-07
Proposed $\ \Sigma_{D_p} - \Sigma_D\ $	1.02e-06	1.32e-06±1.90e-07	1.78e-06

6.2 Distributed execution of DiP-SVM

After data partitioning, we use a modified cascade SVM for distributed learning of support vectors. For each partition D_p obtained using Algorithm 6.1, a sequential SVM model is trained independently using Equation 6.5 at level l ($=0$ initially) which results in local support vectors $S_p^{(l)}$. At level l for p^{th} partition, SMO solves the following dual objective function as defined in [24]:

$$\max_{\alpha_p^{(l)}, b_p^{(l)}} \sum_{i=1}^{N_p} \alpha_i - \frac{1}{2} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i^T, \mathbf{x}_j), \text{ s.t. } \sum_{i=1}^{N_p} \alpha_i y_i = 0, \text{ and } 0 \leq \alpha_i \leq C, \quad (6.5)$$

where $\alpha_i \in \alpha_p^{(l)}$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_p} \in D_p^{(l)}$, and $b_p^{(l)}$ is the bias. At p^{th} virtual machine (VM), the local support vectors $S_p^{(l)}$ are those points having $\alpha_i > 0$ i.e.

$$S_p^{(l)} = D_p^{(l)} \{\alpha_p^{(l)} > 0\} \text{ and } \alpha_p^{(l)} = \alpha_p^{(l)} \{\alpha_p^{(l)} > 0\}. \quad (6.6)$$

The distance $\gamma_{\mathbf{x}_i}$ of each point $(\mathbf{x}_i, y_i) \in \mathbf{D}_p^{(l)}$ from the hyperplane is calculated as

$$\gamma_{\mathbf{x}_i} = \sum_{(\mathbf{x}, y) \in \mathbf{S}_p^{(l)}, \alpha \in \alpha_p^{(l)}} \left(\alpha y K(\mathbf{x}^T, \mathbf{x}_i) + b_p^{(l)} \right). \quad (6.7)$$

All those points having $\gamma_{\mathbf{x}_i} > T_\gamma$ are considered as relevant vectors $\mathbf{R}_p^{(l)}$.

$$\mathbf{R}_p^{(l)} = \mathbf{D}_p^{(l)} \{ \gamma_p^{(l)} \geq T_\gamma \} \text{ and } \alpha_p^{(l)} = \alpha_p^{(l)} \{ \gamma_p^{(l)} \geq T_\gamma \}. \quad (6.8)$$

Here, T_γ is the threshold on distance from the local hyperplane. For $T_\gamma = 1$, only vectors which are on or within the margin are selected. If $T_\gamma < 1$, the number of points transferred over the network reduces but it may increase the loss of accuracy as some key points may be missed. For $T_\gamma > 1$, more points than the points which are on or within the margin are passed over the network which may reduce the loss of accuracy incurred.

These relevant vectors $\mathbf{R}_p^{(l)}$ at level l are then collected and used as training data to learn an SVM model in the next level. The training data for partition p at level $l + 1$ ($\mathbf{D}_p^{(l+1)}$) and the Lagrangian multipliers $\alpha_p^{(l+1)}$ are calculated as follows:

$$\mathbf{D}_p^{(l+1)} = \{ \mathbf{R}_p^{(l)}, \mathbf{R}_{p+\lceil P/2 \rceil}^{(l)} \} \text{ and } \alpha_p^{(l+1)} = \{ \alpha_p^{(l)}, \alpha_{p+\lceil P/2 \rceil}^{(l)} \}. \quad (6.9)$$

This process is repeated $L - 1$ times. Finally, at level $l = L$, it produces the final model which constitutes the global support vectors \mathbf{S} , global Lagrangian multipliers α , and bias b . This is explained in Algorithm 6.2. We make adjustments to T_γ to achieve a trade-off between communication overhead and loss of accuracy.

6.2.1 Empirical evaluation

To show that the LSVs obtained from DiP-SVM are very close to the global support vectors in comparison to other recent methods for partitioning in [132, 40], we consider two cases that can arise: 1) data that can be well clustered and 2) data that has considerable overlap. In the first case, we use a synthetic dataset, which is a mixture of four 2D Gaussian distributions and is well separable into two clusters. In Fig. 6.2, we can see the two clusters obtained by the methods in [132, 40] contain data from both the classes. The local support vectors obtained from both the partitions are dissimilar causing the local decision hyperplanes to completely contradict each other. The final global decision hyperplane also shows high deviation from the decision hyperplane produced using a sequential SVM. On the other hand, the proposed DiP-SVM method produces local and global decision hyperplanes which show high correspondence to each other as well as to the decision hyperplane

Algorithm 6.2 DiP-SVM Learning

Input: $\mathbf{D} : \{(x_n, y_n)\}_{n=1}^N$, $x_n \in \mathbb{R}^d$, $y_n \in \{-1, +1\}$, $N : \#instance$ (vectors) in \mathbf{D} , $d : \#dimensions$, $P : \#partitions$, $K : \#clusters$, $T_\gamma : \text{Threshold}$, $params : \{KernelFunction, KernelParameters\}$.

Output: $\mathbf{S} : \text{Global support vectors}$, $\boldsymbol{\alpha} : \text{Global lagrangian multipliers}$.

```
1:  $\{\mathbf{D}_p\}_{p=1}^P \leftarrow DPP(\mathbf{D}, P, K)$ ;  
2: Level  $l \leftarrow 0$ ;  $\boldsymbol{\alpha}_p^{(l)} \leftarrow 0$ ;  
3: while true do  
4:   for  $p = 1 \rightarrow P$  {Parallely over cluster} do  
5:      $\{\boldsymbol{\alpha}_p^{(l)}, b_p^{(l)}\} \leftarrow svm\_train(\mathbf{D}_p^{(l)}, \boldsymbol{\alpha}_p^{(l)}, params)$ ;  
6:      $\mathbf{S}_p^{(l)} \leftarrow \mathbf{D}_p^{(l)} \{\boldsymbol{\alpha}_p^{(l)} > 0\}$ ;  
7:     if  $P = 1$  then  
8:        $\mathbf{S} \leftarrow \mathbf{S}_1^{(l)}$ ;  
9:       return  $\mathbf{S}$ ;  
10:    else  
11:       $\gamma_p^{(l)} \leftarrow svm\_score(\mathbf{S}_p^{(l)}, \boldsymbol{\alpha}_p^{(l)}, b_p^{(l)}, \mathbf{D}_p^{(l)})$ ;  
12:       $\mathbf{R}_p^{(l)} \leftarrow \mathbf{D}_p^{(l)} (\gamma_p^{(l)} \geq T_\gamma)$ ;  
13:      if  $p > \lceil P/2 \rceil$  then  
14:        Send  $\{\mathbf{R}_p^{(l)}, \boldsymbol{\alpha}_p^{(l)}\}$  to  $VM_{p-\lceil P/2 \rceil}$ ;  
15:      else  
16:        Receive  $\{\mathbf{R}_{p+\lceil P/2 \rceil}^{(l)}, \boldsymbol{\alpha}_{p+\lceil P/2 \rceil}^{(l)}\}$  from  $VM_{p+\lceil P/2 \rceil}$ ;  
17:         $\mathbf{D}_p^{(l+1)} \leftarrow combine(\mathbf{R}_p^{(l)}, \mathbf{R}_{p+\lceil P/2 \rceil}^{(l)})$ ;  
18:         $\boldsymbol{\alpha}_p^{(l+1)} \leftarrow combine(\boldsymbol{\alpha}_p^{(l)}, \boldsymbol{\alpha}_{p+\lceil P/2 \rceil}^{(l)})$ ;  
19:      end if  
20:    end if  
21:  end for  
22:   $P \leftarrow \lceil P/2 \rceil$ ;  $l \leftarrow l + 1$ ;  
23: end while
```

of sequential SVM (cosine similarity ≈ 1). This shows the suitability of the DiP-SVM over the existing clustering-based methods in [40, 132] for well-separated clusters.

The case for DiP-SVM grows stronger in a case of overlapping clusters as it not only produces decision boundaries at the local level which are in strong agreement among themselves but also preserves the LSVs which contribute to the global decision boundary. Fig. 6.3 demonstrates the suitability of the DiP-SVM over the existing clustering-based methods in [40, 132] for overlapping clusters using Gaussian kernel. In order to show the effectiveness of the selected local SVs at any level we use the relevant SV index defined as

$$Relevant\ SV\ Index = \frac{|S^l \cap S^{l+1}|}{|S^l|}.$$

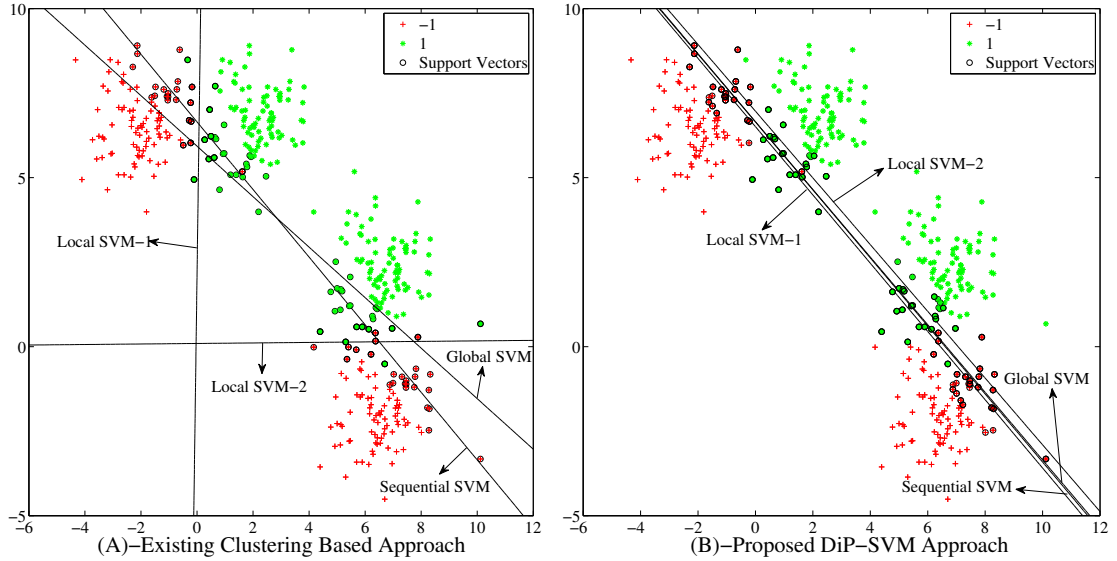


Figure 6.2: Comparison of DiP-SVM with the existing clustering based methods in [40] [132] for local and global solutions on well separable clusters having points from both the classes. (Best viewed in colors)

This index measures the number of SVs at level l which are also considered as SVs at level $l + 1$. In this experiment, we use a synthetic dataset, which is a mixture of four 2D Gaussian distributions. We compare the relevant SV index of the local SVs produced by various clustering based approaches with the GSVs produced at the last step as shown in Fig 6.3. It can be seen in Fig 6.3(B),(C) and (D) that a large number of LSVs produced by the existing clustering based SVMs [132, 40] are not included in GSVs. This makes the relevant SV index between final GSVs and the LSVs quite low (< 0.5). On the other hand, the LSVs produced by DiP-SVM as shown in Fig 6.3(E),(F) and (G) are in close agreement. A high relevant SV index of $\cong 1$ confirms this proposition. Further, the GSVs produced in Fig 6.3(G) is closer to the sequential SVM based SVs (relevant SV index of $\cong 1$) than the GSVs produced in Fig 6.3(D).

6.3 Experiments and results

In literature, most of the existing distributed SVM implementations are based on OpenMPI [34] or Hadoop. In our approach, we use OpenMPI architecture which is a default standard for distributed computing. The local SVMs as well as the global SVMs were trained using LIBSVM [15]. In order to evaluate the proposed approach, we conducted

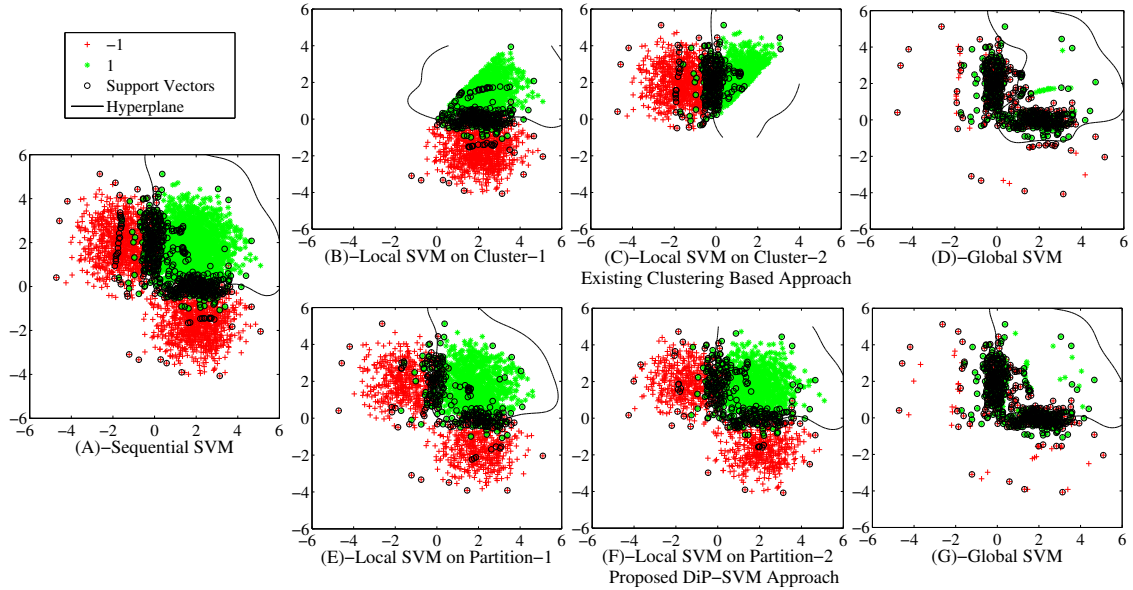


Figure 6.3: Comparison of DiP-SVM with the existing clustering based methods in [40] [132] for local and global solutions using non-linear kernel. (Best viewed in colors)

experiments on several real-world datasets from the domains like computer vision, cyber security, economics, text classification, etc. Table. 6.2 gives the statistics for each of these datasets.

Table 6.2: Details of the datasets used

Dataset	Application Domain	#Dim.	#Train	#Test
gisette [132]	Digit Classification	5000	6000	1000
adult [132]	Economics	123	32561	16281
ijcnn1 [132]	Text Classification	22	49990	91000
cifar [40]	Visual Recognition	3072	50000	10000
webspam[40]	Spam Detection	254	280000	70000
covtype [40]	Forest Classification	54	464810	116202
kddcup99[40]	Intrusion Detection	123	4898431	311029
mnist8m [40]	Digit Classification	784	8000000	100000
url	URL Classification	2396130	3131961	100000

Table 6.3 gives the performance comparison of DiP-SVM with Sequential LIBSVM, DC-SVM [40], CA-SVM [132] on the datasets as listed in Table. 6.2. All results are calculated in a cluster of 11 virtual machine where one virtual machine acts as a master node and remaining 10 virtual machines act as worker nodes. The SVM parameters C and γ for each dataset are selected using grid evaluation. It can be observed from the experiments that the DiP-SVM achieve better performance consistently irrespective of the distribution of the

data. The performance is evaluated in terms of loss of accuracy with respect to sequential SVM. Each dataset shows the only small change in the accuracy which is comparable to sequential SVM (less than 0.5%), and better than existing methods DC-SVM [40] and CA-SVM [132] as shown in Fig. 6.4. However, many times, it also shows small improvements too. This reduction in loss of accuracy is because of the proposed distribution preserving partitioning approach. Fig. 6.5 presents the training time performance which shows that the time take in training on various datasets by proposed DiP-SVM is comparatively less than the sequential SVM and existing distributed implementations of SVM. The results show that the training of DiP-SVM is approximate $9\times$ faster than the training of sequential SVM for each dataset.

Table 6.3: Comparison of classification performance (%) and average runtime (sec.)

Method	LIBSVM		DC-SVM		CA-SVM		Proposed		Change	
	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Scale
gisetete	97.70	125	97.60	299	96.00	81	97.90	29	+0.20	$4\times$
adult	85.08	761	84.79	78	83.00	121	84.01	58	-1.07	$13\times$
ijcnn1	98.69	20	98.53	318	90.16	121	98.61	3	-0.08	$7\times$
cifar	89.50	13892	80.15	22330	63.94	2143	89.49	2378	-0.01	$6\times$
webspam	99.28	15056	99.28	10485	99.11	3093	99.15	1942	-0.13	$8\times$
covtype	96.01	31785	95.95	17456	75.04	34025	93.07	3919	-2.94	$8\times$
kddcup99	99.57	37684	99.49	23346	NA	NA	99.53	3170	-0.04	$12\times$
mnist8m	99.91	$\approx 10^6$	99.91*	NA	NA	NA	99.99	99874	+0.08	$11\times$
url	96.75	486559	NA	NA	NA	NA	96.88	53374	+0.13	$9\times$

Acc.-Accuracy (%), Change-with respect to LIBSVM, NA - Not Available, *taken from [40]

Fig. 6.6 shows the performance of the DiP-SVM (Levels $L = 10$ with binary splitting) on kddcup99 dataset having ≈ 5 million records. It can be observed from the figure that a large portion of the irrelevant data ($\approx 96\%$) is eliminated at first layer itself as shown in Fig. 6.6-(A)&(B). Thus from the second layer onwards the amount of data remaining is quite less for calculating the global decision boundary which leads to low communication overhead. After the first level, very less data is eliminated as most of the data points in hand turn out to be support vectors in the subsequent levels as shown in Fig. 6.6-(C). Fig. 6.6-(D) shows that the data in all the nodes is evenly divided at each layer. Fig. 6.6-(E)&(F) show the amount of data transferred over the network at each level during training phase on respective datasets. This shows that the amount of data transferred decreases as the levels increase. Fig. 6.6-(G) shows the cumulative data transferred. The total data transferred is $\approx 8\%$ of the entire dataset. Fig. 6.6-(H) shows the time taken to train SVM model at each level for respective datasets. This figure shows that as the SVs are combined at each successive level, the computation time increases on an average. However, the top

level shows a reduction in the training time for this two possible reasons are 1) the use of Lagrangian multipliers from the previous level due to which it converges into fewer iterations only, or 2) the size of training data is less in comparison to the previous level. Fig. 6.6-(I) shows the cumulative time taken to train SVM model at each level. The entire training takes ≈ 30 minutes for training on the kddcup99 dataset.

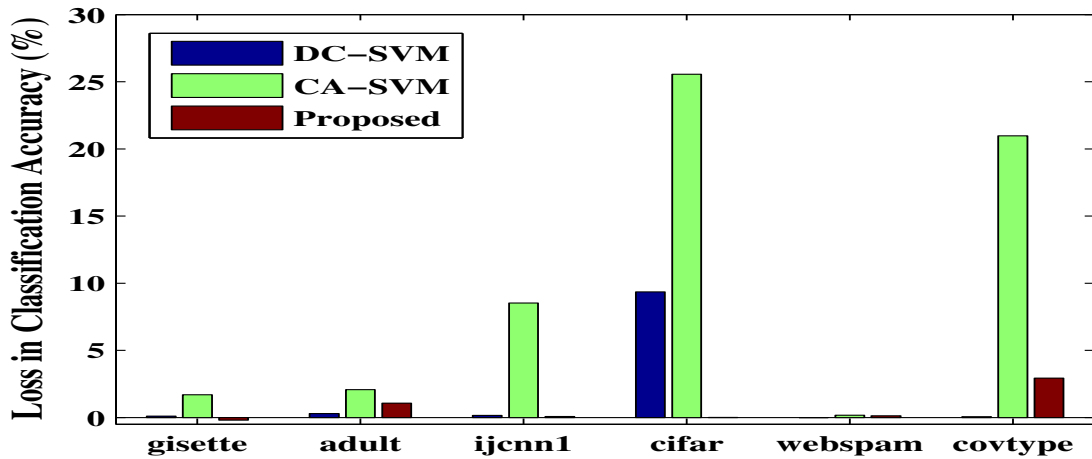


Figure 6.4: A comparison of the loss in classification accuracy (%) of DC-SVM, CP-SVM and proposed distributed SVM with respect to LIBSVM on publicly available datasets.

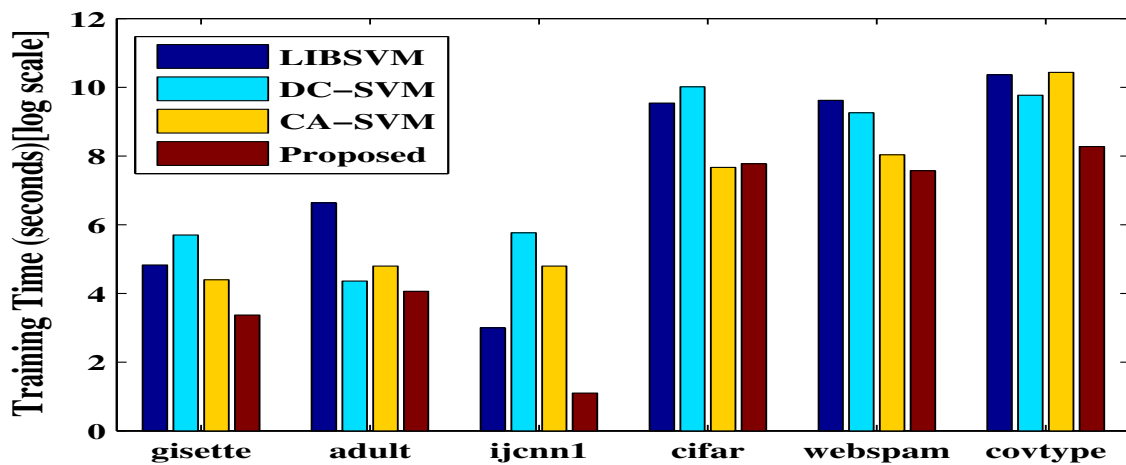


Figure 6.5: A comparison of the training time (seconds) of LIBSVM, DC-SVM, CP-SVM and proposed distributed SVM on publicly available datasets.

Further, the partitions generated using the proposed distribution preserving partitioning (DPP) approach are suitable for mini-batch training of DiP-SVM algorithm. In order to train SVM on a large dataset, mini-batches are generated using DPP. The initial SVM model is trained using first mini-batch. Then, the distance from the decision boundary

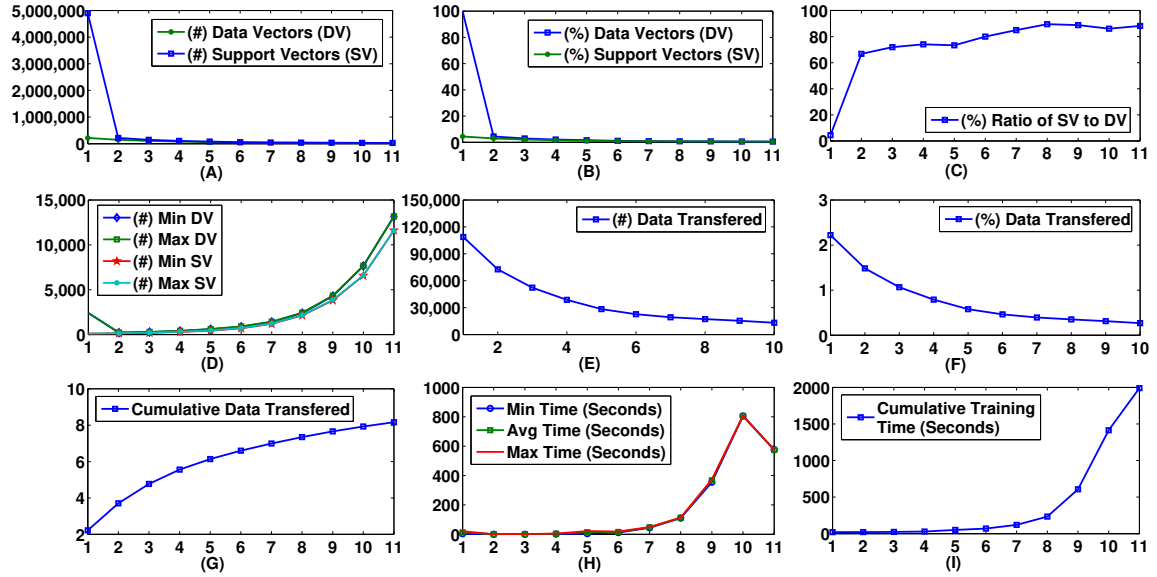


Figure 6.6: Computational and communication efficiency of the proposed approach on the dataset kddcup99 consisting of $\approx 5M$ records.

is calculated for each point in the second mini-batch using Equation (6.7) and only points closer to the decision boundary are selected and merged with the support vectors obtained in the previous step. For these new points, the Lagrangian multipliers are set to zero whereas for the existing support vectors, the old values are retained. Fig. 6.7 presents the results of classification performance for the mini-batch training of DiP-SVM, in which labels 1-10 on the x-axis correspond to 10 mini-batches, label 11 corresponds to distributed training of DiP-SVM and label 12 corresponds to the sequential SVM. It can be observed from the figure that using DPP, even the first mini-batch produces results which are quite close to the final classification results. Also, the accuracy increases as new points are introduced in incremental training. The use of previous Lagrangian multipliers and advanced elimination of irrelevant points before including them into the dataset for next SVM training also result in improved training time.

All the experiments performed with DiP-SVM confirm its suitability in comparison to existing approaches for distributed SVM training both in terms of scaling to large datasets and the performance of classification.

6.4 Summary

While distributed SVMs have generally proven to be much faster than sequential SVMs on large datasets, loss of classification performance and high communication overhead are

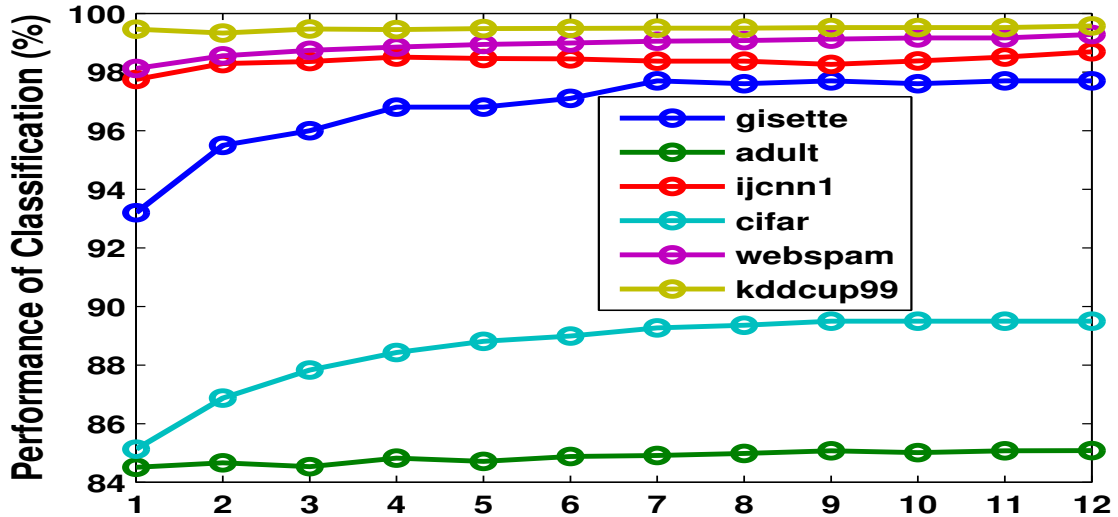


Figure 6.7: Performance of classification for mini-batch training of DiP-SVM.

still challenging issues. Through DiP-SVM, we aimed at solving both these issues by introducing a distribution preserving kernel SVM approach for a distributed environment. It was empirically shown that preserving the first and second order statistics of the entire dataset in each of the partitions helped in obtaining local support vectors which were shown to be in agreement with the global decision boundary. This also helped the proposed approach to achieve comparable classification performance to a sequential SVM while having the computational gains of a distributed approach on benchmark datasets. A comparison with state-of-the-art distributed approaches revealed that owing to the better distribution of data, DiP-SVM performs at-par or better on all the datasets tested. We also showed that the communication overhead between partitions was greatly reduced making it suitable for distributed environments.

Chapter 7

Distributed kernel SVM using subspace partitioning

As discussed in the previous chapter, data partitioning approaches hardly achieve a linear scale, suffer from high loss in accuracy, and generate high communication overhead in a distributed system due to the exchange of a large amount of data over the communication network. The existing tree-based subspace partitioning approaches like DTSVM [16] scale better than data partitioning approaches but suffer from high loss of accuracy due to the use of single attribute only for subspace partitioning. Another issue encountered in kernel SVM is that the large number of support vectors increase the prediction time. In this chapter, we propose *Projection-SVM*, a distributed implementation of kernel support vector machine for large datasets using subspace partitioning. In subspace partitioning, a decision tree is constructed on projection of data along the direction of maximum variance (i.e., dominant eigenvector) to obtain smaller partitions (i.e., subspaces) of the dataset. On each of these partitions, a kernel SVM is trained independently over a cluster thereby reducing the overall training time. Also, it results in reducing the prediction time significantly as the prediction is performed by the SVM classifier with less number of support vectors. We demonstrate the efficacy of the proposed approach on eight standard large datasets from various application domains where *Projection-SVM* is on an average 150 times faster than sequential SVM while maintaining the classification accuracy. The experimental results show the superiority of the *Projection-SVM* over the state-of-the-art approaches for distributed kernel SVMs, such as DCSVM, CASVM, and DTSVM.

The rest of the chapter is organized as follows. Section 2 discusses related work. The proposed distributed kernel SVM approach is discussed in section 3. Section 4 describes the experimental setup, evaluation method, and results. We conclude in section 5 with references at the end.

7.1 Subspace partitioning using decision tree and dominant eigenvector

The proposed approach works in two steps: training and testing. In the first step, a decision tree is constructed using training data. The master node partitions the entire dataset into smaller subsets. For partitioning the dataset, it computes the dominant eigenvector of the entire dataset using an iterative procedure. The entire dataset is projected on the dominant eigenvector. The spread of the projection is partitioned into B bins, where B is the maximum number of branches at any node in the decision tree. A child node is created for each non-empty bin and the data of the bins is assigned to their respective child nodes. Similarly, at each child node, a sub-tree is constructed, recursively. The decision tree partitions the data at each node along the direction of maximum variance in the data as described in next section.

In this work, we partition the entire data space into smaller subspaces. Let $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, 2, \dots, n\}$ be the entire dataset, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d -dimensional data point with class label $y_i \in \{-1, +1\}$. The direction of the maximum variance is given by the dominant eigenvector of the dataset D . In theory, we can use any eigen decomposition method like singular value decomposition (SVD) or eigenvalue decomposition for this purpose. However, we use iterative *power method* for computation of dominant eigenvector to achieve better computational and spatial efficiency. The computational complexity of SVD is $O(nd^2 + d^3)$, which is appropriate for computing all d eigenvectors. As our objective is to compute only dominant eigenvector, we use *power method* with time complexity $O(nd^2)$ for finding dominant eigenvector efficiently.

Definition 7.1 (Power method for dominant eigenvector). *The power method begins with an initial vector \mathbf{v}_0 which has a non-zero component in the direction of the dominant eigenvector. Then dominant eigenvector \mathbf{w} is given by following recurrence relation after t iterations:*

$$\mathbf{w} = \mathbf{v}_t = \frac{\Sigma \mathbf{v}_{t-1}}{\|\Sigma \mathbf{v}_{t-1}\|}, \quad (7.1)$$

where Σ is the covariance matrix of the dataset \mathbf{D} and is computed as

$$\Sigma = \text{cov}(\mathbf{D}) = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T, \quad (7.2)$$

where $\boldsymbol{\mu}$, ($\boldsymbol{\mu} = \frac{1}{n} \sum_i^n \mathbf{x}_i$) is the means of the dataset \mathbf{D} .

Equation (7.1) is solved iteratively, multiplying \mathbf{v}_{t-1} by Σ and then normalized. Initially, \mathbf{v}_0 is set to \mathbf{e} (i.e. the vector with all its values set to one) which guarantees non-zero

component in direction of dominant eigenvector. Once the dominant vector \mathbf{w} is computed, then the projection of a data point \mathbf{x}_i on the \mathbf{w} is given by

$$\hat{x}_i = \mathbf{w}^T \mathbf{x}_i. \quad (7.3)$$

Fig. 7.1 shows an example of the projection of all the points in \mathbf{D} on the dominant eigenvector. The entire spread of the projection is partitioned into B bins. According to these bins, the dataset \mathbf{D} is partitioned into B subsets. The following equation assigns the bin b for a data point \mathbf{x}_i :

$$b = \begin{cases} 1, & \text{if } r \leq 0, \\ \lceil r \rceil, & \text{otherwise,} \\ B, & \text{if } r > B. \end{cases} \quad (7.4)$$

$$r = \frac{\hat{x}_i - \hat{x}_{min}}{\hat{x}_{max} - \hat{x}_{min}} \times B, \quad (7.5)$$

where, \hat{x}_{min} and \hat{x}_{max} are the minimum and maximum values of the spread of projection of data points on the dominant eigenvector, respectively.

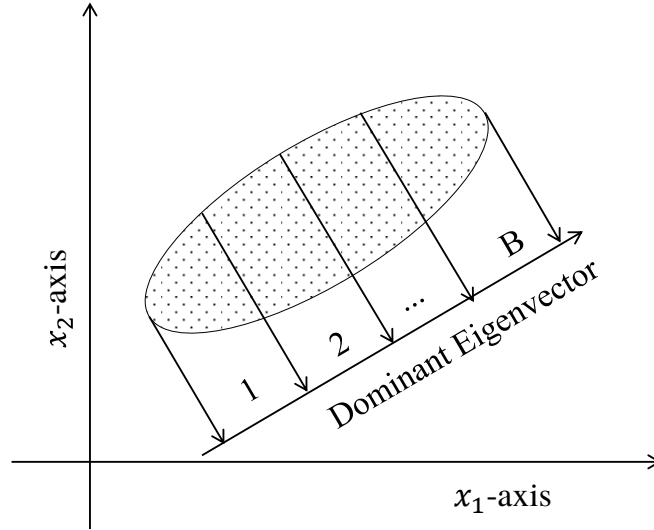


Figure 7.1: Proposed approach for data partitioning using decision tree along the direction of maximum variability.

Finally, the entire dataset \mathbf{D} is divided into B smaller datasets $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_B$. Each dataset \mathbf{D}_b contains n_b data points. If all the points at a node belong to the same class, then that node is declared as a leaf node labeled with the corresponding class. Otherwise, data space at a node is partitioned recursively until it reaches maximum level.

7.2 Training and prediction in distributed environment

The above partitioning method partitions the entire data space into subspaces. In the decision tree, for a leaf node, following two scenarios can occur while partitioning: i) all points in the subspace belong to the same class, or ii) subspace contains data points from both classes. As discussed earlier, node in case (i) is a leaf node with label same as data points. However, for case (ii), a kernel SVM model is trained using data points in that subspace. The smaller kernel SVM models on subspace data are independent and thus enable the proposed approach to be trained in a distributed system. Let $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_P$ represent the data in P subspaces which have data points from both the classes. Then the master node sends the data of these subspaces to P compute nodes. Each node with identifier p trains a SVM model on its data as follows:

$$\min_{\alpha_p} \frac{1}{2} \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \alpha_{p,i} \alpha_{p,j} y_{p,i} y_{p,j} K(\mathbf{x}_{p,i}, \mathbf{x}_{p,j}) - \sum_{i=1}^{n_p} \alpha_{p,i}, \quad (7.6)$$

where, α_p is the set of Lagrangian multipliers for the data of p^{th} subspace. The final SVM model \mathbf{SM}_p constitutes

$$\mathbf{SM}_p = \begin{cases} \mathbf{SV}_p = \mathbf{D}_p(\alpha_p > 0) \\ \alpha_p^* = \alpha_p(\alpha_p > 0), \end{cases} \quad (7.7)$$

where \mathbf{SV}_p are the support vectors and α_p^* are the corresponding non-zero Lagrangian multipliers. Once training is completed at a compute node, then it sends the trained SVM model \mathbf{SM}_p back to the master node. The master node creates a leaf node in the decision tree at the respective branch which contains the returned SVM model \mathbf{SM}_p . Algorithm 7.1 gives the pseudo-code of complete procedures of data partitioning and distributed SVM training for the proposed distributed SVM approach. After successful training, the final tree model looks like a sample tree shown in the Fig. 7.2. A non-leaf node contains 1) dominant eigenvector \mathbf{w} , and 2) \hat{x}_{min} and \hat{x}_{max} are the minimum and maximum values of the spread of projection of data points on the dominant eigenvector, respectively. A leaf node contains either a class label or an SVM model.

7.2.1 Prediction using proposed distributed SVM

In order to test an unknown data point on proposed distributed SVM, we traverse the decision tree from root to leaf; if leaf node has a class label, then this is the predicted label.

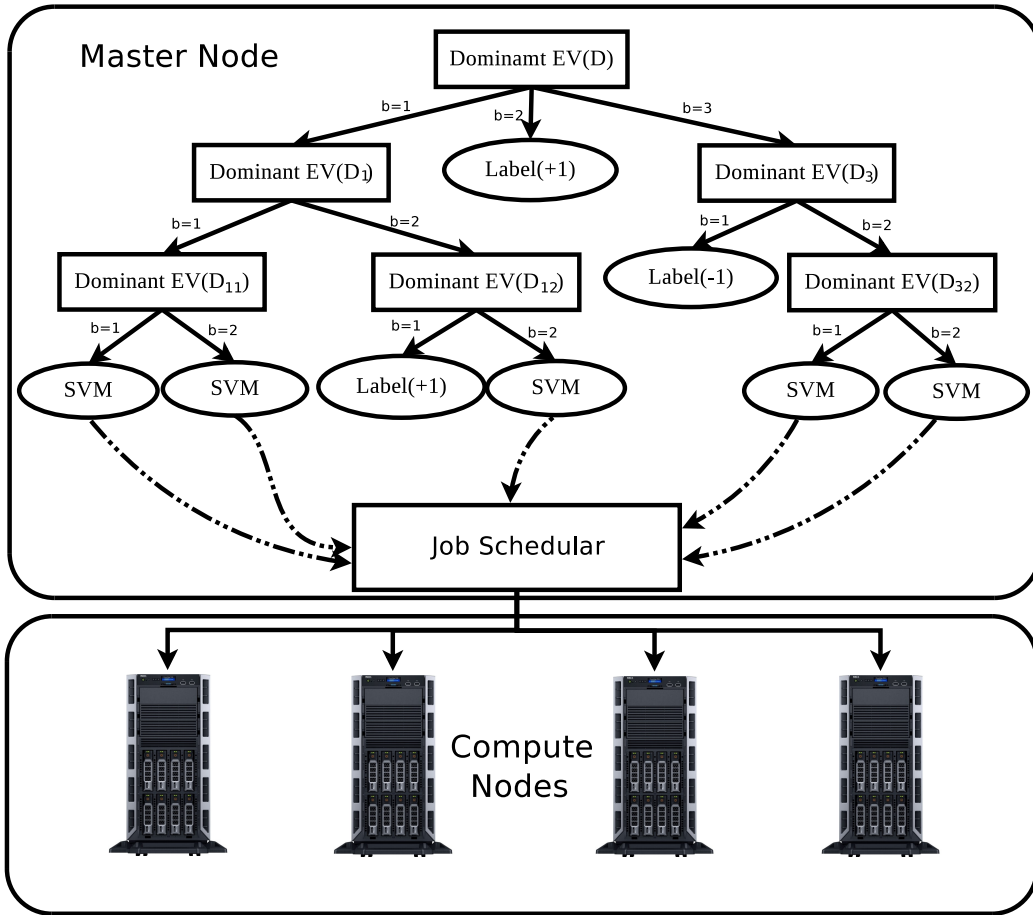


Figure 7.2: Block diagram of the Projection-SVM training over the cluster. Master node contains a sample tree model. The job scheduler evenly distribute the task of training SVMs to compute nodes

If a leaf node has an SVM model, then the classification label is predicted using that SVM model. Let $x \in \mathbb{R}^d$ be a test data point, and $tree$ be the trained model. When traversing the $tree$, at any node, there are three possibilities:

1. *Internal Node*: If the current node is an internal node, then based on the parameters $(\mathbf{w}, \hat{x}_{min}, \hat{x}_{max})$, it computes the bin index using Equations (7.3),(7.4) & (7.5) and selects the corresponding branch. According to the selected branch, it visits a child node, and this procedure is continued until it reaches a leaf node.
2. *Leaf Node with Class Label*: If the current node is a leaf node with a class label, then it assigns the class label of the leaf node as the predicted class of the test point x and the procedure is terminated.
3. *Leaf Node with SVM Model*: If the current node is a leaf node with a trained SVM model SM , then it that SVM model to predict the class of the test point x .

Algorithm 7.1 Training of proposed distributed SVM

Input: \mathbf{D} : $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$, n : #data points in \mathbf{D} , d : #dimensions, B : #branches (max) at each internal node, h : Maximum height of the tree, ϵ : tolerance for evaluating dominant eigenvector, P : #partitions and also #node processors

Output: *tree*: final tree model for prediction

train_SVM(\mathbf{D} , h)

```
1: if  $\forall y_i \in \mathbf{D}, y_i = 1$  then
2:   return Leaf(1);
3: else if  $\forall y_i \in \mathbf{D}, y_i = -1$  then
4:   return Leaf(-1);
5: else if  $h = 0 \parallel n < \text{min\_size}$  then
6:   return Leaf(svm_train( $\mathbf{D}$ ));
7: else
8:    $\boldsymbol{\mu} = \frac{1}{n} \sum_i \mathbf{x}_i$ ;  $\boldsymbol{\Sigma} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$ ;
9:    $t \leftarrow 0$ ;  $\mathbf{v}_0 \leftarrow \mathbf{e}$ ;
10:  while  $\|\mathbf{v}_t - \mathbf{v}_{t-1}\| > \epsilon$  do
11:     $\mathbf{v}_t \leftarrow \boldsymbol{\Sigma} \mathbf{v}_{t-1}$ ;  $t = t + 1$ ;
12:  end while
13:   $\mathbf{w} \leftarrow \mathbf{v}_t$ ;
14:   $\mathbf{D}_b \leftarrow \phi, b = 1, 2, \dots, B$ 
15:   $\hat{x}_i \leftarrow \mathbf{w}^T \mathbf{x}_i, i = 1, 2, \dots, n$ ;
16:   $\mathbf{x}_{min} \leftarrow \min(\hat{x}_i)$ ;  $\mathbf{x}_{max} \leftarrow \max(\hat{x}_i)$ 
17:  tree  $\leftarrow$  Node( $\mathbf{w}, \hat{x}_{min}, \hat{x}_{max}$ )
18:  for  $i = 1, 2, \dots, n$  do
19:     $b \leftarrow \min\left(\max\left(\left\lceil \frac{\hat{x}_i - \hat{x}_{min}}{\hat{x}_{max} - \hat{x}_{min}} \times B \right\rceil, 1\right), B\right)$ ;
20:     $\mathbf{D}_b \leftarrow \mathbf{D}_b \cup (\mathbf{x}_i, y_i)$ ;
21:  end for
22:  for  $b = 1, 2, \dots, B$  {In parallel} do
23:    tree.child $_b \leftarrow$  train_SVM( $\mathbf{D}_b, h - 1$ );
24:  end for
25:  return tree;
26: end if
```

Algorithm 7.2 gives the pseudo-code of complete procedure of prediction using proposed distributed SVM.

7.2.2 Time complexity analysis

The partitioning time at a node includes the time taken in computing the dominant eigenvector, projecting data points on the dominant eigenvector, and partitioning data to each branch, i.e.

Algorithm 7.2 Prediction using Proposed Distributed SVM

Input: $\mathbf{x} \in \mathbb{R}^d$:unlabeled data point, d : #dimensions, B : #branches (max) at each internal node, $tree$: trained tree model

Output: y : predicted label for \mathbf{x}

$predict_SVM(tree, \mathbf{x})$

```
1: if  $isLabel(tree) = true$  then
2:    $y \leftarrow tree.ClassLabel$ ;
3: else if  $isSVM(tree) = true$  then
4:    $svm\_model \leftarrow tree.svm\_model$ ;
5:    $y \leftarrow svm\_model(x)$ ;
6: else
7:    $\mathbf{w} \leftarrow tree.\mathbf{w}$ ;
8:    $\mathbf{x}_{min} \leftarrow tree.\mathbf{x}_{min}$ ;  $\mathbf{x}_{max} \leftarrow tree.\mathbf{x}_{max}$ ;
9:    $\hat{x} \leftarrow \mathbf{w}^T \mathbf{x}$ 
10:   $b \leftarrow \min\left(\max\left(\left\lceil \frac{\hat{x} - \hat{x}_{min}}{\hat{x}_{max} - \hat{x}_{min}} \times B \right\rceil, 1\right), B\right)$ ;
11:   $y \leftarrow predict\_SVM(tree.child_b, \mathbf{x})$ ;
12: end if
13: return  $y$ ;
```

$$T_{partition} = O(nd^2 + nd + n) \approx O(nd^2). \quad (7.8)$$

For sequential decision tree construction, the total time is

$$T_{tree} = \sum_h \sum_{b=1}^{B^h} \frac{n}{B^h} d^2 = O(nd^2 h). \quad (7.9)$$

However, for parallel construction of decision tree, the total time is

$$T_{tree} = \sum_h \frac{n}{B^h} d^2 = nd^2 \sum_h \frac{1}{B^h} \approx O(nd^2), \quad (7.10)$$

since, $1 \leq \sum_h \frac{1}{B^h} \leq 2$. The *best case* for the proposed approach occurs when the data on all children nodes after partitioning belong to one class only as shown in Fig. 7.3-(B). Thus in best case, the training time includes only partitioning time as no SVM is trained. The *average case* is when the decision tree makes the balanced partitions, and each class contains data points from both the classes. Then for height h and maximum number of branches B , the scaling factor (SF) is

$$SF = \frac{n^3}{\left(\frac{n}{B^h}\right)^3} = B^{3h}. \quad (7.11)$$

The *worst case* corresponds to highly imbalanced partitioning and the scaling factor in *worst case* depends on n_{max} , the size of biggest partition among all partitions at height h , i.e.

$$SF = \left(\frac{n}{n_{max}} \right)^3. \quad (7.12)$$

7.3 Experiments and results

The proposed distributed SVM is implemented in C++ using *libsvm* [15], *armadillo* [99], *openmp* and *openmpi* which are the de facto standards for scientific computing over the high-performance cluster (HPC). The master node is responsible for data partitioning and maintaining the tree model. The number of partitions (P) are determined by $P = B^h$ where B is the number of branches and h is the height of the tree. The values of B and h depend on the size of each dataset.

7.3.1 Sketches of correctness

Case-1: Each class as a single Gaussian distribution

If both classes are well separable, and projection of points on the dominant eigenvector is producing two non-overlapping spreads for both classes as shown in Fig. 7.3(A), then it splits the data into disjoint partitions containing data points from either class. In this case, no SVM is trained as the data on each child node belongs to the same class as shown in Fig. 7.3(B) for $B = 2$. The overall classification is similar to Fisher’s linear discriminant analysis (LDA). However, if the projection of the two separable classes is slightly overlapping as shown in Fig. 7.3(C), then for each overlapping bins, it trains SVM as shown in Fig. 7.3(D). In such cases, the overlapping bins contain data points of different classes which are relatively close to each other. Now, the SVM model is trained with data points which are more likely to be the support vectors resulting in a faster training of SVM. If the projection of the points in both the classes on the dominant eigenvector produces two completely overlapping spreads as shown in Fig. 7.3(E) & Fig. 7.3(G), then each bin will contain points from both the classes. If classes are separable, then at some height, a decision tree can discriminate the points of the two classes. For example, the tree shown in Fig. 7.3(F) does it at $h = 2$ for the case shown in Fig. 7.3(E). However, if classes are non-separable as shown in Fig. 7.3(G), then it needs to train an SVM model for each bin as shown in Fig. 7.3(H).

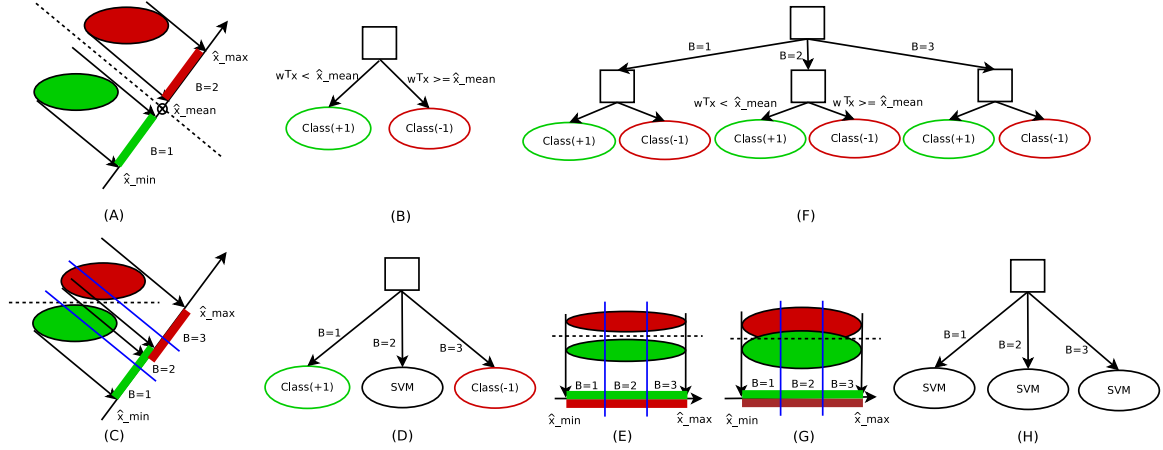


Figure 7.3: Illustration of the working of proposed distributed SVM for well separable classes.

Case-2: Each class as a mixture of Gaussian distributions

Let's consider a complex case where each class is composed of several distributions spread non-uniformly in the space as shown in Fig. 7.4(A). In the first step, the proposed SVM splits the complex set of data points into smaller subsets which are relatively less complex as shown in Fig. 7.4(C). Here, the classification is relatively easier, faster, and may lead to good performance because it focuses on the local data points only. However, it is also possible that local boundaries are less regularized in comparison to global decision boundary. Fig. 7.4(B)&(F)-(G) show the local decision boundaries corresponding to each subset. The experiments on randomly generated data points show 2% – 10% improvement in the classification performance in comparison to LIBSVM.

First, we conducted experiments on the synthetic data to show that the proposed method works well for the data having complex distributions. For this, we generated a mixture of K -Gaussian distributions, where $K = 10, 20, 30, 40, 50, 60$. The labels to each Gaussian distribution are assigned $+1$ or -1 randomly. One such data which is a mixture of 50 Gaussian distributions is shown in Fig. 7.5(A). In the data, both the classes are spread over the entire space with significant overlap of positive and negative examples. Fig. 7.5(B) showed the comparisons of the classification performance for sequential SVM and proposed method on the synthetic datasets. The proposed approach performs similar to sequential SVM for the low values of K , but for the high values of K , it achieved much better performance than sequential SVM. Because finding a single separating hyperplane using SVM for such a complex data distribution is very hard. However, proposed distributed SVM converts a complex large problem into multiple simple smaller problems and then solves them independently.

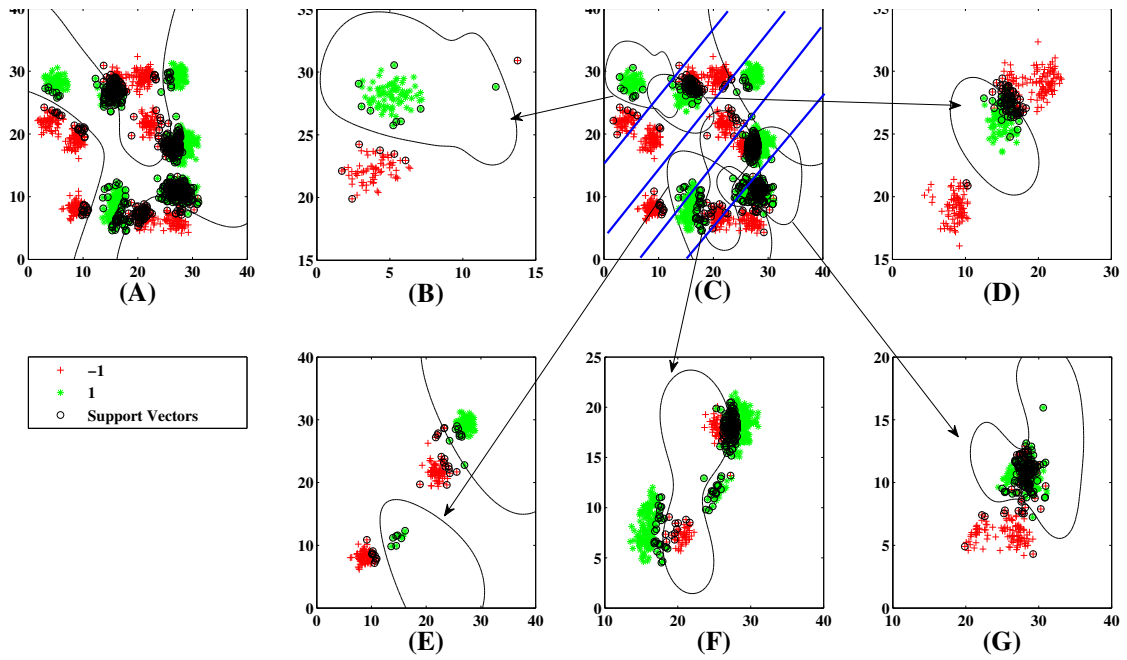


Figure 7.4: A comparison of the sequential SVM and the proposed distributed SVM on a sample 2D-data which is a mixture of the 20-Gaussian distributions.

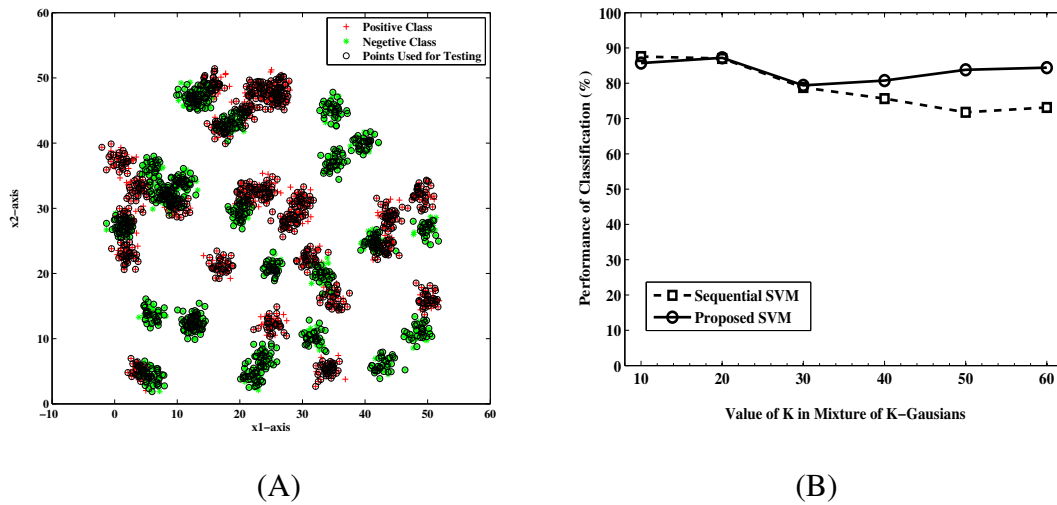


Figure 7.5: A comparison of the classification performance (%) for sequential SVM and proposed distributed SVM. (A) Sample dataset. (B) Performance at various mixture of K -Gaussian distributions. Class labels are assigned randomly. $K = 10, 20, 30, 40, 50, 60$, Number of data points $n = 2400$.

7.3.2 Comparison with state-of-the-art methods

In order to evaluate the performance of the proposed distributed SVM, experiments are conducted on various large scale high dimensional datasets from different application do-

mains. The datasets used for benchmarking are publicly available at [1, 2]. The datasets used are same as in Table 6.2 of the previous chapter. The details of the hyper-parameters (C, γ) for SVM along with values of B and h are given in Table 7.2.

Table 7.1: Performance of classification (%) of proposed distributed SVM and comparison with LIBSVM, DC-SVM, CA-SVM and DT-SVM.

Method	LIBSVM		DC-SVM		DT-SVM		Proposed		Change	
	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Scale
gisette	97.70	125	97.60	299	97.60	355	97.50	43	-0.20	3×
adult	85.08	761	84.79	78	84.79	45	84.46	9	-0.62	85×
ijcnn1	98.69	20	98.53	318	94.33	27	98.82	1	+0.13	20×
cifar	89.50	13.9K	80.15	22.3K	75.82	540	87.11	193	-2.39	72×
webspam	99.28	15.1K	99.28	10.5K	NA	NA	98.81	28	-0.47	538×
covtype	96.01	31.8K	95.95	17.5K	NA	NA	95.77	119	-0.24	267×
kddcup99	99.57	37.7K	99.49	23.3K	NA	NA	99.02	40	-0.55	942×
mnist8m	99.91	562K	99.91	NA	NA	NA	99.77	6786	-0.14	166×

NA - Particular method is unable to calculate

Table 7.2: Various evaluation metrics for effectiveness & efficiency of the proposed distributed SVM.

Dataset	C	γ	B	h	Precision (%)	Recall (%)	F-measure (%)	Kappa Index (-1,+1)	Partition Time (Seconds)	Train Time (Seconds)	Test Time (Seconds)
gisette	1	2e-4	2	1	97.03	98.00	97.51	0.9500	0.94	42.18	6.86
adult	32	2 ⁻⁷	2	2	87.00	93.65	90.20	0.5292	0.02	3.98	7.00
ijcnn1	32	2	2	5	94.69	92.80	93.74	0.9309	0.07	0.61	0.35
cifar	8	2 ⁻²²	2	4	88.16	90.70	89.41	0.7295	10.30	183.19	111.27
webspam	8	32	2	10	99.45	98.59	99.01	0.9751	5.49	22.08	12.23
covtype	32	32	2	10	96.23	95.53	95.88	0.9153	3.71	115.20	4.37
kddcup99	256	0.5	2	10	97.13	98.43	97.78	0.9715	23.95	15.64	1.52
mnist8m	1	2 ⁻²¹	2	10	99.74	99.80	99.77	0.9954	1184.1	5602	123.56

The loss of classification accuracy with respect to sequential SVM as well as the existing distributed SVMs is used for comparison of performance. Table 7.1 shows the performance of the classification and training time on all datasets. The details of various evaluation metrics used for evaluation of the proposed approach is given in Table 7.2. The proposed distributed SVM approach reduces the loss in the classification accuracy, and the results are approximately equal to the results of the sequential SVM. On all the datasets considered for evaluation, the proposed approach achieves the least drop in the classification accuracy among existing approach (i.e. DCSVM [40], CASVM [133], and

DTSVM [16]) as compared to sequential SVM as shown in Fig. 7.6. One possible reason for this reduction in the loss of classification accuracy is that the proposed approach finds the decision boundary in the smaller subspaces only, which may help in better classification within the subspaces. The decision tree splits the large complex problem into smaller simple problems which are then solved with more precision.

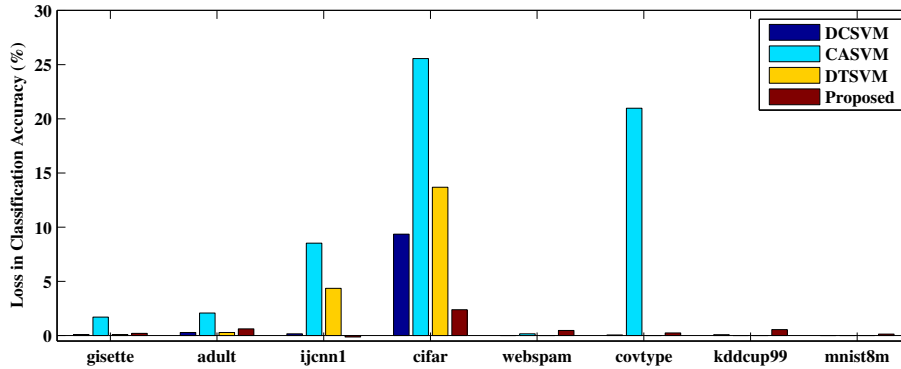


Figure 7.6: A comparison of the loss in classification accuracy (%) of DCSVM, CASVM, DTSVM, and proposed distributed SVM with respect to LIBSVM on various datasets.

The proposed SVM approach reduces the training time as well as testing time significantly as shown in Fig. 7.7 and Fig. 7.8, respectively. The reduction in training time can be attributed to the distributed training of smaller independent SVMs. The proposed approach uses decision tree for partitioning of the dataset which is computationally less expensive as compared to kernel-clustering approach for very large datasets. Use of the dominant eigenvector efficiently divides the entire space along the direction of maximum variance. This partitioning leads to the reduction in the variance of the data points in subspace. As there is no conquer step, so it further reduces the training time. This approach also reduces the communication overhead significantly as it does not send the data from one level to another after training as required in [40]. The proposed approach needs to communicate twice, once to send data from the master node to worker nodes and later to receive model parameters back from worker nodes to master node.

Finally, the tree model predicts the label at leaf nodes using leaf label or trained SVM. There are three types of nodes in tree model: 1) Internal node which decides to which subspace the test data point belongs. 2) Leaf node with a class label which directly predicts the classification label for the test data point without any computation. 3) Leaf node with SVM model which uses the trained SVM model to predict classification label. As these SVMs are trained on smaller sub-datasets which generally will contain less number of support vectors with respect to global SVM.

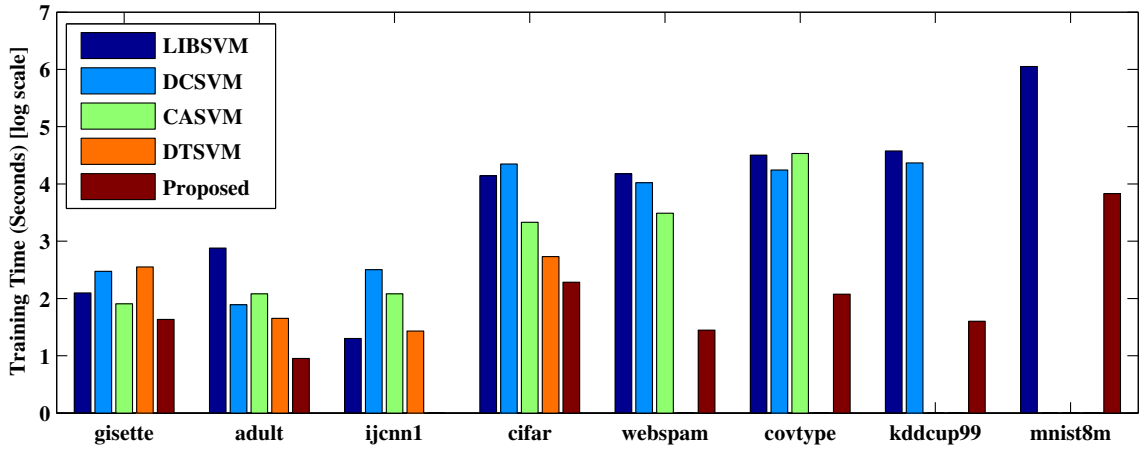


Figure 7.7: A comparison of the training time (seconds) for LIBSVM, DCSVM, CASVM, DTSVM, and proposed distributed SVM on various datasets.

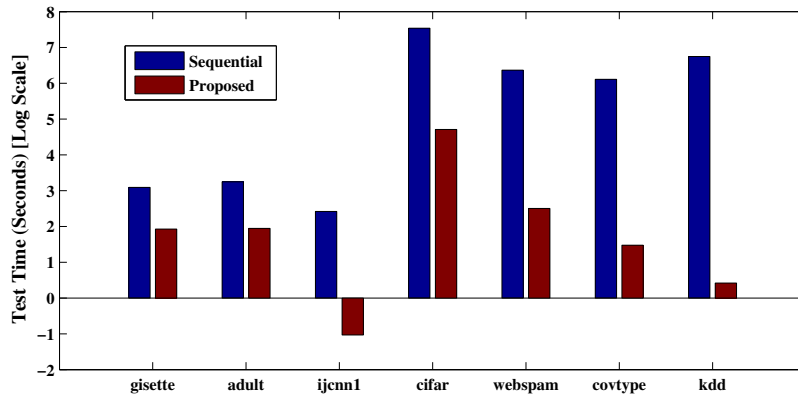


Figure 7.8: Comparison of the test time of sequential SVM and the proposed approach

7.4 Summary

In this work, a distributed SVM for big data using decision tree and dominant eigenvector is proposed. This distributed SVM approach trains the model faster and requires less time in prediction for new data points. The use of dominant eigenvector and decision tree for partitioning of the dataset is also computationally less expensive with a complexity of $O(nd^2)$ in comparison to kernel k -means approach with a complexity of $O(n^2d)$ as proposed in [40] [133]. The proposed approach also achieves good classification performance with a small change in accuracy. The experimental results on eight standard datasets confirm that the proposed approach is on an average ≈ 150 times faster than sequential SVM and $\approx 10 - 50$ times faster than other existing distributed SVM approaches, namely, DCSVM, CA-SVM, and DTSVM while sustaining the accuracy.

Chapter 8

Edge computing-based framework for city-scale traffic incident detection

In the previous chapters, we proposed scalable and distributed methods for feature representation and modeling techniques. However, deploying visual computing methods for real-time detection of traffic rule violations and incidents in a city-wide surveillance network is a challenging task because it needs to perform computationally complex analytics on the live video streams of large number of cameras, concurrently. The desired objectives for such a system include high accuracy of the detection task and real-time inference. However, the existing robust models for object detection contain large number of parameters and need significant computational resources for the real-time inference. Another issue is the centralized infrastructure or cloud computing based frameworks for deploying such systems suffer from high network latency due to congestion on the communication links in transferring live video streams from cameras at site to the central server making them inefficient for real-time deployment. In this chapter, we propose an efficient framework using edge computing to deploy our two visual computing applications for active traffic monitoring in smart city. To overcome the network latency issue, we placed the detector module in the vicinity of the capturing devices on a embedded hardware called edge-node. All the edge-nodes send their detected alerts to a central alert database where the end users access these alerts through a web interface. The first application is the automatic detection of bike-riders driving without helmet and second application is the automatic detection of the accident incidents.

The rest of the chapter is organized as follows. Section 8.1 presents the proposed edge-computing framework. We present the proposed approach for helmetless motorcyclists detection in Section 8.2 along with empirical results. Section 8.3 presents the proposed approach for accident detection along with empirical results. We summarize in Section 8.4.

8.1 Edge computing framework for traffic monitoring

In a city-scale surveillance scenario, the central computing infrastructures are unable to perform in real-time due to large network delay from video sensor to the central computing server. For accurate and real-time detection of traffic violations and incidents in such a scenario, we propose an efficient framework using edge computing for deploying large-scale visual computing applications which reduces the latency and the communication overhead in the camera network as shown in Fig. 8.1.

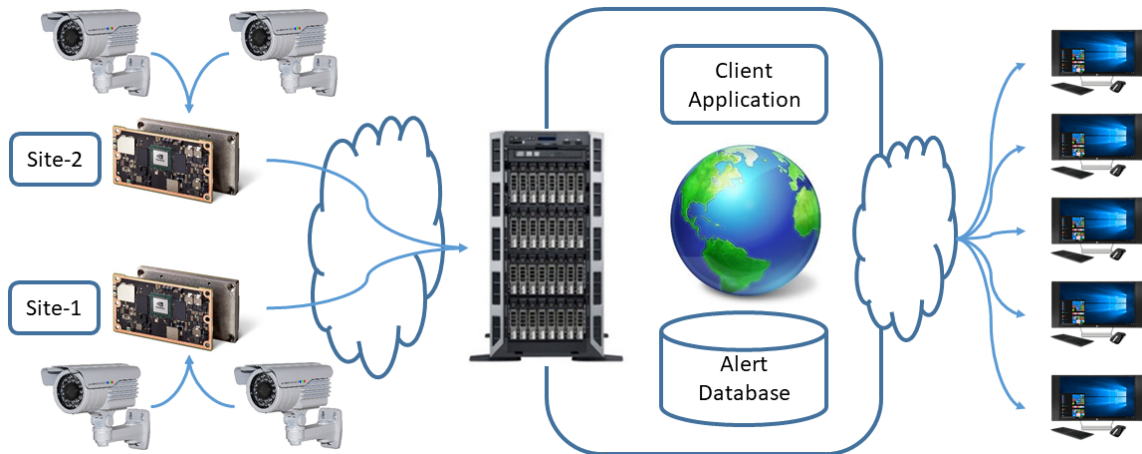


Figure 8.1: The proposed edge computing-based framework for traffic monitoring

The entire system architecture consists of three parts, namely, compute node, central servers, and client interface. The compute nodes are the embedded devices placed in the close vicinity of the cameras at the sites. The detector modules are placed on the computing nodes where they process the live video footage from the cameras in real-time without any delay and send their detected alert to a central alert database at the central server. As the central servers have better computational resources, we further evaluate the received alerts using a more reliable model. For example, in helmetless motorcyclists detection, the tiny-YOLO is used at the detector module at compute nodes with a relatively low confidence score for the detection of violators and full YOLO with a high confidence score is used at the server for re-evaluation. Similarly, for the accident detection, at the detector module, the accident score is generated based on the anomaly score only with a low threshold, while on the central server, it also combines the intersection of trajectories. This re-evaluation helps in the reduction of false alarms. The end users can access the detected alerts from the central alert database through a web interface. We propose two applications of surveillance video analysis which are discussed in details in the next subsections.

8.2 Real-time detection of motorcyclists without helmet

As discussed in the chapter 2 existing methods for the detection of motorcyclists without helmet are not been able to accurately identify motorcyclists without helmets under challenging conditions such as occlusion, illumination, poor quality of video, varying weather conditions, etc. Some of the reasons for the poor performance of existing approaches are: (i) the use of not so efficient handcrafted features for object classification, (ii) the consideration of irrelevant objects against the objective for the detection of motorcyclists without helmet, and (iv) most of the existing methods are computationally complex and thus not suitable to be used in real-time. The deep networks have gained much attention with state-of-the-art results in complex recognition tasks such as image classification [57], object recognition [48], tracking [45, 51], detection, and segmentation [94, 126] due to their ability to learn discriminatory features directly from raw data without resorting to manual tweaking. The challenges in the existing recognition systems and the recent advancement in deep learning motivate us to design an efficient framework for the detection of motorcyclists driving without helmet in real-time that can handle wide variation in viewpoints and environmental conditions. Specifically, we design a robust and compact method for detection of moving motorcyclists in real-time using convolutional neural network (CNN) as shown in Fig. 8.2. The entire framework consists of four steps: (i) detection of motorcycles using a CNN based object detector, (ii) localization of the upper body part of the person riding the motorcycle, (iii) prediction using HH-Net a CNN classifier trained for binary classification of head and helmets, and (iv) temporal consolidation of the alert to generate more reliable alerts. The details of the methods used in these steps are discussed in the following subsections.



Figure 8.2: Block diagram of proposed framework for the detection of motorcyclists without helmet.

8.2.1 Detection of motorcyclist using CNN based object detector

The first step of the proposed framework is the detection of motorcyclists in the incoming live video stream. This problem falls into the broad category of object detection. The existing methods for object detection such as YOLO and SSD are showing much-improved

detection accuracy but unable to perform in real-time on embedded cards with limited computational resources because of the large number of convolutional operations. In this work, we derive a compact CNN model from tiny-YOLO to detect the motorcyclists in the incoming live video frames from a CCTV camera. As our objective is to detect motorcyclist only, we restrict YOLO to a single class of motorcycle only. In this way, we get the bounding boxes of all the bikes present in a frame. In order to accelerate the convolutional operations, we ternarize the weights in the convolutional layers of the pre-trained network into $\{-1,0,+1\}$. The ternarization of the weights accelerates the detection because it reduces the number of multiplication operations in convolutional layers. The ternarization of the weights results into the loss of accuracy in comparison to the original network. However, we recover this loss by fine-tuning the last fully connected layer and the softmax layer.

8.2.2 Localization of the rider's head

The output of the previous step is a set of bounding boxes $mbox$ for each detection of a motorcycle, we again search for its rider (i.e., a person) and if detected then extend its height slightly up words to guarantee the complete coverage of the rider's head. The upper one-third part of this extended bounding box is the final location for the rider's head and the output will be a bounding box $bbox(x, y, w, h)$, where (x, y) are the coordinates of the center of the bounding box and (w, h) are the width and the height of the bounding box where all values are the ratio with respect to the size of full input image.

8.2.3 Classification of head and helmet using CNN

The output of the previous step is a $bbox(x, y, w, h)$ locating the region of image consisting of the motorcyclist's upper body part as shown in Figure 8.3.

The next task is to ensure whether the detected motorcyclist is wearing the helmet or not. If the resulted faces are clearly visible then one can easily detect violators by applying methods such as Viola-Jones [115], HoG [26], SIFT [70], LBP [35], DeepFace [112], VGG_face CNN descriptor [85], Deep Head Pose [79], etc. to classify them into face vs. non-face categories. However, in spite of a good resolution camera, the faces are not clearly visible due to the size of their appearance while covering the entire road. This makes the task of detection of violators non-trivial and thus all the techniques as mentioned earlier fail to address this task. The deep learning model like the convolutional neural network can be applied to extract hidden information relevant for discriminating the heads from the helmets. As mentioned earlier, most relevant pre-trained deep model VGG16 is also not able to solve this task due to unclear face appearance and tiny images of the head/helmet.



Figure 8.3: The sample images of the located motorcycle riders with and without a helmet of various style in different viewpoints.

The other models have a large number of parameters and thus unable to give a real-time performance and require a significant number of training samples. Also, the two large CNNs in Vishnu *et al.* [116] increased their prediction time and may lead to overfitting as they are trained from the scratch. To address above mentioned issues, we design a simple, fast, and robust convolutional neural network classifier which can be trained from relatively small number of training examples. As we know that the first few layers of the large CNNs extract generic features and can be used for learning variety of tasks thereafter. Thus, we leverage this fact and design a tiny network on top of the activation filters received from the output of an intermediate layer of the detector network. Fig. 8.4 shows the architecture of the proposed CNN model called HH-Net used for the classification of motorcyclist with helmet and without a helmet.

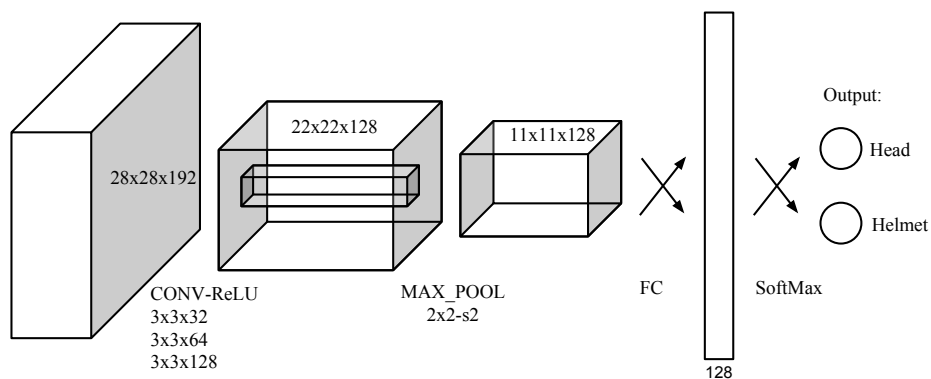


Figure 8.4: Architecture of the proposed network HH-Net for head vs. helmet classification.

The input to HH-Net is a tensor of size $28 \times 28 \times 192$ which is cropped from the output of

the second convolutional layer of the detector network. Let $bb\text{ox}(x, y, w, h)$ be the bounding box located the head of a motorcyclist as computed in the previous step and $O_l(\text{width} \times \text{height} \times \text{channels})$ is the output of the l^{th} convolutional layer, then the input I to the HH-Net is determined as¹

$$I = O_l(\text{width} * (x - \frac{w}{2}) : \text{width} * (x + \frac{w}{2}), \quad (8.1)$$

$$\text{height} * (y - \frac{h}{2}) : \text{height} * (y + \frac{h}{2}), 1 : \text{channels}), \quad (8.2)$$

$$I = \text{resize}(I, [28 \times 28 \times \text{channels}]). \quad (8.3)$$

HH-Net consists of three convolutional layers with rectified linear unit (ReLU) as activation, followed by one 2×2 max-polling layer, and one fully connected layer of 128 neurons. Finally, the network uses a softmax layer with two classes. The resulting activation of each layer is shown in Fig. 8.5. It can be observed from the figure that the model gives high activation values corresponds to the helmet while low activation values corresponds to the head. Also, there is an increase in the intensities of the activation values for the deeper layers.

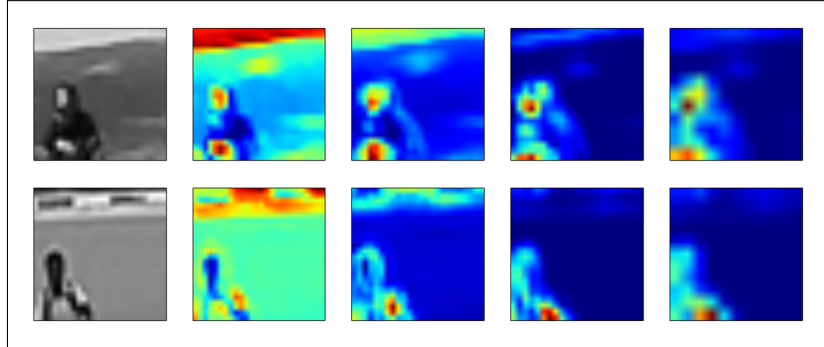


Figure 8.5: The activations produced by the various layers of the proposed CNN classifier after training for helmet (top) and head (bottom).

8.2.4 Temporal consolidation of the alerts:

From the previous phase, we obtain decision on each individual frame whether it contains the violator(s) (motorcyclists without helmet) or not. As the proposed approach is applied on continuous video stream, there are multiple alarms raised for a single violator in multiple frames. However, the correlation between continuous frames is completely neglected in the detector module. Thus, we consolidated the alerts generated from the detector module over

¹the notation $i:j$ represent the range for selection similar to MATLAB notation

the continuous frames in order to generate less number of alerts with increased reliability i.e. reducing the number of false alerts. Let y_i be the label for i^{th} frame which is either +1 (i.e. at-least one violator is detected) or 0 (i.e. no violator detected). Then for the n frames, the violation alarm is triggered as

$$Alarm = \begin{cases} True, & \text{if } \frac{1}{n} \sum_{i=1}^n y_i > T_f, \\ False, & \text{otherwise} \end{cases} \quad (8.4)$$

where the threshold T_f is determined empirically. In our case, the value of $T_f = 0.6$ and $n = 5$ are used.

8.2.5 Experiments and results

The experiments are conducted on a machine running Ubuntu 16.04 Xenial Xerus having specifications Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz×48 processor, 128GB RAM with NVIDIA Corporation GK110GL [Tesla K20c]×2 GPUs. The programs for helmet detection are written in C & CUDA with the help of the various libraries such as *OpenCV* – 2.4.13 for image processing and vision tasks. For training deep models, we use *darknet* [90], an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.

Datasets used

The performance of the proposed approach is evaluated on two video datasets *IITH_Helmet_1* and *IITH_Helmet_2* containing sparse traffic and dense traffic, respectively. Both the datasets are collected by us because there is no public dataset available till the date to the best of our knowledge. The datasets are made public for future use by the research community². The brief descriptions for both the datasets are as follows.

IITH_Helmet_1: This dataset is collected from the surveillance network at Indian Institute of Technology Hyderabad, India (IITH) campus. It is a two-hour surveillance video data collected at 30 frames per second. Fig. 8.6 presents sample frames from the collected dataset. We have used the first one hour of the video for training and the remaining for testing purpose. The training video contains 42 motorcycles, 13 cars, and 40 humans. Whereas, the testing video contains 63 motorcycles, 25 cars, and 66 humans.

IITH_Helmet_2: This second dataset is acquired from the CCTV surveillance network of Hyderabad city in India. It is a 1.5 hour video collected at 25 frames per second. The

²<https://sites.google.com/site/dineshsinghindian/dataset/>

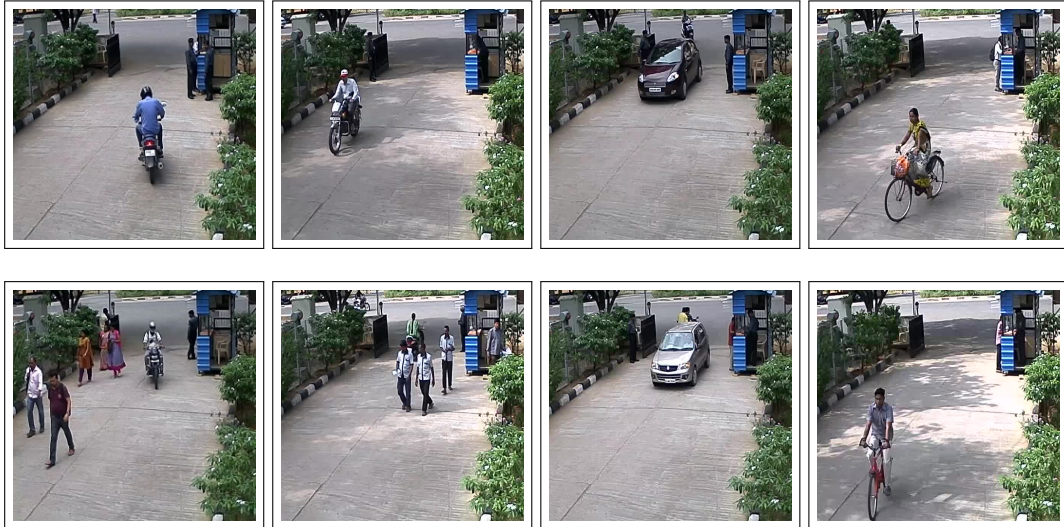


Figure 8.6: Sample frames from *IITH_Helmet_1* dataset showing the various difficulties.

sample frames from this dataset are presented in Fig. 8.7. The first half an hour of the video is used for training the model and the remaining for testing purpose. The training video contains 1261 motorcyclists and 4960 non-motorcyclists. Whereas, the testing video contains 2312 motorcycles, and 9112 non-motorcyclists.



Figure 8.7: Sample frames from *IITH_Helmet_2* dataset showing the various difficulties.

Evaluation of classification accuracy

As explained in the section 8.2.1, the use of YOLO is proved to be more robust and reliable for accurate detection of the motorcyclists irrespective of variations in their appearance and environmental condition. The proposed model successfully detects all the motorcycles in

the case of sparse traffic as the case in *IITH_Helmet_1* dataset while using a low threshold of 0.3 on the confidence score without a single false detection. While the performance of motorcycle detection using GMM as used in [25, 116] is $\approx 98\%$ on *IITH_Helmet_1* dataset. Also on the dense traffic dataset *IITH_Helmet_2*, it did not raise any false alarms even on a very low threshold of 0.2. However, due to high occlusion of various vehicles as well as the size of their appearance, it missed few motorcyclist. This problem occurs since the videos collected in the dataset are unconstrained and the cameras are not placed explicitly for such a task and thus can be solved easily by putting the camera in an appropriate place.

However, the classification of the head vs. helmet is challenging as shown in Fig. 8.9 (A) & (C) depict the 2D visualization of the spread of the extracted train and test samples from *IITH_Helmet_1* dataset, respectively. Here, the pattern classes correspond to head, and helmets are overlapping each other showing that the patterns share high inter-class similarity along with intra-class dissimilarities which make the classification task more complex. Thus, the performance of the previously used methods *GMM + HoG* [25] achieved only 93.80% on *IITH_Helmet_1* dataset and while score a low performance of 57.78% on *IITH_Helmet_2* dataset as shown in Table 8.1. However, the performance is improved slightly using *CNN* [116] which achieved 98.63% on *IITH_Helmet_1* dataset and 87.11% on *IITH_Helmet_2* dataset. However, the proposed deep CNN model as explained in section 8.2.3 addresses this problem more precisely in comparison to previously used methods. Here, we present an extensive evaluation of the proposed approach.

The filters for various samples of both the classes as shown in the Fig. 8.8 reveals the success of the proposed model. The learned hidden deep structures are of discriminating in nature as well as self-explanatory. The feature maps across the various samples of class helmet contain a consistently high activation values for the pixels corresponds to a helmet in the input images. However, for the another class (i.e. head), this structure is clearly different. As can be clearly shown from the two figures that instead of the head region it produces high activation for the region corresponds to the solder.

The final observation is the transformation in the distribution of the train and test datasets from the first input image to the final deep representation (i.e. the output of the last fully-connected layer of the trained CNN model). Fig. 8.9 show the scatter plots of the *IITH_Helmet_1*.

In both the figures the sub-figures (A) & (C) show the original (input raw pixels) distributions for train and test sets, respectively. Similarly, sub-figures (B) & (D) show the distributions of the final deep representation for train and test sets, respectively. It can be observed from the scatter plots that the proposed model learns the distribution of the two

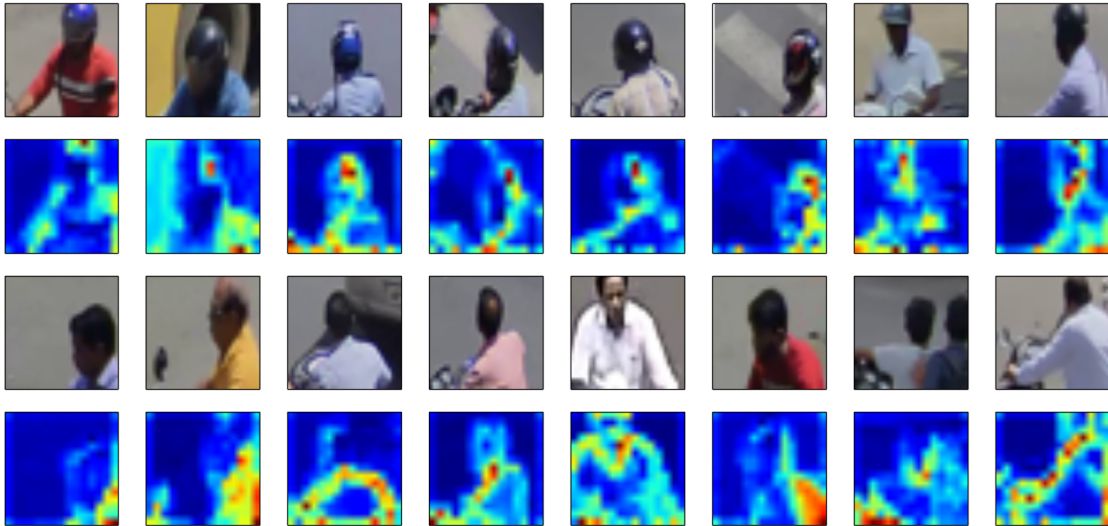
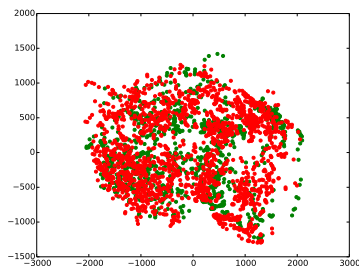
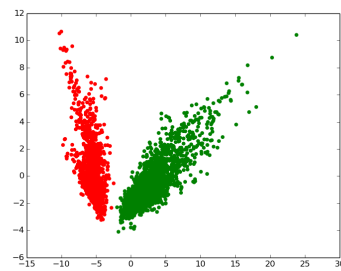


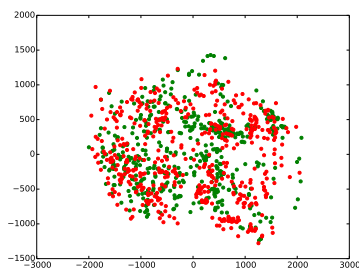
Figure 8.8: Sample images and their respective activation maps for the two classes, namely, helmet (followers) and head (violators). [Best viewed in color]



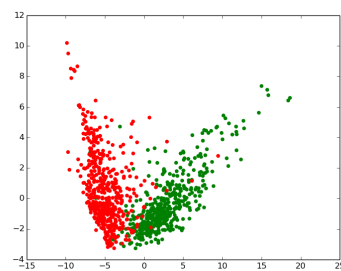
(A) Original distribution of train dataset.



(B) Final distribution of train dataset.



(C) Original distribution of test dataset.



(D) Final distribution of test dataset.

Figure 8.9: Scatter plots *IITH_Helmet_1* dataset showing distributions of the two classes, namely, helmet (green dots) and head (red dots) in train and test datasets before and after the training. [Best viewed in color]

classes and transforms them into space where they are easy to classify, and the learned weight of the model also follows the same kind of distribution. Thus it can be concluded

that the model transforms a complicated and hard to classify distribution of the two classes into a distribution where the points from two categories are easy to identify. This transformation results in a high classification accuracy in comparisons to other approached.

The comparisons with the various recently proposed approaches on both the datasets are presented in the Table 8.1. The proposed system outperforms the existing methods with a margin of 4.90%, 0.07% on *IITH_Helmet_1* and 36.38, 6.95% on *IITH_Helmet_2* datasets, respectively.

Table 8.1: Comparison of classification performance of ‘helmet’ vs. ‘without helmet’

Method	IITH_Helmet_1 (%)	IITH_Helmet_2 (%)
GMM+HOG+SVM [25]	93.80	57.78
GMM+CNN+CNN [116]	98.63	87.11
Proposed Approach:	98.70	94.16

Evaluation of space & time requirement

The proposed approach for the detection of the motorcyclist without a helmet can process a real-time stream at a speed of 22 *fps* on GPU (Tesla K20c) and take a total of 943*MB* space in device memory. The space requirement on the device for weights of detector CNN model is 889*MB* and an additional memory requirement of 54*MB* for input, intermediate, and output variables. Similarly, the space requirement on the device for weights of HH-Net classifier model is 8*MB* and an additional memory requirement of 1*MB* for input, intermediate, and output variables. Thus the proposed framework is highly scalable for processing multiple real-time cameras streams. To reduce the device memory usage, we shared the weights of the models of size 952*MB* across all threads on a GPU card, and each new thread requires an additional memory of 55*MB* for storing input, intermediate, and output variables. Table 8.2 shows the space and processing speed when processing multiple streams on a single GPU card.

Table 8.2: Space & time requirements of the proposed models.

#Streams	Size of Model		Processing Speed		
	Detector (MB)	Classifier (MB)	Detector (MS)	Classifier (MS)	Combined (<i>fps</i>)
1	943	9	40	5	22
2	997	10	59	7	15
3	1051	11	81	10	11
4	1105	12	148	18	6

8.3 Deep spatio-temporal representation for detection of road accident

The course of accident can be divided into three stages: pre-collision, collision, and post-collision. Each stage gives us a significant amount of information but also involves several difficulties as discussed below.

Pre-collision: The pre-collision case is the most vital information to explain an accident scenario. Also, this information may become a good evidence for crime scene investigation. The pre-collision situation is a clear violation of traffic rules by any/both the vehicles, which include violation of traffic lane, violation of signals at intersections, violation of speed limit at congested roads, abrupt motion on the road, etc. Finally, we can say that pre-collision stage is an unusual activity and thus can be easily detected by applying anomaly [64, 129] detection methods based on the various parameters, such as speed, trajectories, position, etc.

Collision: The collisions are essential to accident detection, but it is very complicated to detect and cannot be directly detectable by any general purpose computer vision technique. One way to detect a collision is to identify the joints of the trajectories of the vehicles over spatiotemporal dimensions. However, the major challenge is the discrimination between collision and occlusion. For this we use the trajectories over space-time interest points [59] and improved dense trajectories [121, 122].

Post-collision: As stated above that the collision and occlusions are hard to classify and may lead to false alarms. These false alarms further can be refined by considering the post-collision scene. The two most common post-collision scenes include: 1) Fallen objects at the collision point: As we stated that the intersection of the trajectories of two vehicles might be a collision or an occlusion. However, after the intersection, if both the trajectories are continued, and no abrupt or zig-zag motion resulted. Then, the intersection is merely an occlusion, not a collision. However, if some abrupt motion or discontinued trajectories have occurred, then the possibility of a collision is high. Measure the time for which the object remains static. 2) Crowd attention towards the collision point: The last and final stage of the accident is the crowded road or pedestrians running towards the collision point.

As shown in Fig. 8.10 the proposed framework for automatic detection of accident incident composed of abnormality detection using the deep representation of spatiotemporal video volumes (STVVs) and collision detection using intersection points of trajectories. The anomaly detection works in two steps, the first step is the automatic training of the deep features and the second step is to determine the outlier score for unknown incidents.

The separately stacked denoising autoencoder (SDAE) trained over STVVs from the previously seen normal traffic video one for each representation is used to generate the deep representation for the STVVs from the unseen traffic video. The possibility of an accident is determined based on the reconstruction error and the likelihood of the deep representations for which outlier score is generated using one-class SVM. All these individual scores (a.k.a. local score) are then fused to compute the final decision to declare an incident as an accident. We present the detail description of these steps in the following subsections.

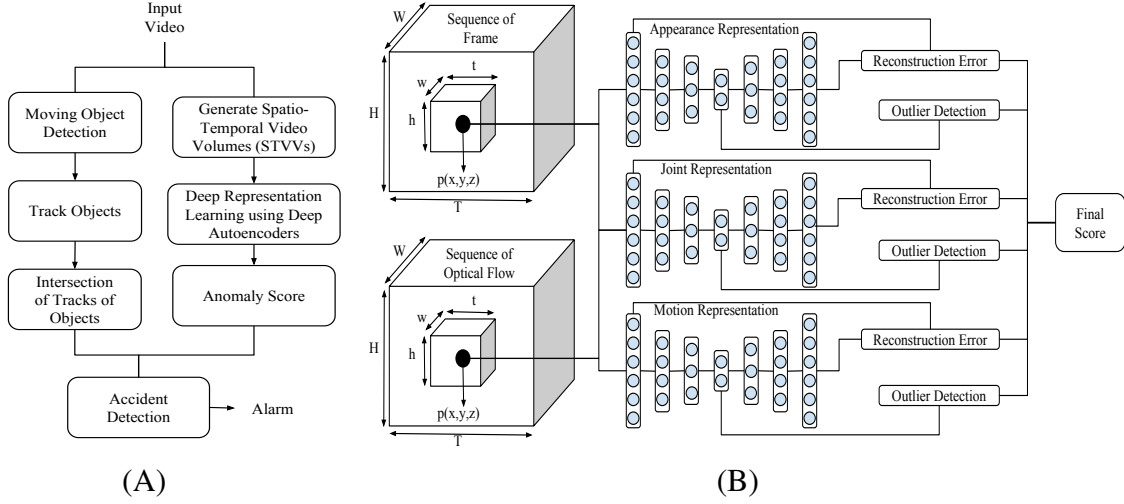


Figure 8.10: The architecture of the proposed framework for accident detection. (A) Overview of the framework. It consists of two streams, one for the generation of the collision score using the trajectories of the moving vehicles and the other one for generation of abnormality score using deep representation. (B) A detailed diagram of abnormality detection using deep stacked autoencoder on three modalities, namely, appearance, motion, and joint representation.

8.3.1 Spatio-temporal volume generation

In order to localize the accident incident, we divided the entire video into several smaller size volumes called spatio-temporal video volumes (STVVs) similar to [62], with different scales in both space and time as well as across the modalities such as appearance, motion, and joint representations. Fig. 8.11 shows a STVV at a pixel $p(x, y, z)$ in a 3D video volume.

Definition 8.1 (STVV). *Lets, $\mathbf{v} \in \mathbb{R}^{W \times H \times T}$ given continuous video sequence where point $\mathbf{v}(x, y, z) \in \mathbb{R}$ gives the intensity of the pixel (x, y, z) for all $x \in [0, W], y \in [0, H]$, and $z \in [0, T]$. Here, $\mathbf{v}(0 : W, 0 : H, z)$ represents the z^{th} frame. The $\mathbf{v}(x - \frac{w-1}{2} : x + \frac{w-1}{2}, y - \frac{h-1}{2} : y + \frac{h-1}{2}, z - \frac{t-1}{2} : z + \frac{t-1}{2})$ is a space-time video volume (STVV) of size*

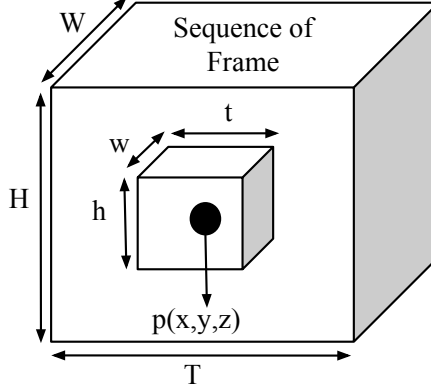


Figure 8.11: The generation of the spatio-temporal video volumes (STVVs). The STVVs are the pixels in the immediate vicinity of a point $p(x, y, z)$ covered by a 3D sliding window of size (w, h, t) .

$w \times h \times t$ around the pixel (x, y, z) . These STVVs are then normalized and vectorized into a vector $\mathbf{x} \in \mathbb{R}^{wht}$. Finally, we have a datasets $\mathbf{X} = \{\mathbf{x}_i\}, i = 1, 2, \dots, n$ where n is total number of such STVVs.

8.3.2 Stacked denoising autoencoder (SDAE)

A denoising autoencoder (DAE) is a simple one-hidden-layer neural network with unsupervised learning using backpropagation algorithm. The objective of a DAE is to transform given partially corrupted samples into a compressed representation to learn latent patterns by minimizing the amount of distortion in reconstructed samples. The denoising autoencoder consists of two processes:

1. *Encoding*: The encoder takes a nonlinear mapping denoted as $f_e(\mathbf{x}_i|\mathbf{W}, \mathbf{b})$ from the partially corrupted input to a hidden representation. For a given corrupted input $\tilde{\mathbf{x}}_i$, a compressed hidden layer representation \mathbf{h}_i can be obtained as below:

$$\mathbf{h}_i = f_e(\tilde{\mathbf{x}}_i|\mathbf{W}, \mathbf{b}) = \sigma(\mathbf{W}\tilde{\mathbf{x}}_i + \mathbf{b}). \quad (8.5)$$

Typically, corrupted inputs are obtained by drawing samples from a conditional distribution $p(\mathbf{x}|\tilde{\mathbf{x}})$, for example the Gaussian white noise or salt-pepper noise.

2. *Decoding*: The decoder is used to map the hidden representation back to a reconstruction representation through a similar transformation $f_d(\mathbf{h}_i|\mathbf{W}', \mathbf{b}')$. For a given hidden representation \mathbf{h}_i , a reconstructed representation $\hat{\mathbf{x}}$ is computed as below:

$$\hat{\mathbf{x}}_i = f_d(\mathbf{h}_i|\mathbf{W}', \mathbf{b}') = s(\mathbf{W}'\mathbf{h}_i + \mathbf{b}'). \quad (8.6)$$

Here, $\langle \mathbf{W}, \mathbf{b} \rangle$, and $\langle \mathbf{W}', \mathbf{b}' \rangle$ denote the weights and the bias terms of the encoder and decoder, respectively. The $\sigma(\cdot)$ and $s(\cdot)$ are activation functions. Typically, the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ is used as the activation function. The network can learn a more stable and robust representations of the input using this encoder/decoder structure. An stacked denoising autoencoder (SDAE) is a cascade of several denoising autoencoders (DAEs) as shown in Fig. 8.12.

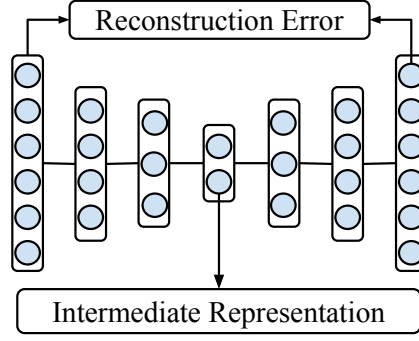


Figure 8.12: The network topology of the proposed stacked autoencoder used to model the baseline for the normal traffic. The network consists three decoder layers followed by three decoder layers. The reconstruction error is the Euclidean distance of the input and output layers. The output of the middle layers is the latent intermediate representation.

The parameters $(\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}')$ are learned for a given training set $X = \{\mathbf{x}_i\}_{i=1}^n$ by minimizing the following regularized least square optimization problem:

$$\min_{\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}'} \sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2), \quad (8.7)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The first term $\sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$ is the average reconstruction error, while the second term $(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2)$ is the weight penalty for regularization. The importance of these two terms is balanced by parameter λ . Typically, sparsity constraints are also imposed on the output of the hidden units to discover meaningful representations from the data.

8.3.3 Detection of intersection points in trajectories

First, we detect moving objects by subtracting background images, and then the moving objects are tracked. In an STVV, if two tracks are intersecting each other then it represents either a collision or an occlusion as shown in Fig. 8.13. In the presented frame, we found that the trajectories of the bike and car intersect each other. Also, the trajectories of several

other vehicles touch each other several time. Since the trajectories continue in the subsequent frames, they are simply considered as the occlusions, not collisions. But, there is no further progress in the trajectories of the bike and car so this is considered as a collision. The collision scores \mathcal{C} of a STVV is the simple count of such points in that STVV.



Figure 8.13: The intersection of two trajectories during an accident. The trajectories of the motorcycle and car intersect each other also there is no further progress in the trajectories of the motorcycle and car, thus considered as a collision.

8.3.4 Accident score generation

We use one-class SVM to generate the outlier score γ of intermediate representation \mathbf{h} for a given STVV. One-class SVM requires only one class data and fits an outer boundary around this data. In this case, we use only STVVs from normal traffic for building model. The outlier score γ for a given \mathbf{h} is computed from one-class SVM as below:

$$\gamma = f(\mathbf{h}) = \sum_{i=1}^m \alpha_i K(\mathbf{h}_i, \mathbf{h}) - \rho, \quad (8.8)$$

where, $\{\mathbf{h}_1, \dots, \mathbf{h}_m\}$ are the m support vectors with their respective Lagrange multipliers α_i , ρ is the threshold value. If the weighted density of a feature vector with support vectors is above a threshold ρ then feature vector is classified as normal and abnormal otherwise. The values of these parameters are computed by solving below dual problem for n training points $\{\mathbf{h}_1, \dots, \mathbf{h}_n\}$:

$$\max_{\alpha} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(\mathbf{h}_i, \mathbf{h}_j) \text{ s.t. } \sum_{i=1}^n \alpha_i = 1 \text{ and } 0 \leq \alpha_i \leq \frac{1}{vn}, \quad (8.9)$$

where $v \in (0, 1)$ control the penalty imposed on the nonzero slack variables.

For a STVV \mathbf{x} the reconstruction error ξ is computed as below

$$\xi = \|\mathbf{x} - \hat{\mathbf{x}}\|_{\mathcal{F}}^2. \quad (8.10)$$

where, $\hat{\mathbf{x}}$ is the reconstruction of the STVV \mathbf{x} and $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. The high reconstruction error shows that the particular STVV is less likely drawn from the previously seen patches and thus increases the likelihood that it may belong to an accident scene.

For each STVV \mathbf{v} , we extract three representation: (i) appearance representation \mathbf{x}^A based on still frames, (ii) motion representation \mathbf{x}^M based on optical flow, and (iii) joint representation \mathbf{x}^J by early fusion (concatenation) of both appearance and motion representation. For each representation, we extract deep representation using stacked denoising auto-encoder and compute anomaly scores $\gamma^A, \gamma^M, \gamma^J$ using Equation (8.8) and reconstruction errors ξ^A, ξ^M, ξ^J using Equation (8.10). Also, the collision score \mathcal{C} is computed as discussed in previous section. Finally, we use post-fusion of scores to get single final score. We consider the linear combination to keep less number of parameters and reduced computation in comparison to a non-linear combination. The computation of non-linear transformation leads to a large number of parameters and increased computation time. The final accident score s is given as below:

$$s = \beta_1 \gamma^A + \beta_2 \gamma^M + \beta_3 \gamma^J + \beta_4 \xi^A + \beta_5 \xi^M + \beta_6 \xi^J + \beta_7 \mathcal{C}, \quad (8.11)$$

where, $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6$, and, β_7 are free parameters to control false alarms. The final decision of whether \mathbf{v} corresponds to an accident or not is taken based on the threshold s_T which is given as.

$$Decision = \begin{cases} Accident, & \text{for } s > s_T \\ Normal, & \text{otherwise.} \end{cases} \quad (8.12)$$

The parameters in Equation (8.11) are computed using linear regression on a small amount of manually labeled data as follows. Let, \mathbf{X} be the set of STVVs with corresponding label set \mathbf{y} , where $y_i = \{-1, +1\}$, and $\mathbf{S} = [\gamma^A, \gamma^J, \gamma^M, \xi^A, \xi^J, \xi^M, \mathcal{C}]$ be the set of corresponding scores. Then the parameters set $\beta = [\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7]^T$ is given by

$$\beta = (\mathbf{S}^T \mathbf{S} + \lambda \mathbf{I})^{-1} \mathbf{S}^T \mathbf{y}, \quad (8.13)$$

where $\lambda = 10^{-6}$ is the regularization parameter. While the best performing threshold s_T is decided empirically.

8.3.5 Experiments and results

Since there is no public video dataset available for accident detection, we collected own dataset from the CCTV surveillance network of Hyderabad City in India. Video clips collected from City surveillance network are captured at 30 frames per second. Fig. 8.14 presents samples from the collected dataset. Each video clip starts few minutes before the incident of an accident and contains several minutes after the incident. First few minutes of video which contains normal situation are used for training the model and remaining for testing. There are total 127138 normal frames, and 863 frames contain partial or full accidents labeled manually. For training 94720 normal frames are used. For testing we used 33280 frames 32417 normal and 863 accident frames. The dataset is made public for the research community for further comparison ³.



Figure 8.14: Sample frames from the video dataset used to evaluate the performance of proposed approach. The dataset contains videos of accidents during the various environmental conditions such as high sunlight, night, early morning as well as from different cameras and view angles.

The STVVs are generated at various scales in both space and time. For experiments, we generate STVVs of spatial scale of 11×11 , 13×13 , and 15×15 pixels. For each spatial scale, we generate three temporal scales of 3, 5, and 7 frames. Thus finally we generate

³https://sites.google.com/site/dineshsinghindian/iith_accident-dataset

9 STVVs at each spatiotemporal point. The denoising stacked autoencoder projects high dimensional data onto a lower dimension where it forms a manifold as shown in Fig. 8.15. The projections of the unknown patterns which are drawn from the similar class patterns from which the SDAE is trained are very close to the other points in the manifold and very far otherwise. Thus the one-class-SVM with RBF kernel generates the score based on how likely an unknown pattern is drawn from the normal traffic patterns. A high deviation score confirms that the particular pattern belongs to some unusual/unseen/abnormal/outlier event, which further increases the possibility of an accident as an accident is also a rare event.

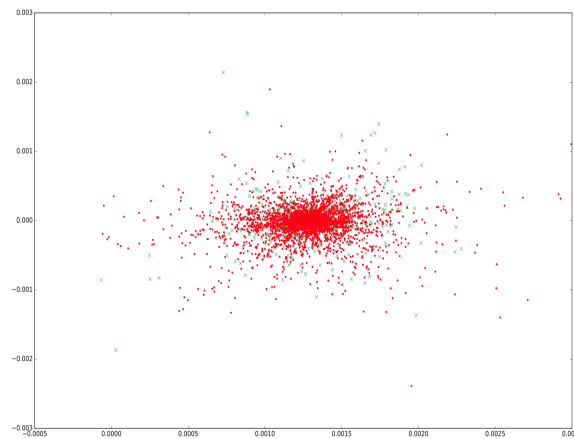


Figure 8.15: The 2-D visualization of the distribution of the STVVs data generated from a sample video. The STVVs during normal and accident are shown using the green cross and red dot, respectively. [Best viewed in color]

Since the proposed method is an unsupervised method and the final classification is based on a threshold. Changing the threshold results into a change in the performance. Thus to find the optimal threshold we computed various performance scores on hundred different threshold values. The trade-off between the sensitivity (i.e., true positive rate) and the specificity (i.e., false positive rate) is shown via ROC curve. In a ROC curve, the red dotted line shows the random prediction (50%) line, and the solid black line is the equal error rate (EER) line. We analyzed the discrimination ability for all three representations for reconstruction error at various layers of SAE as well as the outlier score of the corresponding intermediate representations using one-class-SVM. Fig. 8.16 illustrates the ROC curve for the experiments conducted for various thresholds (s_T) for reconstruction error at different layers of the stacked denoising autoencoder. The area under the curve (AUC) increases with an increase in the number of stacked autoencoders. But after stack of two autoencoders, there is a very slight increment, and thus the performance (AUC) for them is

not changing significantly than the performance of layer-2. The final performance (AUC) of the accident detection based on the reconstruction error alone are 76.54%, 51.57%, and 76.28 for appearance, motion, and joint representations, respectively.

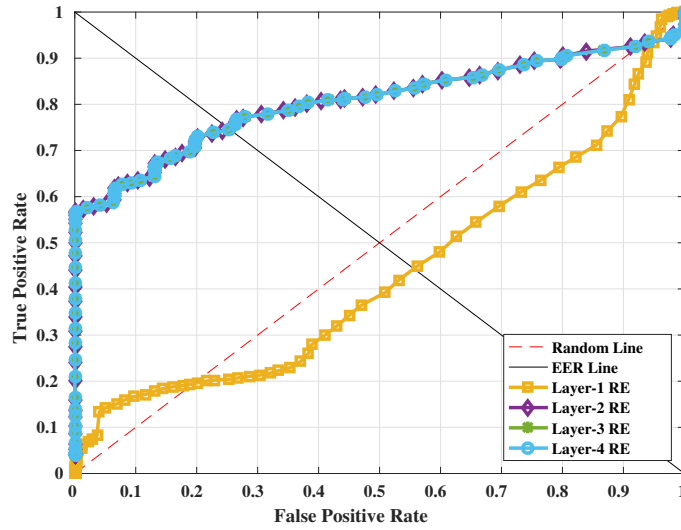


Figure 8.16: ROC curve for accident detection using reconstruction error at various Layers.

A similar phenomenon is also seen for outlier score using one-class SVM on the intermediate representation. The performance increases with an increase in the number of autoencoders. The final performance (AUC) of the accident detection based on the intermediate representation using one-class SVM is 77.54%, 62.87%, and 74.21% for appearance, motion, and joint representations, respectively. Fig. 8.17 illustrate the ROC curve for the experiments conducted for various threshold s_T for reconstruction error, one-class-SVM, and their combination for the stack of three auto-encoders. The AUC for the combined is more than both reconstruction error and one-class-SVM alone.

However, the performance increases when we combine both the scores. Fig. 8.18 illustrates the ROC curve for the experiments conducted for various threshold s_T for reconstruction error, one-class-SVM, and their combination for the appearance, motion, and their joints representations. The AUC using different modalities and methods is listed in Table 8.3. The AUC for the combined is more than both reconstruction error and one-class-SVM alone. The final performance of the accident detection based on the combined representation is 77.60%, 62.91%, and 81.06% for appearance, motion, and joint representations, respectively.

Fig. 8.19 show the examples of the detected accident regions using the anomaly scores from various samples from the collected dataset. The red region is the predicted accident regions using a single score either by appearance, motion, or intersection of tracks while

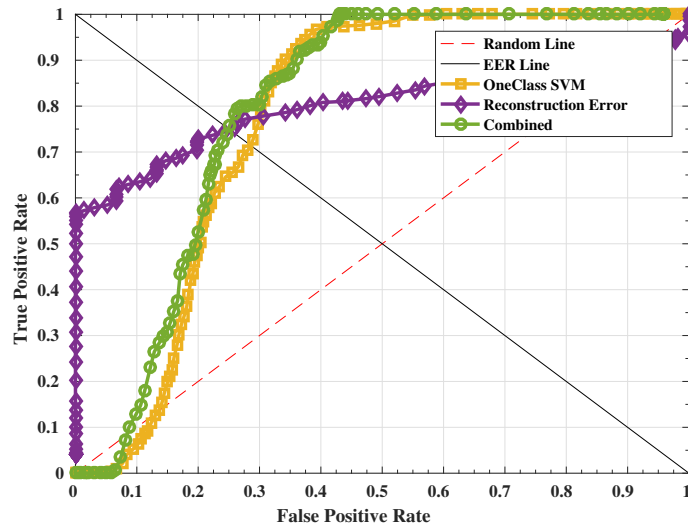


Figure 8.17: ROC curve for accident detection using reconstruction error (RE), one class SVM, and their combination.

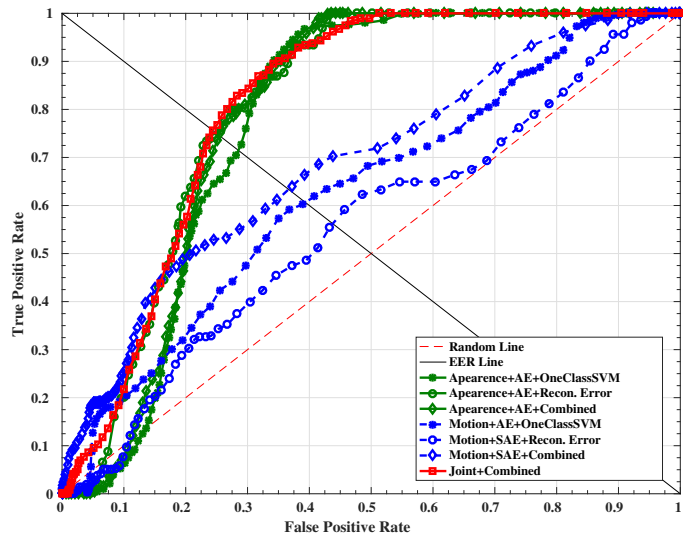


Figure 8.18: ROC curve for accident detection using reconstruction error, one class SVM, and their combination for the appearance, motion, and their joints representations.

Table 8.3: Area under curve (AUC) for various modalities and methods

Representation	Recon. Error	OneClassSVM	Combined
Appearance Representation	0.7654	0.7754	0.7760
Motion Representation	0.5157	0.6287	0.6291
Joint Representation	0.7628	0.7421	0.8106

the green box shows the region decided using final score. Here, the accident detection rate using anomaly score is very high as it accurately detects almost all the regions which are

declared accident manually (i.e. ground-truth). However, it also detects false accident in several regions which are actually normal which leads to high false alarms rate. Although, with the help of the complementary information from the trajectories of the moving vehicle these alarms are further refined.



Figure 8.19: Accident detected using proposed approach for different videos. The red region is the predicted accident regions using a single score either by appearance, motion, or intersection of tracks while the green box shows the region decided using final score.

Finally, the proposed method can localize the accident events as we are using STVVs instead of entire frame or full video clip. Also, on the collected video dataset of real accidents which contains accidents in various lighting conditions as day, high sun and night it is giving on average 0.775 detection rate at equal error rate (EER) of 0.225.

Comparison with the existing methods

Instead of a highly desirable task, there is a limited work done in this domain due to unavailability of the public benchmark dataset. Since the existing methods use small private collection of datasets and do not make them public, comparing them may not be fair at this stage. But still, we listed the performance achieved by the existing methods on individual datasets. ARRS [55] achieve 63% detection rate and 6% false alarms. RTADS [44] achieve 92% detection rate and 0.77% false alarms. The method of Sadek *et al.* [97] shows a recognition rate 99.6% with false alarm rate at 5.2%. K. Yun *et al.* [136] achieves 0.8950 AUC. However, all the above methods can easily lead to over-fitting for limited samples and do not guarantee the same performance for new scenarios. While, our method is generalized,

robust to the over-fitting, and tested on the real traffic with various challenges in the videos. The dataset is made public for the research community for further comparison.

8.4 Summary

The main goal of this chapter is to design a system framework to deploy various visual computing based traffic monitoring applications in a city-scale surveillance camera network. We propose a framework for real-time detection of motorcyclists driving without helmets in diverse surveillance conditions. The proposed framework recognizes violators accurately as compared to the existing methods. Further, there is a significant reduction in the number of false alarms because of the use of cascaded CNNs. Even with such a high detection rate, our approach process incoming video stream in real-time. Though the current model is able to be work in real-time, the time performance can be further improved by applying model compression techniques. The incorporation of convolutional auto-encoder for deep feature representation in proposed framework for accident scene recognition outperforms the existing hand-crafted features based approaches. The method is further strengthened using complementary appearance and motion information together. The dual measures of the outlier scores and reconstruction error for detection of the accidents using complimentary modalities based on appearance, motion, and joint representation increase detection rate of the accidents. The incorporation of the collision of the intersection points of a vehicle's track reduce the false alarm rate, and thus enhances the reliability of the overall system. Since we are using STVVs instead of entire frame or full video clip, it not only detects the accident but also able to localize the accident events. The proposed method is able to detect on average 77.5% accidents correctly with 22.5% false alarms on real accidents videos captured under various lighting conditions. The experimental results are encouraging and show the efficacy of the proposed approach. However, challenges such as low visibility at night, occlusions, and large variations in the normal traffic pattern still pose significant challenges which need to be addressed in future. Also, the proposed framework automatically adapts to new scenarios if required, with slight tuning. This framework can be extended for detection of other rule violations as well as to detect and report number plates of violators.

Chapter 9

Summary and future work

In this thesis, we presented scalable and distributed methods for various visual computing tasks. We showed that the proposed methods were able to reduce the time complexity and the loss in the effectiveness of the various feature representation and modeling techniques. We reduce the computation time of BoW feature generation for both hard and soft vector-quantization with time complexities $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$, respectively, by replacing the process of finding the closest cluster center with a softmax classifier which is scaled by applying quantization and hashing on its weights. Further to reduce the computation time and increase the effectiveness of the video representation, we represented video action/activities as a graph of local feature descriptors extracted at key space-time interest points. This graph representation not only improves the recognition performance but also helps in spatio-temporal localization of the action/activities.

Then, we proposed distributed implementation of kernel SVM while sustaining classification performance. First, we propose *Genetic-SVM* which makes use of the distributed genetic algorithm for the fast optimization of the SVM objective function. Further, we proposed *DiP-SVM*, a distribution preserving kernel SVM which reduces the chance of missing important global support vectors by retaining the first and second order statistics of the entire dataset in each of the partitions. The *Projection-SVM* reduces both training and prediction time by avoiding the task of combining the local SVMs using subspace partitioning using a decision tree constructed on the projection of the data along the direction of maximum variance.

Also, we designed compact methods for the detection of helmet-less motorcyclists and accident incidents that are capable to infer in real-time on resource-constrained embedded hardware. Further, we demonstrated that the use of edge computing based framework reduces the network latency arise due to high communication overhead by placing edge devices near the video cameras.

9.1 Contributions of the thesis

1. *Fast-BoW* which scales the computational complexity of hard and soft BoW generation to $O(|\mathbf{h}| \log_2 k)$ and $O(|\mathbf{h}| k)$ from $O(d \log_2 k)$ and $O(d k)$, respectively, using model approximation while sustaining the effectiveness of the generated features.
2. A novel graph representation of video activities for better recognition and localization by leveraging the structural information among the various entities or same entity over the time.
3. *GeneticSVM*, an evolutionary computing based distributed approach to find optimal solution of kernel SVM. The novel encoding method and crossover operation help in obtaining the better solution over the GPU cluster.
4. *DiP-SVM*, a distribution preserving kernel SVM which obtains local decision boundaries in close agreement with the global decision boundary by retaining first and second order statistics of the entire dataset in each partition, thereby reducing the chance of missing important global support vectors.
5. *Projection-SVM*, construct a tree on the projection of data along the direction of maximum variance recursively to obtain smaller subspaces and train a kernel SVM on each subspace independently to reduce the overall training time which also reduces the prediction time significantly.
6. Efficient methods for edge device and a system architecture for real-time detection of traffic incidents such as traffic rule violation and accident detection in a city-wide video surveillance networks.

9.2 Directions for Further Research

In future, we would like to extend our edge-computing framework for various other traffic violations and incidents to strengthen the safety and security of road transportation. Also, we would like to investigate generative adversarial networks to improve the abnormal activity recognition and accident detection. For better storage and content based retrieval of the visual data, we would like to design efficient binary hashing techniques using binary or ternary weight only. We would also like to explore scalable and distributed kernel methods for non-linear feature selection and modeling of high dimensional big bio-medical data. Specifically, we would like to design an efficient non-linear least angle regression using kernel tricks which is both data and computation intensive because of the involvement of several large kernel matrices.

Bibliography

- [1] cifar. <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [2] DataSets. Available at <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>.
- [3] T. Abdullah, A. Anjum, M. F. Tariq, Y. Baltaci, and N. Antonopoulos. Traffic Monitoring Using Video Analytics in Clouds. In *IEEE/ACM Int. Conf. Utility and Cloud Computing*, pages 39–48, 2014.
- [4] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 30(3):555–560, 2008.
- [5] Ö. Aköz and M. E. Karşlıgil. Video-based Traffic Accident Analysis at Intersections using Partial Vehicle Trajectories. In *Proc. IEEE Signal Processing and Communications Applications Conf. (SIU)*, pages 4693–4696, Diyarbakir, Turkey, 22-24 Apr 2010.
- [6] N. K. Alham, M. Li, Y. Liu, and S. Hammoud. A MapReduce-based distributed SVM algorithm for automatic image annotation. *Computers and Mathematics with Applications*, 62(7):2801–2811, 2011.
- [7] B. Antic and B. Ommer. Video parsing for abnormality detection. In *Proc. IEEE Conf. on Computer Vision (ICCV)*, pages 2415–2422, Barcelona, Spain, Nov 6–13 2011.
- [8] C. Behera, R. Ravi, L. Sanjeev, and D. T. A comprehensive study of motorcycle fatalities in south delhi. *Journal of Indian Academy of Forensic Medicine*, 31(1):6–10, 2009.
- [9] M. Bertini, A. Del Bimbo, and L. Seidenari. Multi-scale and real-time non-parametric approach for anomaly detection and localization. *Computer Vision and Image Understanding*, 116(3):320–329, 2012.
- [10] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Machine Learning, Proc. Twenty-Third Int. Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 97–104, 2006.
- [11] J. Bian, X. Gao, and X. Wang. Bayesian matrix factorization for face recognition. In *IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, pages 2109–2113, 2013.
- [12] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

- [13] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby. Detecting anomalies in people’s trajectories using spectral graph analysis. *Computer Vision and Image Understanding*, 115(8):1099–1111, 2011.
- [14] F. Ö. Çatak and M. E. Balaban. CloudSVM: Training an SVM Classifier in Cloud Computing Systems. In *Pervasive Computing and the Networked World - Joint Int. Conference, ICPCA/SWS*, pages 57–68, Istanbul, Turkey, 28–30 Nov 2012.
- [15] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:1–27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [16] F. Chang, C.-Y. Guo, X.-R. Lin, C.-C. Liu, and C.-J. Lu. Tree Decomposition for Large-Scale SVM Problems. *J. Machine Learning Research*, 11:2935–2972, 2010.
- [17] R. Chaudhry, a. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1932–1939, Miami, FL, Jun 20–25 2009.
- [18] X.-w. Chen, S. Member, and X. Lin. Big Data Deep Learning : Challenges and Perspectives. *IEEE Access*, 2:514–525, 2014.
- [19] Y. L. Chen, T. S. Chen, T. W. Huang, L. C. Yin, S. Y. Wang, and T. C. Chiueh. Intelligent urban video surveillance system for automatic vehicle detection and tracking in clouds. In *Int. Conf. Advanced Information Networking and Applications (AINA)*, pages 814–821, 2013.
- [20] K.-w. Cheng, Y.-t. Chen, and W.-h. Fang. Video Anomaly Detection and Localization Using Hierarchical Feature Representation and Gaussian Process Regression. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2909–2917, Boston, Jun 8–10 2015.
- [21] C.-C. Chiu, M.-Y. Ku, and H.-T. Chen. Motorcycle detection and tracking system with occlusion segmentation. In *Proc. Int. Workshop on Image Analysis for Multimedia Interactive Services*, pages 32–32, Santorini, Greece, 6–8 June 2007.
- [22] J. Chiverton. Helmet presence classification with motorcycle detection and tracking. *IET Intelligent Transport Systems (ITS)*, 6(3):259–269, 2012.
- [23] Y. Cong, J. Yuan, and J. Liu. Sparse reconstruction cost for abnormal event detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3449–3456, Colorado Springs, CO, USA, Jun 20–25 2011.
- [24] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
- [25] K. Dahiya, D. Singh, and C. K. Mohan. Automatic detection of bike-riders without helmet using surveillance videos in real-time. In *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, pages 3046–3051, Vancouver, Canada, 24–29 July 2016.
- [26] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, San Diego, CA, USA, Jun 25–25 2005.

- [27] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *IEEE Trans. Information Theory*, 55(7):3229–3242, July 2009.
- [28] A. Dopfer and C.-c. Wang. What can we learn from accident videos ? In *Proc. Int. Automatic Control Conf. (CACCS)*, pages 68–73, Sun Moon Lake, Taiwan, 2–4 Dec 2013.
- [29] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. A Wiley-Interscience publication. Wiley, 1973.
- [30] A. Fischer and C. Igel. An Introduction to Restricted Boltzmann Machines. In *LNCS: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 7441, pages 14–36. 2012.
- [31] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proc. Computational Learning Theory and Kernel Machines*, pages 129–143, Washington, DC, USA, Aug 24–27 2003.
- [32] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: The cascade svm. In *Proc. of Advances in Neural Information Processing Systems*, pages 521–528, 2005.
- [33] S. Graf and H. Luschgy. *Foundations of Quantization for Probability Distributions*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.
- [34] R. L. Graham, G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca, and A. Lumsdaine. Open MPI: A high-performance, heterogeneous MPI. In *Proc. of IEEE Int. Conf. on Cluster Computing*, pages 1–9, 2006.
- [35] Z. Guo, D. Zhang, and L. Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Trans. Image Processing*, 19(6):1657–1663, 2010.
- [36] M. Hasan and A. K. Roy-Chowdhury. Continuous Learning of Human Activity Models using Deep Nets. In *European Conf. Computer Vision*, pages 705–720, 2014.
- [37] O. Herrera and A. Kuri. An approach to support vector regression with genetic algorithms. In *Proc. Fifth Mexican Int. Conf. on Artificial Intelligence, MICAI*, pages 178–186, 2006.
- [38] S. Herrero-Lopez. Accelerating SVMs by integrating GPUs into MapReduce clusters. In *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 1298–1305, 2011.
- [39] S. Herrero-lopez, J. R. Williams, and A. Sanchez. Parallel multiclass classification using SVMs on GPUs. In *Proc. of Workshop on General-Purpose Computation on Graphics Processing Units - GPGPU '10*, page 2, 2010.
- [40] C.-J. Hsieh, S. Si, and I. Dhillon. A Divide-and-Conquer Solver for Kernel Support Vector Machines. In *Proc. of Int. Conf. on Machine Learning*, volume 32, pages 566–574, 2014.
- [41] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, 2006.

- [42] Z. Hui, X. Yaohua, M. Lu, and F. Jiansheng. Vision-based real-time traffic accident detection. In *Proc. World Congress on Intelligent Control and Automation (WCICA)*, pages 1035–1038, Shenyang, China, 29 Jun–4 Jul 2014.
- [43] S. S. Husain and M. Bober. Improving large-scale image retrieval through robust aggregation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 39(9):1783–1796, 2017.
- [44] J.-w. Hwang, Y.-s. Lee, and S.-b. Cho. Hierarchical Probabilistic Network-based System for Traffic Accident Detection at Intersections. In *Proc. Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, pages 211–216, Shaanxi, China, 26–29 Oct 2010.
- [45] N. Hyeonseob and H. Bohyung. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302, Las Vegas, United States, June 26th - July 1st 2016.
- [46] N. Inoue and K. Shinoda. Fast coding of feature vectors using neighbor-to-neighbor search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 38(6):1170–1184, 2016.
- [47] R. Janning, C. Schatten, and L. Schmidt-Thieme. HNNP - A Hybrid Neural Network Plait for Improving Image Classification with Additional Side Information. In *IEEE Int. Conf. Tools with Artificial Intelligence*, pages 24–29, 2013.
- [48] D. Jeff, J. Yangqing, V. Oriol, H. Judy, Z. Ning, T. Eric, and D. Trevor. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *Int. Conf. on Machine Learning (ICML)*, 32(1):647–655, 2014.
- [49] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011.
- [50] F. Jiang, J. Yuan, S. a. Tsafaris, and A. K. Katsaggelos. Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, 115(3):323–333, 2011.
- [51] Z. Kaihua, L. Qingshan, Wu, and Y. Ming-Hsuan. Robust Visual Tracking via Convolutional Networks without Training. *IEEE Trans. Image Processing*, 25(4):1779–1792, 2016.
- [52] V. Kaltsa, A. Briassouli, I. Kompatsiaris, L. J. Hadjileontiadis, and M. G. Strintzis. Swarm Intelligence for Detecting Interesting Events in Crowded Environments. *IEEE Trans. on Image Processing*, 24(7):2153–2166, 2015.
- [53] X. Ke, R. Jin, X. Xie, and J. Cao. A Distributed SVM Method based on the Iterative MapReduce. In *Proc. of IEEE Int. Conf. on Semantic Computing (ICSC)*, number 4, pages 7–10, 2015.
- [54] P. Kelly and S. Moezzi. Visual Computing Laboratory. *IEEE Multimedia*, 2(1):94–99, 1995.
- [55] Y.-k. Ki and D.-y. Lee. A Traffic Accident Recording and Reporting Model at Intersections. *IEEE Trans. Intelligent Transportation Systems*, 8(2):188–194, 2007.

- [56] J. Kim and K. Grauman. Observe Locally, Infer Globally: a Space-Time MRF for Detecting Abnormal Activities with Incremental Updates. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2921–2928, Miami, FL, Jun 20–25 2009.
- [57] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, Lake Tahoe, Nevada, United States, 3–6 December 2012.
- [58] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE Int. Conf. Computer Vision*, pages 2556–2563, Barcelona, Spain, 6-13 Nov 2011.
- [59] I. Laptev. On space-time interest points. *Int. J. Computer Vision*, 64(2-3):107–123, 2005.
- [60] I. J. Lee. An Accident Detection System on Highway Using Vehicle Tracking Trace. In *Proc. Int. Conf. on Information and Communication Technology Convergence (ICTC)*, pages 716–721, Seoul, South Korea, 28–30 Sep 2011.
- [61] K. Li and J. Malik. Fast k-nearest neighbour search via dynamic continuous indexing. In *Proc. 33rd Int. Conf. Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 671–679, 2016.
- [62] N. Li, H. Guo, D. Xu, and X. Wu. Multi-scale Analysis of Contextual Information within Spatio-temporal Video Volumes for Anomaly Detection. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pages 2363–2367, Paris, France, 27–30 Oct 2014.
- [63] T. Li, H. Chang, M. Wang, B. Ni, and R. Hong. Crowded Scene Analysis : A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(3):367–386, 2015.
- [64] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan. Crowded scene analysis: A survey. *IEEE Trans. Circuits and Systems for Video Technology (TCSVT)*, 25(3):367–386, 2015.
- [65] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 36(1):18–32, 2014.
- [66] L. Liu, L. Wang, and C. Shen. A generalized probabilistic framework for compact codebook creation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 38(2):224–237, 2016.
- [67] W. Liu, H. Liu, D. Tao, Y. Wang, and K. Lu. Multiview Hessian regularized logistic regression for action recognition. *Signal Processing*, 110(5):101–107, 2015.
- [68] W. Liu, Z. J. Zha, Y. Wang, K. Lu, and D. Tao. P-Laplacian Regularized Sparse Coding for Human Activity Recognition. *IEEE Transactions on Industrial Electronics*, 63(8):5120–5129, 2016.
- [69] Z. Liu, H. Li, and G. Miao. MapReduce-based Backpropagation Neural Network over large scale mobile data. In *Int. Conf. Natural Computation*, pages 1726–1730, 2010.

- [70] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.
- [71] C. Lu, J. Shi, and J. Jia. Abnormal event detection at 150 FPS in MATLAB. In *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2720–2727, Sydney, Australia, Dec 1–8 2013.
- [72] Y. Lu, V. Roychowdhury, and L. Vandenberghe. Distributed Parallel Support Vector Machine in Strongly Connected Networks. *IEEE Trans. on Neural Networks*, 19(7):1167–1178, 2008.
- [73] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1975–1981, San Francisco, CA, USA, Jun 13–18 2010.
- [74] T. Mantec şn, C. R. del Blanco, F. Jaureguizar, and N. Garcia. Visual face recognition using bag of dense derivative depth patterns. *IEEE Signal Processing Letters*, 23(6):771–775, 2016.
- [75] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pages 935–942, 2009.
- [76] P. Merz and B. Freisleben. Genetic algorithms for binary quadratic programming. In *Proc. Genetic and Evolutionary Computation Conference*, pages 417–424, 1999.
- [77] MIT. StarCluster.
- [78] M. F. M ller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.
- [79] S. S. Mukherjee and N. M. Robertson. Deep head pose: Gaze-direction estimation in multimodal video. *IEEE Trans. Multimedia*, 17(11):2094–2107, Nov 2015.
- [80] A. Munawar, M. Wahib, M. Munetomo, and K. Akama. Advanced genetic algorithm to solve MINLP problems over GPU. In *Proc. of IEEE Congress of Evolutionary Computation (IEEE CEC)*, pages 318–325, 2011.
- [81] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1–21, 2015.
- [82] A. Navia-V zquez, D. Guti rrez-Gonz lez, E. Parrado-Hern ndez, and J. J. Navarro-Abell n. Distributed support vector machines. *IEEE Trans. on Neural Networks*, 17(4):1091–1097, 2006.
- [83] M. Oiso, T. Yasuda, K. Ohkura, and Y. Matumura. Accelerating steady-state genetic algorithms based on CUDA architecture. In *Proc. of IEEE Congress of Evolutionary Computation (IEEE CEC)*, pages 687–692, 2011.
- [84] J. W. Park, C. H. Yun, S.-g. Kim, H. Y. Yeom, and Y. W. Lee. Cloud computing platform for GIS image processing in U-city. In *Int. Conf. Advanced Communication Technology (ICACT2011)*, pages 1151–1155, 2011.
- [85] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conf.*, 2015.

- [86] O. P. Popoola and K. Wang. Video-Based Abnormal Human Behavior Recognition: A Review. *IEEE Trans. on Systems, Man, and Cybernetics*, 42(6):865–878, 2012.
- [87] R. Raghavendra, A. Del Bue, M. Cristani, and V. Murino. Optimizing interaction force for global anomaly detection in crowded scenes. In *Proc. IEEE Int. Conf. on Computer Vision Workshops (ICCVW)*, pages 136–143, Barcelona, Spain, Nov 6–13 2011.
- [88] A. Raghunathan, P. Jain, and R. Krishnaswamy. Learning mixture of gaussians with streaming data. In *Advances in Neural Information Processing Systems*, pages 6608–6617, Long Beach, CA, USA, 4-9 Dec 2017.
- [89] W. Rattapoom, B. Nannaphat, T. Vasan, T. Chainarong, and P. Pattanawadee. Machine vision techniques for motorcycle safety helmet detection. In *Proc. Int. Conf. Image and Vision Computing New Zealand (IVCNZ)*, pages 35–40, Wellington, New Zealand, 27–29 November 2013.
- [90] J. Redmon. Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>, 2013–2016.
- [91] J. Ren, Y. Chen, L. Xin, J. Shi, B. Li, and Y. Liu. Detecting and positioning of traffic incidents via video-based analysis of traffic states in a road segment. *IET Intelligent Transport Systems*, 10(6):428–437, 2016.
- [92] M. J. Roshtkhari and M. D. Levine. An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions. *Computer Vision and Image Understanding*, 117(10):1436–1452, 2013.
- [93] M. J. Roshtkhari and M. D. Levine. Online dominant and anomalous behavior detection in videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2611–2618, Portland, OR, USA, Jun 23–28 2013.
- [94] G. Ross, D. Jeff, D. Trevor, and M. Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, Columbus, Ohio, 24–27 June 2014.
- [95] M. Ryden, K. Oh, A. Chandra, and J. Weissman. Nebula: Distributed edge cloud for data-intensive computing. In *Int. Conf. Collaboration Technologies and Systems (CTS)*, pages 491–492, 2014.
- [96] C. Ryu, D. Lee, M. Jang, C. Kim, and E. Seo. Extensible Video Processing Framework in Apache Hadoop. In *IEEE Int. Conf. Cloud Computing Technology and Science*, pages 305–310, 2013.
- [97] S. Sadek, A. Al-hamadi, B. Michaelis, and U. Sayed. Real-time Automatic Traffic Accident Recognition Using HFG. In *Proc. Int. Conf. Pattern Recognition (ICPR)*, pages 3348–3351, Istanbul, Turkey, 23-26 Aug 2010.
- [98] V. Saligrama and Z. Chen. Video Anomaly Detection Based on Local Statistical Aggregates. In *IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2112–2119, 2012.
- [99] C. Sanderson and R. Curtin. Armadillo: a template-based C++ library for linear algebra. *J. Open Source Software*, 1:1–26, 2016.

- [100] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *Int. Conf. Pattern Recognition*, pages 32–36, Cambridge, UK, 23-26 Aug 2004.
- [101] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *IEEE Conf. Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 37–45, 2015.
- [102] J. P. Silva and A. R. d. R. Neto. Sparse Least Squares Support Vector Machines via Genetic Algorithms. In *BRICS Congress on Computational Intelligence*, pages 248–253, 2013.
- [103] R. V. Silva, T. Aires, and V. Rodrigo. Helmet detection on motorcyclists using image descriptors and classifiers. In *Proc. Graphics, Patterns and Images (SIBGRAPI)*, pages 141–148, Rio de Janeiro, Brazil, 27–30 August 2014.
- [104] D. Singh and C. K. Mohan. Distributed quadratic programming solver for kernel SVM using genetic algorithm. In *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 152–159, 2016.
- [105] D. Singh and C. K. Mohan. Graph formulation of video activities for abnormal activity recognition. *Pattern Recognition*, 65:265–272, 2017.
- [106] D. Singh, D. Roy, and C. K. Mohan. DiP-SVM : Distribution Preserving Kernel Support Vector Machine for Big Data. *IEEE Trans. Big Data*, 3(1):79–90, 2017.
- [107] D. Singh, C. Vishnu, and C. K. Mohan. Visual big data analytics for traffic monitoring in smart city. In *Proc. IEEE Conf. Machine Learning and Application (ICMLA)*, Anaheim, California, 18–20 December 2016.
- [108] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.
- [109] K. Subashini, S. Palanivel, and V. Ramalingam. Audio-video based segmentation and classification using SVM. In *IEEE Int. Conf. Computing, Communication and Networking Technologies*, pages 1–6, 2012.
- [110] W. Sultani and J. Y. Choi. Abnormal traffic detection using intelligent driver model. In *Proc. Int. Conf. Pattern Recognition (ICPR)*, pages 324–327, Istanbul, Turkey, 23–26 Aug 2010.
- [111] Z. Sun and G. Fox. Study on Parallel SVM Based on MapReduce. In *Int. Conf. Parallel and Distributed Processing Techniques and Applications*, pages 16–19, 2012.
- [112] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conf. Comput. Vision and Pattern Recognition*, pages 1701–1708, June 2014.
- [113] H. Tan, J. Zhang, and J. Feng. Vehicle Speed Measurement for Accident Scene Investigation. In *IEEE Int. Conf. E-Business Engineering*, pages 389–392, Shanghai, China, 10–12 Nov 2010.
- [114] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core VectorMachines: Fast SVMTraining on Very Large Data Sets. *Journal of Machine Learning Research*, 33(2):211–220, 2008.

- [115] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.
- [116] C. Vishnu, D. Singh, and C. K. Mohan. Detection of motorcyclists without helmet in videos using convolutional neural network. In *2017 Int. Joint Conf. Neural Networks (IJCNN)*, May 2017.
- [117] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 1449–1456, Vancouver, British Columbia, Canada, Dec 4–7 2006.
- [118] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(1):1201–1242, 2010.
- [119] M. Wahib, A. Munawar, M. Munetomo, and K. Akama. Optimization of Parallel Genetic Algorithms for nVidia GPUs. In *Proc. of IEEE Congress of Evolutionary Computation (IEEE CEC)*, pages 803–811, 2011.
- [120] H. Wang and S. Bengio. THE MNIST DATABASE Of handwritten upper-case letters, 2002.
- [121] H. Wang, A. Kl, C. Schmid, L. Cheng-lin, H. Wang, A. Kl, C. Schmid, L. C.-l. Action, and A. Kl. Action Recognition by Dense Trajectories. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, Colorado Springs, CO, USA, Jun 20–25 2011.
- [122] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pages 3551–3558, Sydney, Australia, Dec 1–8 2013.
- [123] K. Wang and Z. Shen. A GPU-Based Parallel Genetic Algorithm for Generating Daily Activity Plans. *IEEE Trans. on Intelligent Transportation Systems*, 13(3):1474–1480, 2012.
- [124] X. Wang and S. Matwin. A Distributed Instance-weighted SVM Algorithm on Large-scale Imbalanced Datasets. In *Proc. of IEEE Int. Conf. on Big Data*, pages 45–51, 2014.
- [125] Y.-K. Wang, C.-T. Fan, and J.-F. Chen. Traffic Camera Anomaly Detection. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 4642–4647, Stockholm, Sweden, Aug 24–28 2014.
- [126] Z. Wang, D. Xiang, S. Hou, and F. Wu. Background-driven salient object detection. *IEEE Trans. Multimedia*, 19(4):750–762, April 2017.
- [127] S. Wu, B. E. Moore, and M. Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pages 2054–2060, 2010.
- [128] S. Xia, J. Xiong, Y. Liu, and G. Li. Vision-based Traffic Accident Detection Using Matrix Approximation. In *Proc of the Asian Control Conf. (ASCC)*, pages 1–5, Kota Kinabalu, Malaysia, 31 May–3 Jun 2015.
- [129] C. Xu, D. Tao, and C. Xu. Multi-View Learning with Incomplete Views. *IEEE Transactions on Image Processing*, 24(12):5812–5825, 2015.

- [130] K. Xu, C. Wen, Q. Yuan, X. He, and J. Tie. A MapReduce based Parallel SVM for Email Classification. *Journal of Networks*, 9(6):1640–1647, 2014.
- [131] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA.*, pages 311–321, 1993.
- [132] Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: Communication-Avoiding Support Vector Machines on Distributed Systems. In *Proc. of IEEE Int. Parallel and Distributed Processing Symposium*, pages 847–859, Hyderabad, India, 2015.
- [133] Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. Design and Implementation of a Communication-Optimal Classifier for Distributed Kernel Support Vector Machines. *IEEE Trans. Parallel and Distributed Systems*, 28(4):974–988, 2017.
- [134] H. Yu, J. Yang, J. Han, and X. Li. Making svms scalable to large data sets using hierarchical cluster indexing. *Data Min. Knowl. Discov.*, 11(3):295–321, 2005.
- [135] G. Yuan, X. Zhang, Q. Yao, and K. Wang. Hierarchical and Modular Surveillance Systems in ITS. *IEEE Intelligent Systems*, 26(5):10–15, 2011.
- [136] K. Yun, H. Jeong, K. M. Yi, S. W. Kim, and J. Y. Choi. Motion Interaction Field for Accident Detection in Traffic Surveillance Video. In *Proc. Int. Conf. Pattern Recognition (ICPR)*, pages 3062 – 3067, Stockholm, Sweden, 24-28 Aug 2014.
- [137] L. Zanni, T. Serafini, and G. Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7:1467–1492, 2006.
- [138] X. Zeng and T. R. Martinez. Distribution-balanced stratified cross-validation for accuracy estimation. *J. Exp. Theor. Artif. Intell.*, 12(1):1–12, 2000.
- [139] C. Zhang and E.-c. Chang. Processing of Mixed-Sensitivity Video Surveillance Streams on Hybrid Clouds. In *IEEE Int. Conf. Cloud Computing*, pages 9–16, 2014.
- [140] H.-j. Zhang and N.-f. Xiao. Parallel implementation of multilayered neural networks based on Map-Reduce on cloud computing clusters. *Soft Computing*, 19(2):1–13, 2015.
- [141] J. Zhao, L. Wang, R. Cabral, and F. D. la Torre. Feature and region selection for visual learning. *IEEE Trans. Image Processing*, 25(3):1084–1094, 2016.
- [142] W. L. Zhao, C. W. Ngo, and H. Wang. Fast covariant vlad for image search. *IEEE Trans. Multimedia*, 18(9):1843–1854, 2016.

Publications

Patents

1. Dinesh Singh, C. Vishnu, D. Roy and C. Krishna Mohan, "Method and System for Detection of Accident in Traffic Surveillance Videos," *Indian Complete Patent*, Application no. 201841003604, Jan. 2018
2. Dinesh Singh, C. Vishnu, D. Roy and C. Krishna Mohan, "A Method and System for Real-time Detection of Traffic Violation by Two-wheeled Riders," *Indian Complete Patent*, Application no. 201741038813, Nov. 2017
3. D. Roy, Dinesh Singh, C. Vishnu and C. Krishna Mohan, "Method and System for Detection of Crime Events in Surveillance Videos," *Indian Complete Patent*, Application No: 201741041239, Nov. 2017

Journals

1. Dinesh Singh, Bedanta Das, Sumit Mamtani and C. Krishna Mohan, "Weakly-supervised Spatio-Temporal Action Localization based on Graph Representation in Untrimmed Videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (Under Review), Nov 2018
2. Dinesh Singh, C. Vishnu and C. Krishna Mohan, "Real-time Detection of Motorcyclists without Helmet using cascade of CNNs," *IEEE Transactions on Intelligent Transportation Systems*, (Under Review), Nov 2018
3. Dinesh Singh and C. Krishna Mohan, "Deep Spatio-Temporal Representation for Detection of Road Accident using Stacked Autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, May 2018. doi 10.1109/TITS.2018.2835308
4. Dinesh Singh and C. Krishna Mohan, "Graph formulation of video activities for abnormal activity recognition," *Pattern Recognition (Elsevier)*, vol.65, pp.265-272, May 2017
5. Dinesh Singh, D. Roy and C. Krishna Mohan, "DiP-SVM: Distribution Preserving Kernel Support Vector Machine for Big Data," *IEEE Transactions on Big Data*, vol.3, pp.79-90, Jan 2017

Conferences

1. Dinesh Singh and C. Krishna Mohan, "Projection-SVM: Distributed Kernel Support Vector Machine for Big Data using Subspace Partitioning," accepted in *IEEE International Conference on Big Data (IEEE BigData 2018)*, Seattle, WA, USA, Dec 2016
2. Dinesh Singh, A Bhure, S Mamtani and C. Krishna Mohan, "Fast-BoW: Scaling Bag-of-Visual-Words Generation," in *Proc. British Machine Vision Conference (BMVC 2018) Newcastle upon Tyne, UK, Sep 2018, pp.1–10*
3. C. Vishnu, Dinesh Singh, C. Krishna Mohan and S Babu, "Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network," in *Proc. Int. Joint Conf. Neural Network (IJCNN 2017)*, Anchorage, AK, USA, pp. 3036–3041, May 2017
4. Dinesh Singh, C. Vishnu and C. Krishna Mohan, "Visual Big Data Analytics for Traffic Monitoring in Smart City," in *Proc. IEEE Int. Conf. Machine Learning and Applications (IEEE ICMLA 2016)*, Anaheim, CA, USA, pp. 886–891, Dec 2016
5. Nazil Perveen, Dinesh Singh and C. Krishna Mohan, "Spontaneous Facial Expression Recognition: A Part Based Approach," in *Proc. IEEE Int. Conf. Machine Learning and Applications (IEEE ICMLA 2016)*, Anaheim, CA, USA, pp. 819–824, Dec 2016
6. Dinesh Singh and C. Krishna Mohan, "Distributed Quadratic Programming Solver for Kernel SVM using Genetic Algorithm," in *Proc. IEEE Congress on Evolutionary Computation (IEEE CEC 2016)*, Vancouver, BC, Canada, pp. 152–159, Jul 2016
7. K Dahiya, Dinesh Singh and C. Krishna Mohan, "Automatic Detection of Bike-riders without Helmet using Surveillance Videos in Real-time," in *Proc. Int. Joint Conf. on Neural Networks (IJCNN 2016)*, Vancouver, BC, Canada, pp. 3046–3051, Jul 2016

Award/Recognition

- Won first prize in the **8th IDRBT Doctoral Colloquium** held on December 7, 2018 for the paper on "Graph Formulation of Video Activities for Abnormal Activity Recognition". First prize carries a prize money of **₹1,00,000** along with a certificate.

CURRICULUM VITAE

Name: Dinesh Singh

Date of Birth: December 26, 1988

Educational Qualifications:

- July 2014 - present : *Doctor of Philosophy*, Computer Science and Engineering, Indian Institute of Technology Hyderabad
- July 2011 - June 2013 : *Master of Technology*, Computer Engineering, National Institute of Technology Surat
- July 2006 - June 2010 : *Bachelor of Technology*, Information Technology, R. D. Engineering College, Ghaziabad

DOCTORAL COMMITTEE MEMBERS

Chairperson: Dr. M. V. Panduranga Rao

Advisor: Prof. C. Krishna Mohan

Members:

- Dr. Sathya Peri (Dept. of CSE)
- Dr. Manish Singh (Dept. of CSE)
- Dr. Sumohana S. Channappayya (Dept. of EE)