# DETECTION AND CLASSIFICATION OF ABNORMAL EVENTS IN SURVEILLANCE SCENES

A Thesis submitted to Gujarat Technological University

for the Award of

**Doctor of Philosophy**

in

**Computer / IT Engineering**

by

**Kinjal Vishnuprasad Joshi**
169999913006

under supervision of

**Dr. Narendra M. Patel**

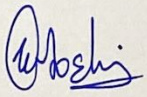

**GUJARAT TECHNOLOGICAL UNIVERSITY**
**AHMEDABAD**

**July - 2022**

# DECLARATION

I declare that the thesis entitled **Detection and Classification of Abnormal Events in Surveillance Scenes** submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from **May 2017 to December 2021** under the supervision of **Dr. Narendra M. Patel** and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.

Signature of the Research Scholar:  Date: 6/7/22

Name of Research Scholar: Kinjal Vishnuprasad Joshi

Place: V V Nagar, Anand

# CERTIFICATE

I certify that the work incorporated in the thesis **Detection and Classification of Abnormal Events in Surveillance Scenes** submitted by **Ms. Kinjal Vishnuprasad Joshi** was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of Supervisor: Date: 6/7/22

Name of Supervisor: Dr. Narendra M. Patel

Place: V V Nagar, Anand

# Course-work Completion Certificate

This is to certify that **Ms. Kinjal Vishnuprasad Joshi** enrolment no.**169999913006** is a PhD scholar enrolled for PhD program in the branch **Computer/IT Engineering** of Gujarat Technological University, Ahmedabad.

**(Please tick the relevant option(s))**

☐ He/She has been exempted from the course-work (successfully completed during M.PhilCourse)

☐ He/She has been exempted from Research Methodology Course only (successfully completedduring M.Phil Course)

☑ ~~He~~/She has successfully completed the PhD course work for the partial requirement for theaward of PhD Degree. His/ Her performance in the course work is as follows-

| Grade Obtained in Research Methodology (PH001) | Grade Obtained in Self Study Course (Core Subject) (PH002) |
|---|---|
| AB | AB |

Supervisor's Sign

Dr. Narendra M. Patel

# Originality Report Certificate

It is certified that PhD Thesis titled **Detection and Classification of Abnormal Events in Surveillance Scenes** by **Kinjal Vishnuprasad Joshi** has been examined by us. We undertake the following:

a. Thesis has significant new work / knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled / analysed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using **Ouriginal software** (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index <10%).

Signature of the Research Scholar: Date: 6/7/22

Name of Research Scholar: Kinjal Vishnuprasad Joshi

Place: V V Nagar, Anand

Signature of Supervisor: Date: 6/7/22

Name of Supervisor: Dr. Narendra M. Patel

Place: V V Nagar, Anand

Curiginal

## Document Information

| | |
|---|---|
| **Analyzed document** | Thesis_169999913006.pdf (D130224231) |
| **Submitted** | 2022-03-13T14:27:00.0000000 |
| **Submitted by** | Kinjal |
| **Submitter email** | KINJALJOSHI@GCET.AC.IN |
| **Similarity** | 3% |
| **Analysis address** | kinjaljoshi.gtuni@analysis.urkund.com |

## Sources included in the report

| | | |
|---|---|---|
| W | URL: http://livrepository.liverpool.ac.uk/3033096/1/bare_jrnl.pdf<br>Fetched: 2021-05-14T11:22:57.8370000 | 8 |
| SA | **Report_Asouis_Bousselham_v0.0.7.docx**<br>Document Report_Asouis_Bousselham_v0.0.7.docx (D110451342) | 2 |
| W | URL: https://en.wikipedia.org/wiki/Autoencoder<br>Fetched: 2019-09-25T09:12:24.5530000 | 2 |
| W | URL: https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64<br>Fetched: 2019-10-05T16:21:03.5200000 | 5 |
| W | URL: http://users.stat.ufl.edu/~winner/sta6208/notes1.pdf<br>Fetched: 2020-11-18T11:13:17.2830000 | 3 |
| SA | **Manuscript-Revised.docx**<br>Document Manuscript-Revised.docx (D77621646) | 4 |
| W | URL: https://etheses.whiterose.ac.uk/22443/1/Anomaly%20detection%20in%20Video%20-%20Hanh%20Tran.pdf<br>Fetched: 2020-11-18T11:54:24.7470000 | 8 |
| W | URL: http://www.ijngc.perpetualinnovation.net/index.php/ijngc/article/download/185/22<br>Fetched: 2021-10-29T07:18:41.5900000 | 2 |
| W | URL: https://deepai.org/publication/realgait-gait-recognition-for-person-re-identification<br>Fetched: 2022-01-17T18:59:36.9300000 | 1 |
| SA | **Automated Crowd Anomaly Detection and Localization using Video Analysis by Suprit Dhananjay Bansod (Ph.D.Engineering).pdf**<br>Document Automated Crowd Anomaly Detection and Localization using Video Analysis by Suprit Dhananjay Bansod (Ph.D.Engineering).pdf (D54487556) | 2 |
| SA | **final report.pdf**<br>Document final report.pdf (D73695691) | 1 |

# PhD THESIS Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being a PhD Research Scholar at GTU and in the interests of the facilitation of research at GTU and elsewhere, I, **Kinjal Vishnuprasad Joshi** having Enrollment No. **169999913006** hereby grant a non-exclusive, royalty free and perpetual license to GTU on the following terms:

a) GTU is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;
b) GTU is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);
c) GTU is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";
d) The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;
e) I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.
f) I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.
g) If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have

obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.

h) I retain copyright ownership and moral rights in my thesis, and may deal with the copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.

i) I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this non- exclusive license.

j) I am aware of and agree to accept the conditions and regulations of PhD including allpolicy matters related to authorship and plagiarism.

Signature of the Research Scholar: Date: 6/7/22

Name of Research Scholar: Kinjal Vishnuprasad Joshi

Place: V V Nagar, Anand

Signature of Supervisor: Date: 6/7/22

Name of Supervisor: Dr. Narendra M. Patel

Place: V V Nagar, Anand

Seal:

# Thesis Approval Form

The viva-voce of the PhD Thesis submitted by **Ms. Kinjal Vishnuprasad Joshi** (Enrollment No. **169999913006**) entitled **Detection and Classification of Abnormal Events in Surveillance Scenes** was conducted on ...6/7/22... at Gujarat Technological University.

**(Please tick any one of the following option)**

☑ The performance of the candidate was satisfactory. We recommend that he/she be awarded the PhD degree.

☐ Any further modifications in research work recommended by the panel after 3 months from the date of first viva-voce upon request of the Supervisor or request of Independent Research Scholar after which viva-voce can be re-conducted by the same panel again.

| (briefly specify the modifications suggested by the panel) |
|---|
| |

☐ The performance of the candidate was unsatisfactory. We recommend that he/she should not be awarded the PhD degree.

| (The panel must give justifications for rejecting the research work) |
|---|
| |

6/7/22

Dr. Narendra M Patel
Name and Signature of Supervisor with Seal

Dr. Shobha G.
1) (External Examiner 1) Name and Signature

Dr. Sanjay Garg
2) (External Examiner 2) Name and Signature

Dr. Sumantra Dutta Roy
3) (External Examiner 3) Name and Signature

# ABSTRACT

Automatic abnormal event detection and recognition in a surveillance scene is very significant because of more consciousness about public safety. Because of usefulness and complexity, currently, it is an open research area. Automatic abnormal event detection is a challenging task because the definition of abnormality is subjective. A normal event in one situation can be considered an abnormal event in another case. In the surveillance video with a dense crowd, automatic anomaly detection becomes very difficult because of clutter and severe occlusion.

As per situation, supervised and unsupervised both types of approaches are applicable to detect anomalous behaviour. As many researchers are doing work in this domain, various benchmark datasets are publicly available. In this work we have used six different datasets namely, UMN, Airport-WrongDir, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local. The detected abnormal events are running, moving in opposite direction, vehicle entry at restricted place, Violent behaviour of a crowd, avoiding payment at entry gate etc. We have also classified various abnormal events like Arrest, Assault, Burglary, Robbery, Stealing, and Vandalism.

In this research work, supervised machine learning algorithms and supervised and unsupervised deep learning algorithms are proposed to detect abnormal events in a crowd scene. The performance is measured by accuracy and Area Under the Curve (AUC). The experimental results determine the efficacy of the proposed approaches.

Based on supervised machine learning algorithms, two methods are proposed. In the first one, a combination of Histogram of Oriented Gradients (HOG) and Optical flow features is used with Neural Network and Support Vector Machine classifiers. As the HOG features represent shape information and optical flow represents motion, the hybrid features provide higher accuracy compared to individual features using both classifiers. The second method uses features extracted by autoencoder and optical flow features with the Neural Network classifier. As the encoder component of autoencoder extracts important features from an

image, when it is combined with optical flow it provides higher accuracy compared to individual features.

Based on supervised deep learning algorithms, three approaches are proposed, namely Stacked Autoencoder based approach, Convolutional Neural Network (CNN) based approach and Convolutional Neural Network and Long Short Term Memory network (CNN-LSTM) based approach. For Stacked autoencoder based approach, two encoders are stacked with softmax layer. It provides adequate results using various datasets. For the CNN based approach, the new CNN architecture is developed which performs better than the pretrained GoogLeNet architecture. The developed CNN architecture is evaluated using different partitions of data. The experiments are also done using the combined dataset which includes all images of four different datasets, for which the achieved accuracy is 99.79% and AUC value is 1. We have also applied SVM classifier with deep features extracted using the proposed CNN architecture, for which the achieved accuracy is 99.88% and AUC value is 0.99 using the combined dataset. CNN-LSTM architecture is developed using the proposed CNN architecture. LSTM network uses temporal information with spatial features, so CNN-LSTM approach gives better performance on challenging weakly labelled UCFCrime2Local dataset. The CNN and CNN-LSTM based approaches are applied for classification of various abnormal events in which CNN-LSTM performs better.

Although supervised learning based approaches provide excellent results using various datasets, unsupervised learning based approaches are also important because every time at every situation the labelled data may not be available. For every type of anomalous behaviour labels cannot be generated, so two unsupervised deep learning architectures are presented, i.e. Convolutional Autoencoder for spatial domain and Long Short Term Memory (LSTM) Autoencoder for spatiotemporal domain. For LSTM autoencoder based approach, features extracted by CNN architecture are used, for that experiments are done using the proposed CNN architecture and GoogLeNet architecture. LSTM autoencoder provides adequate results using various datasets.

# Acknowledgement

This thesis is the result of a journey of work where I have been accompanied and supported by many people. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them. Firstly, I offer my adoration to God for giving me the strength, direction and enthusiasm to complete my research work. I would like to express my deep gratitude to my Guide Dr. Narendra M. Patel, Professor, Department of Computer Engineering, Birla Vishwakarma Mahavidyalaya, V V Nagar for his guidance and constant support during this entire period. I would like to thank my doctoral progress committee: Dr. Narendra C. Chauhan, Professor and Head, Department of Information Technology, A D Patel Institute of Technology, V V Nagar and Dr. Tanmay D. Pawar, Professor and Head, Department of Electronics Engineering, Birla Vishwakarma Mahavidyalaya, V V Nagar for their insightful suggestions and encouragement during the entire period of research work.

I am grateful to Dr. H. B. Soni, Principal, Dr. Maulika S. Patel, Head of Computer Engineering department and my colleagues (teaching and non-teaching) at G H Patel College of Engineering & Technology (GCET) for their help, motivation and support.

I extend my heartfelt gratitude to my parents, whose hard work has raised me at this level. I would like to thank my husband Devang for his understanding and support throughout my research work. I am also grateful to my son Dheer whose love has inspired me during the entire work.

I thank my sisters for their love and support. I am thankful to one and all who were involved directly or indirectly in this long journey.

Kinjal Joshi

# Table of Content


Declaration……. ii
Certificate……. iii
Course-work Completion Certificate…… iv
Originality Report Certificate……. v
Ph.D. THESIS Non-Exclusive License…….. vii
Thesis Approval Form…… ix
Abstract……... x
Acknowledgment…… xii
Table of Content……. xiii
List of Abbreviations…….. xvi
List of Figures…… xviii
List of Tables…….. xx

**1 Introduction…… 1**
1.1 Overview…….. 1
1.2 Motivation…… 3
1.3 Objective and Scope of Work…….. 3
1.4 Dataset Description……. 4
1.5 Original Contribution by the Thesis …… 6
1.6 Organization of the Thesis……... 10

**2 Literature Review……..... 11**
2.1 Handcrafted Features Based Approaches……. 11
2.2 Deep Learning Based Approaches……..... 14
2.3 Performance Evaluation Metrics……. 16
2.3.1 Confusion Matrix……..... 16
2.3.2 Accuracy……. 16
2.3.3 Area Under the Curve ……. 17
2.4 Definition of the Problem…… 19

**3 Approaches Based on Machine Learning Algorithms……..... 20**
3.1 HOG and Optical Flow Based Approach…….. 20
3.1.1 Foregroud Extraction…….. 21
3.1.2 Corner Detection……. 22
3.1.3 Histogram of Oriented Gradients……... 24
3.1.4 Optical Flow …….. 25
3.1.4.1 Optical Flow Equation…….. 25
3.1.4.2 Horn–Schunck Method……..... 27
3.1.5 Principal Component Analysis……..... 28
3.1.6 Neural Network and Support Vector Machine……. 29
3.1.6.1 Neural Network……... 29

| | | |
|---|---|---|
| | 3.1.6.2 Support Vector Machine……..... | 30 |
| | 3.1.7 Experimental Results……... | 32 |
| 3.2 | Autoencoder and Optical Flow Based Approach…….. | 34 |
| | 3.2.1 Autoencoder ……... | 34 |
| | 3.2.2 The Proposed Approach…….. | 37 |
| | 3.2.3 Implementation Details and Experimental Results……. | 38 |
| 3.3 | Comparison of the Proposed Approaches……. | 40 |
| 3.4 | Concluding Remarks…… | 41 |
| **4** | **Approaches Based on Supervised Deep Learning Algorithms……..……** | **42** |
| 4.1 | Introduction……. | 42 |
| 4.2 | Stacked Autoencoder Based Approach…… | 43 |
| | 4.2.1 Stacked Autoencoder…….……. | 43 |
| | 4.2.2 The Proposed Approach…….…… | 43 |
| | 4.2.3 Experimental Results…….. | 45 |
| 4.3 | Convolutional Neural Network Based Approach…… | 46 |
| | 4.3.1 Convolutional Neural Network……..……. | 46 |
| | 4.3.2 Transfer Learning Approach……..……. | 48 |
| | 4.3.2.1 GoogLeNet Architecture……...……. | 48 |
| | 4.3.2.2 Implementation Details and Experimental Results…… | 49 |
| | 4.3.3 The Proposed CNN Architectures……..……. | 50 |
| | 4.3.3.1 CNN Architectre1……. | 51 |
| | 4.3.3.2 CNN Architecture2……...…… | 55 |
| 4.4 | CNN-LSTM Based Approach…….…… | 71 |
| | 4.4.1 LSTM Network……...……. | 71 |
| | 4.4.2 BiLSTM Network……..……. | 72 |
| | 4.4.3 CNN-LSTM Architecture…….... | 73 |
| | 4.4.4 Experimental Results…….……. | 74 |
| 4.5 | Comparison of the Proposed Approaches…… | 75 |
| 4.6 | Concluding Remarks…….……... | 77 |
| **5** | **Approaches Based on Unsupervised Deep Learning Algorithms……** | **79** |
| 5.1 | Convolutional Autoencoder Based Approach……..……..…….. | 79 |
| | 5.1.1 Convolutional Autoencoder……..……..…… | 79 |
| | 5.1.2 The Proposed Architecture……..……..…….. | 80 |
| | 5.1.3 Experimental Results……..……..…….. | 82 |
| 5.2 | LSTM Autoencoder Based Approach……..……..…….. | 83 |
| | 5.2.1 The Proposed Approach……..……..……. | 84 |
| | 5.2.2 Experimental Results……..……..…….. | 86 |
| 5.3 | Comparison of the Proposed Approaches…… | 89 |
| 5.4 | Comparison of all the Proposed Approaches with State of the Art Methods…….. | 89 |
| 5.5 | Concluding Remarks…… | 94 |
| **6** | **Conclusion and Future Scope……..……..……... ** | **95** |

6.1 Conclusion……..……..……..……..……..……. 95
6.2 Future Scope……..……..……..……..……..…….. 96
**List of References……...……...……..... 98**
**List of Publications…….…...……...……..... 105**

# List of Abbreviation

| | |
|---|---|
| Ab | Abnormal |
| AdaBoost | Adaptive Boosting |
| AE | Autoencoder |
| AMC | Adjacency-Matrix based Clustering |
| ANN | Artificial Neural Network |
| AUC | Area Under the Curve |
| BiLSTM | Bi-directional Long Short Term Memory |
| BM | Bayesian Model |
| CCTV | Closed-Circuit Television |
| CI | Chaotic Invariants |
| CMI | Crowd Motion Intensity |
| CNN | Convolutional Neural Network |
| ConvAE | Convolutional Autoencoder |
| ConvVAE | Convolutional Variational Autoencoder |
| DOC | Deep One Class |
| FF | Force Field |
| FN | False Negative |
| FNN | Feedforward Neural Network |
| FP | False Positive |
| FPR | False Positive Rate |
| GAN | Generative Adversarial Nets |
| GMM | Gaussian Mixture Model |
| GPR | Gaussian Process Regression |
| HOCG | Histogram of Oriented Contextual Gradient |
| HOG | Histogram of Oriented Gradients |
| HOF | Histogram of Optical Flow |
| HOIF | Histograms of the Orientation of Interaction Force |
| HOT | Histogram of Oriented Tracklets |
| IF | Interaction Force |
| LBPCM | Local Binary Pattern Co-Occurrence Matrix |
| LSTM | Long Short Term Memory |
| LTP | Local Trinary Patterns |
| MATLAB | MATrix LABoratory |
| MDT | Mixture of Dynamic Textures |
| MPPCA | Mixture of Probabilistic Principal Component Analyzers |
| MSE | Mean Squared Error |
| N | Normal |
| NN | Neural Network |
| OViF | Oriented Violent Flows |

| | |
|---|---|
| PCA | Principal Component Analysis |
| RBF | Radial basis function |
| R-CNN | Region-based Convolutional Neural Network |
| R-ConvAE | Recurrent Convolutional Autoencoder |
| R-ConvVAE | Recurrent Convolutional Variational Autoencoder |
| ReLU | Rectified Linear Unit |
| RGB | Red, Green and Blue |
| RNN | Recurrent Neural Network |
| ROC | Receiver Operating Characteristic |
| ROI | Region of Interest |
| SC | Sparse Cuboids |
| SF | Social Force |
| SGD | Stochastic Gradient Descent |
| SGDM | Stochastic Gradient Descent with Momentum |
| SHOT | Simplified Histogram of Oriented Tracklets |
| SRC | Sparse Reconstruction Cost |
| SVM | Support Vector Machine |
| TCNN | Tube Convolutional Neural Network |
| TN | True Negative |
| TP | True Positive |
| TPR | True Positive Rate |
| UCSD | University of California and San Diego |
| UMN | University of Minnesota |
| VGG | Visual Geometry Group |
| ViF | Violent Flows |

# List of Figures


| | | |
|---|---|---|
| FIGURE 1.1 | Crowd analysis modules……..... | 2 |
| FIGURE 1.2 | Normal and abnormal sample images of various datasets……. | 5 |
| FIGURE 1.3 | Sample images of UCFCrime2Local dataset……. | 6 |
| FIGURE 2.1 | TP vs. FP rate at different classification thresholds……... | 17 |
| FIGURE 2.2 | Area under the ROC curve…….... | 18 |
| FIGURE 3.1 | HOG and optical flow based approach……..... | 21 |
| FIGURE 3.2 | (a) Input frame (b) Extracted foreground…….. | 22 |
| FIGURE 3.3 | Detected corners…… | 24 |
| FIGURE 3.4 | HOG features around corners…… | 25 |
| FIGURE 3.5 | Optical flow representation…… | 27 |
| FIGURE 3.6 | Biological neuron and basic ANN architecture……. | 29 |
| FIGURE 3.7 | Simple feedforward neural network……..... | 30 |
| FIGURE 3.8 | Two dimensional linearly separable data…….. | 31 |
| FIGURE 3.9 | Optimal hyperplane…… | 31 |
| FIGURE 3.10 | Generated pattern recognition network for the UMN dataset scene - 3…… | 33 |
| FIGURE 3.11 | Performance of HOG and optical flow features using NN and SVM classifiers ……..... | 34 |
| FIGURE 3.12 | Autoencoder……... | 35 |
| FIGURE 3.13 | Sparse autoencoder…… | 36 |
| FIGURE 3.14 | Autoencoder and optical flow based approach……..... | 38 |
| FIGURE 3.15 | Features extracted by autoencoder……..... | 39 |
| FIGURE 3.16 | Generated pattern recognition network for the UMN dataset scene - 1…… | 39 |
| FIGURE 3.17 | Performance of optical flow and AE features using neural network classifier……. | 40 |
| FIGURE 3.18 | Performance of the proposed approaches using neural network classifier…… | 41 |
| FIGURE 4.1 | Performance evaluation for deep learning and machine learning algorithms……... | 42 |
| FIGURE 4.2 | Stacked autoencoder based approach…… | 44 |
| FIGURE 4.3 | Generated stacked network…… | 45 |
| FIGURE 4.4 | CNN architecture for object classification…… | 46 |
| FIGURE 4.5 | ROC curve for the UMN dataset using GoogLeNet architecture……..... | 50 |
| FIGURE 4.6 | The proposed CNN Architecture1……. | 51 |
| FIGURE 4.7 | The largest activation at each convolution layer using CNN Architecture1……. | 54 |
| FIGURE 4.8 | ROC curves using CNN Architecture1……. | 55 |
| FIGURE 4.9 | The proposed CNN Architecture2……. | 56 |
| FIGURE 4.10 | The largest activation at each convolution layer using CNN Architecture2……. | 58 |
| FIGURE 4.11 | Training progress of CNN Architecture2 using the UMN dataset ……... | 60 |
| FIGURE 4.12 | Training progress of CNN Architecture2 using the combined dataset……..... | 60 |
| FIGURE 4.13 | ROC curves for the combined dataset with 20% test data……..... | 62 |
| FIGURE 4.14 | ROC curves for the combined dataset with 30% test data……..... | 63 |

FIGURE 4.15 ROC curves for the combined dataset with 10-fold cross-validation…… 63
FIGURE 4.16 ROC curves for abnormal event detection using the UCFCrime2Local dataset……... 66
FIGURE 4.17 The largest activation at each convolution layer for the arrest event…… 67
FIGURE 4.18 The largest activation at each ReLU layer for the arrest event……..... 67
FIGURE 4.19 The largest activation at each max pooling layer for the arrest event……... 68
FIGURE 4.20 The largest activation at each batch normalization layer for the arrest event……..... 68
FIGURE 4.21 ROC curves for events classification using the UCFCrime2Local dataset as per given train - test split……..... 70
FIGURE 4.22 ROC curves for events classification using the UCFCrime2Local dataset with 20% test data……..... 71
FIGURE 4.23 ROC curves for events classification using the UCFCrime2Local dataset with 5-fold cross-validation…….. 71
FIGURE 4.24 LSTM architecture……. 72
FIGURE 4.25 BiLSTM network……... 73
FIGURE 4.26 CNN-LSTM architecture……... 73
FIGURE 4.27 ROC curves for classification results using the CNN-LSTM architecture……..... 75
FIGURE 4.28 Performance of the proposed approaches for abnormal event detection……..... 76
FIGURE 4.29 Performance of the proposed approaches for events classification……... 76
FIGURE 5.1 Input and reconstructed images……. 82
FIGURE 5.2 The proposed LSTM autoencoder architecture……..... 85
FIGURE 5.3 Reconstruction error over the UMN test data using LSTMAE……. 88
FIGURE 5.4 Performance of the proposed approaches using various datasets……..... 89
FIGURE 5.5 Performance of the proposed deep learning based approaches……. 93

# List of Tables


| | | |
|---|---|---|
| TABLE 2.1 | Confusion matrix for binary classification…….. | 16 |
| TABLE 3.1 | Experimental results using neural network…….. | 33 |
| TABLE 3.2 | Experimental results using SVM…….. | 33 |
| TABLE 3.3 | Confusion matrix for the UMN dataset…… | 40 |
| TABLE 4.1 | Experimental results for abnormal event detection using stacked autoencoder…….. | 45 |
| TABLE 4.2 | Confusion matrices for various datasets using stacked autoencoder……... | 45 |
| TABLE 4.3 | GoogLeNet architecture……... | 49 |
| TABLE 4.4 | CNN Architecture1……... | 51 |
| TABLE 4.5 | Experimental results using CNN Architecture1……... | 54 |
| TABLE 4.6 | CNN Architecture2……... | 57 |
| TABLE 4.7 | Experimental results using CNN Architecture2 with 20% test data……... | 61 |
| TABLE 4.8 | Experimental results using CNN Architecture2 with 30% test data……..... | 61 |
| TABLE 4.9 | Experimental results using CNN Architecture2 with 10-fold cross-validation…….. | 62 |
| TABLE 4.10 | Confusion matrices for various datasets using CNN classifier with 20% test data..... | 64 |
| TABLE 4.11 | Confusion matrices for various datasets using SVM classifier with 20% test data…. | 64 |
| TABLE 4.12 | Confusion matrices for various datasets using CNN classifier with 30% test data…. | 64 |
| TABLE 4.13 | Confusion matrices for various datasets using SVM classifier with 30% test data…. | 64 |
| TABLE 4.14 | Confusion matrices for various datasets using CNN classifier with 10-fold cross-validation……. | 64 |
| TABLE 4.15 | Confusion matrices for various datasets using SVM classifier with 10-fold cross-validation…….. | 65 |
| TABLE 4.16 | Confusion matrix for the UCFCrim2Local dataset using CNN classifier…… | 66 |
| TABLE 4.17 | Confusion matrix for the UCFCrim2Local dataset using SVM classifier……... | 66 |
| TABLE 4.18 | Classification results using the UCFCrime2Local dataset as per given train-test split…..…… | 69 |
| TABLE 4.19 | Classification results using the UCFCrime2Local dataset with 20% test data…… | 69 |
| TABLE 4.20 | Classification results using the UCFCrime2Local dataset with 5-Fold cross-validation……. | 70 |
| TABLE 4.21 | Overall classification accuracy…… | 70 |
| TABLE 4.22 | The proposed LSTM architecture……..... | 74 |
| TABLE 4.23 | Experimental results for abnormal event detection using the CNN-LSTM architecture…..…….. | 74 |
| TABLE 4.24 | Confusion matrices for various datasets using the CNN-LSTM architecture…….... | 75 |
| TABLE 4.25 | Experimental results for events classification using the CNN-LSTM architecture..... | 75 |
| TABLE 5.1 | The proposed convolutional autoencoder architecture……. | 81 |
| TABLE 5.2 | Reconstruction error for various datasets using ConvAE…… | 82 |
| TABLE 5.3 | Experimental results for abnormal event detection using ConvAE …… | 83 |

TABLE 5.4 The proposed LSTM autoencoder architecture…… 85
TABLE 5.5 Reconstruction error for various datasets using LSTMAE…….. 86
TABLE 5.6 Experimental results for abnormal event detection using LSTMAE……..... 87
TABLE 5.7 Confusion matrices for various datasets using LSTMAE with GoogLeNet Architecture…….. 87
TABLE 5.8 Confusion matrices for various datasets using LSTMAE with the proposed CNN Architecture2…… 88
TABLE 5.9 Performance of various methods on the UMN dataset …… 90
TABLE 5.10 Performance of various methods on the UCSDPed1 dataset……... 91
TABLE 5.11 Performance of various methods on the Violent Flows dataset……... 92
TABLE 5.12 Performance of various methods on the Subway Entrance dataset…….. 92
TABLE 5.13 Performance of various methods on the UCFCrim2Local dataset……... 93

# CHAPTER - 1

# Introduction

## 1.1 Overview

Nowadays, the demand for security of people and personal properties is constantly increasing, so video surveillance has become major daily anxiety. The prominence of this demand has led to the placement of cameras widely, which produce a large quantity of video. Most existing video surveillance systems are fully supervised by humans. Video monitoring is a very cumbersome and time-consuming task. A human can't find out the abnormal events from large size videos. However, even one small mistake could cause an unacceptable loss. Thus, it is essential to develop a system dealing with many video frames and alert people for a punctual and functional response when an abnormal event occurs. So, considerable research on automatic video surveillance is going on. The major applications of automatic abnormal event detection and recognition in surveillance scenes are building security, traffic analysis, video monitoring etc.

In this research, a crowded environment is considered for surveillance scene, and the focus is to detect an abnormal event in a crowd scene. A crowd is a unique group of individuals, or something involves a community or society [1]. Crowd scenes can be categorized into two types: structured crowd scenes and unstructured crowd scenes. The term structured crowd scene can be described as the crowd moving coherently in a common direction, and motion direction does not vary with time. Each spatial location of the scene supports only one dominant crowd behaviour over time, for example, marathon race, queues of people and traffic on the road. Meanwhile, the term unstructured crowd scene represents the random crowd motion; different participants move in various directions at different time. Each spatial location supports multi-model crowd behaviour, for example, people walking on a zebra crossing in the opposite direction, exhibition, railway station and airport. An immense amount of research is done and going on for crowd analysis in the computer

vision field because of its applications like Crowd Management, Public Space Design, Virtual Environments, Visual Surveillance and Intelligent Environments. Crowd analysis can be represented with four modules: People detection and tracking, People counting, Behavior analysis and Abnormality detection. Figure 1.1 shows the amount of research work done in each module. Abnormal event detection is considered as one module of crowd analysis. Zitouni et al. [2] have done a comparative study of the research work done for the different modules of crowd analysis and reported that 30% of the research work of complete crowd analysis is done in the abnormality detection module.

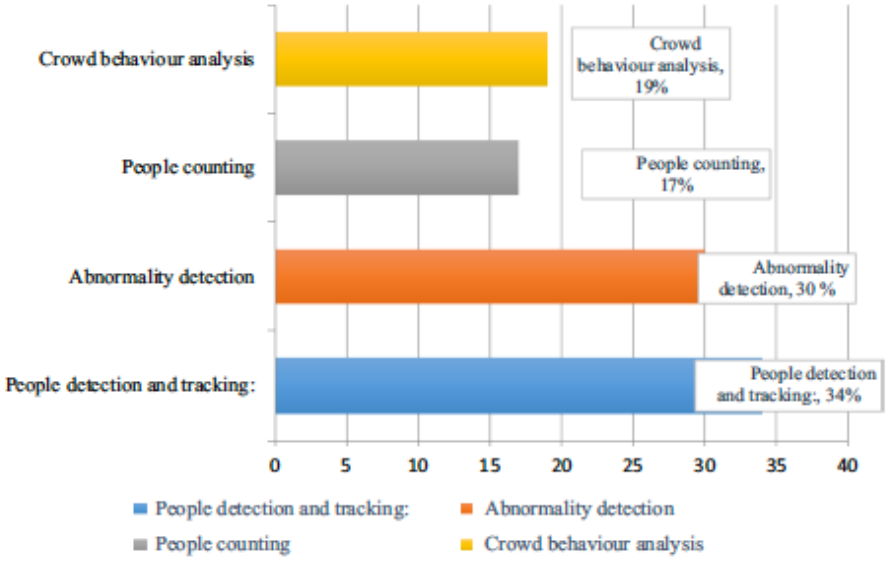



**FIGURE 1.1 Crowd analysis modules [3]**

Automatic abnormal event detection is challenging because the definition of abnormal is subjective or context-dependent. The event which occurs rarely can be considered as an abnormal event. Abnormality detection in a crowd scene is the problem of intricate sequential visual patterns' recognition, as the crowd scene contains occlusion and clutter.

Different researchers have used supervised and unsupervised learning based approaches. It is impossible to generate labels of all types of abnormal behaviours, so general-purpose abnormality detection may not be possible using supervised learning. However, as per requirements at a particular place, it is possible to generate labels of abnormal events and supervised learning can be used. For example, only pedestrians are allowed in some areas, so vehicle entry is abnormal. Access without making a payment is considered deviant behaviour at some paid entry points. Robbery can be viewed as an abnormal event at some

places like banks or shops. For surveillance cameras positioned on the road, the events like suddenly running, fighting, accident, crowd formation can be considered abnormal. In an examination hall, talking or copying the other student's answer sheet is abnormal behaviour. So, as per the situation, it is possible to generate labels of abnormal events and supervised learning can be used. For an unsupervised learning based approach, only normal event videos are used in training. If any abnormal event exists in the test videos, it can be detected based on a logic that if it differs from the training video, it contains an abnormal event. But, any new normal event in test data, which is not available in the training set, can also be considered abnormal. An unsupervised learning-based approach can be applied in any real-world situation where labels are not available. As both supervised and unsupervised techniques have their limitations, both types of approaches are proposed in this research, giving a comparable performance with state-of-the-art methods.

## 1.2 Motivation

Currently, abnormal event detection and classification is an open research area. A fully automatic system is not available to detect and recognize various abnormal events in surveillance videos specially in crowded environment. Many real-world benchmark datasets are publicly available for research. CCTV (Closed-circuit television) cameras are available at almost all places, and video surveillance is a basic need to consider the security of people. The generation of a tremendous amount of surveillance videos and increasing concern about security and safety is the primary source of inspiration to work on automatic detection and classification of abnormal events in surveillance scenes.

## 1.3 Objective and Scope of Work

This work intends to implement various methods to detect and classify anomalous events in surveillance scenes. The research work proposes to achieve the following objectives.

- To propose the approaches based on various handcrafted features and machine learning algorithms for abnormal event detection.
- To propose the efficient supervised deep learning architecture for abnormal event detection and classification.

- To propose unsupervised deep learning approaches for abnormal event detection.

The scope of the research work is as follows.

- This research work is done to detect and classify abnormal events in a crowded environment.
- In this work, supervised and unsupervised learning based approaches are proposed.
- This research work is carried out on six different benchmark publicly available datasets.

## 1.4 Dataset Description

In this work, six publicly available challenging datasets are used, namely UMN, UCSDPed1, Violent Flows, Subway Entrance, Airport-WrongDir and UCFCrime2Local. In all these datasets, ground truth and a sufficient amount of abnormal images are available. The UMN [4] dataset is developed by University of Minnesota. It contains two outdoor and one indoor video samples with 320 × 240 pixels image resolution. Each video consists of an initial part of walking as a normal state and ends with sequences of running as an abnormal state.

The UCSDPed1 [5] is developed by University of California and San Diego. It is a real-world dataset that contains 34 training video samples and 36 testing video samples. Both train and test samples have 200 frames of dimension 238 x158. In this dataset, pedestrians walking on the walkway is considered normal behaviour. Commonly occurring anomalies include bikers, skaters, small carts, and people walking across a walkway or in the grass surrounding it.

Violent Flows Dataset [6] is real-world, video footage of crowd violence, along with standard benchmark protocols. It mainly presents the crowd violence behaviour, and most of the scenes are dynamic, which significantly increase the detection difficulty. This dataset contains 246 video clips with 123 violent samples and 123 nonviolent ones. The resolution for video frames is 320 × 240.

The real-world dataset Subway is provided by A. Adam et al. [7]. This dataset contains two videos. One video monitors the entrance gate, which is 1 hour 36 minutes long and the

second monitors the exit gate, which is 43 minutes long. In this research work, the video sample of the entrance gate is used, which contains 384 × 512 pixels image resolution. People moving in the wrong direction and avoiding payment at the entry gate are abnormal events. The number of anomalies is less in this dataset.

The Airport-WrongDir dataset [8] contains one video sample with 300 × 300 pixels image resolution, in which people moving in a specific direction is considered a normal event and anyone moving in the opposite direction is considered an abnormal event. Different researchers do not widely use this dataset, so in this research also, it is used for only one approach.


Normal: Walking | Abnormal: Running | Normal: Pedestrian | Abnormal: Vehicle

**UMN** | **UCSDPed1**

Normal: Nonviolent crowd | Abnormal: Violent crowd | Normal: Making payment at entry | Abnormal: Avoiding payment at entry

**Violent Flows** | **Subway Entrance**

Normal: Moving in specific direction | Abnormal: Moving in opposite direction

**Airport-WrongDir**

**FIGURE 1.2 Normal and abnormal sample images of various datasets**

The UCFCrime2Local dataset [9] is subset of UCFCrime dataset. It contains 300 videos of seven different categories: Arrest, Assault, Burglary, Normal, Robbery, Stealing, and Vandalism. From these 300 videos, 200 videos are of regular activities and 100 videos are of six different categories' abnormal events. This dataset contains a train-test split. The training set includes 141 normal event videos and 69 abnormal event videos. The test set contains 59 normal event videos and 31 abnormal event videos. This dataset contains weakly labelled videos. Video level labelling is available, i.e. video is normal or has an anomaly somewhere. In our research, this dataset is used for abnormal event detection and classification.

Except for UCFCrime2Local dataset, all datasets contain only two categories Normal and Abnormal. Figure 1.2 represents the normal image and abnormal image of each dataset. Figure 1.3 represents the images of seven different categories of the UCFCrime2Local dataset. The implementation is done with MATLAB-2019a, 16 GB RAM, i9 Processor CPU machine.

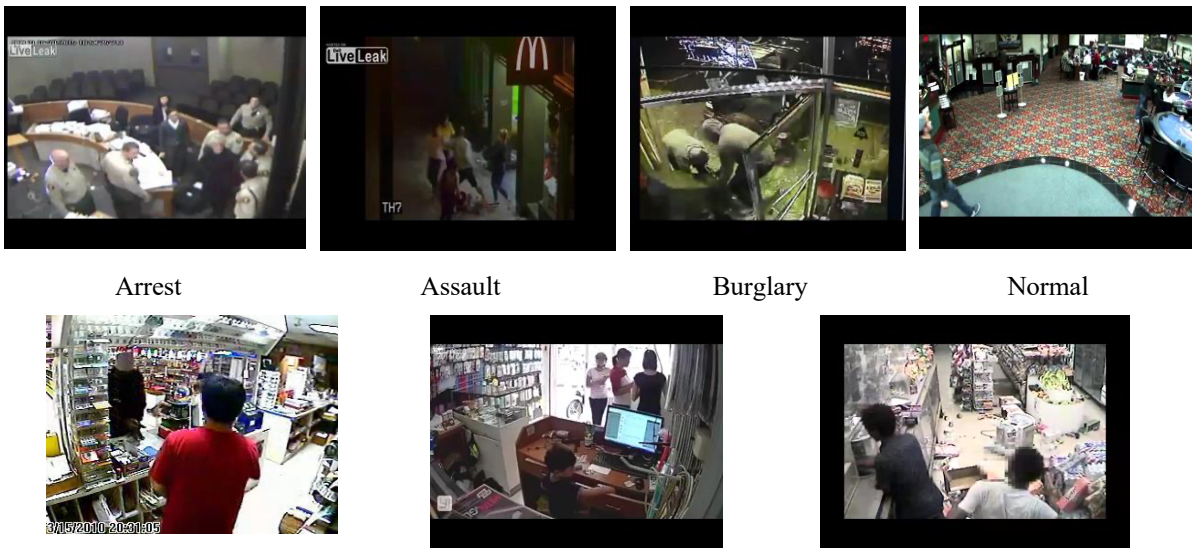



**FIGURE 1.3 Sample images of UCFCrime2Local dataset**

## 1.5 Original Contribution by the Thesis

The objective of the proposed work is to implement efficient methods for abnormal event detection and classification in crowd scenes. In this research, supervised and unsupervised learning based approaches are presented as follows.

**Supervised Machine Learning Based Approaches**

Two methods based on handcrafted features and machine learning algorithms are implemented to detect abnormal events i.e. (1) HOG and Optical flow based approach and (2) Autoencoder and Optical flow based approach.

- **HOG and Optical Flow Based Approach**
  - In this approach, HOG features are extracted for person's shape information and optical flow features are extracted for motion information.
  - Neural Network and SVM are used as classifiers.
  - Experiments are done using only HOG, only optical flow features and a combination of both.
  - The combined features give higher accuracy compared to the individual one using both classifiers.
  - The achieved accuracy is 94.48% using Neural Network and 99.35% using SVM classifier on UMN dataset.

- **Autoencoder and Optical Flow Based Approach**
  - As encoder component of autoencoder extracts meaningful features from an image, in this approach features extracted by autoencoder are used with optical flow features.
  - Neural Network is used as a classifier.
  - Experiments are done using only features extracted by autoencoder, only optical flow features and a combination of both.
  - The combined features give higher performance compared to the individual one. The achieved accuracy is 100 % and AUC value is 1 using UMN dataset.

**Supervised Deep Learning Based Approaches**

Three methods based on supervised deep learning algorithms are implemented i.e. Stacked Autoencoder based approach, CNN based approach and CNN-LSTM based approach. Among these three approaches CNN and CNN-LSTM based approaches are also applied

for classification of various events.

- **Stacked Autoencoder Based Approach**
  - This approach works on spatial domain. The encoder component of autoencoder is used for feature extraction.
  - In the proposed architecture, two encoders are stacked with softmax layer and back propagation is applied for fine tuning.
  - This approach gives 96.9%, 98.6%, 93.1%, 99.6% and 81.4% accuracy on UMN, UCSDped1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets, respectively.

- **CNN Based Approach**
  - The pre-trained image classification network GoogLeNet is retrained and evaluated using UMN dataset. The achieved accuracy is 86.21% and AUC value is 0.9487.
  - A small CNN Architecture1 consisting of 14 layers has been proposed which gives good performance on small and simple datasets. It gives 99.64% accuracy and 0.99 AUC value using UMN dataset and 97.28% accuracy and 0.98 AUC value using Airport-WrongDir dataset.
  - The other CNN Architecture2 consisting of 20 layers has been proposed which gives excellent performance on various complex datasets with different partitions of data. It gives 99.5%, 99.89%, 99.86%, 99.99% and 74.38% using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively.
  - SVM is trained with deep features extracted using the proposed CNN Architecture2 which also gives excellent performance on various complex datasets with different partitions of data. It gives 99.43%, 99.89%, 99.95%, 99.99% and 71.16% using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively.
  - The proposed CNN Architecture2 is applied on combined dataset which includes all images of UMN, UCSDPed1, Violent Flows and Subway Entrance. It gives 99.79% accuracy and AUC value 1.
  - The proposed CNN Architecture2 is applied for classification of various events using complex weakly labelled UCFCrime2Local dataset and

provides adequate AUC value for each class with various partitions of data. It gives 71.79% and 70.20% classification accuracy using CNN and SVM classifiers respectively.

- **CNN-LSTM Based Approach**
  - To consider temporal domain with spatial domain, a CNN-LSTM integrated architecture based approach is proposed for event detection and classification.
  - This architecture is formed using the proposed CNN Architecture2.
  - This approach gives 100%, 100%, 98%, 96.77% and 73.33% accuracy using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively for abnormal event detection.
  - The achieved overall classification accuracy is 73.86% using UCSFCrime2Local dataset. This approach gives higher AUC values for various events compared to CNN based approach.

**Unsupervised Deep Learning Based Approaches**

The labelled data are not available every time for each type of event. In this situation, unsupervised learning based approach is required to be applied to detect anomalous behaviour. In this research work two such architectures are proposed, one is convolutional autoencoder and the other is LSTM autoencoder.

- **Convolutional Autoencoder Based Approach**
  - This approach works on spatial domain.
  - In this method, a simple convolutional autoencoder architecture consisting of 10 layers has been proposed.
  - This approach gives 0.86, 0.70, 0.66, 0.82 and 0.59 AUC values for UMN, UCSDPed1, Violent-Flow, Subway Entrance and UCFCrime2Local datasets respectively.

- **LSTM Autoencoder Based Approach**
  - This approach works on spatiotemporal domain.
  - In this method, a simple LSTM autoencoder architecture consisting of 9

layers has been proposed.

- The LSTM autoencoder reconstructs the input feature sequence. The pre-trained GoogLeNet Architecture and the proposed CNN Architecture2 are used to extract feature sequences from input video.
- This approach gives 1, 0.75, 0.48, 0.83 and 0.52 AUC values for UMN, UCSDPed1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets respectively with the proposed CNN Architecture2.
- This approach gives 1, 0.93, 0.5, 0.82 and 0.63 AUC values for UMN, UCSDPed1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets respectively with the GoogLeNet architecture.

Based on all the proposed methods, three research papers are published in different reputed peer-reviewed international journals, as mentioned at the end of the thesis.

## 1.6 Organization of the Thesis

The thesis is organized as follows. Chapter 2 presents a literature review of abnormal event detection and classification in surveillance crowd scenes. In this chapter, various methods implemented by different researchers based on machine learning and deep learning algorithms are discussed.

Chapter 3 presents the implemented methods based on handcrafted features and machine learning algorithms for abnormal event detection.

Chapter 4 presents the developed supervised deep learning architectures for abnormal event detection and classification. Implementation details and experimental results using the various partitions of different datasets are discussed in this chapter.

Chapter 5 presents the developed unsupervised deep learning architectures for abnormal event detection. The comparison of all the proposed approaches with state-of-the-art methods on a specific dataset is also shown in this chapter.

Chapter 6 presents the conclusion and future scope of this thesis.

# CHAPTER - 2

# Literature Review

This chapter describes existing methods for abnormal event detection and recognition, which represent all crucial work proposed in this area. The various performance evaluation metrics used to evaluate the proposed approaches are also discussed here.

To detect any abnormality, feature extraction from images is a necessary step. Feature descriptors can be classified as handcrafted features and deep learning based features. Handcrafted features relate to properties derived using various algorithms using the information present in the image itself like edge, corner, HOG, optical flow etc. These features are extracted and given as input to any machine learning algorithm. Deep learning based features are the feature descriptors extracted by the deep learning model. The images are provided as input, and the deep learning model automatically extracts features as per tuned parameters and hyperparameters. Various handcrafted features based and deep learning based approaches proposed by different researchers are as follows.

## 2.1 Handcrafted Features Based Approaches

Mehran R et al. [10] have used the social force model to estimate interaction force between moving individuals. Then, this force is mapped into the image plane to get force flow for each pixel in every frame. Spatiotemporal volumes of force flow is used to represent normal crowd behaviour. They have classified frames as normal and abnormal using a bag of words approach. Generally, abnormal activity contains higher social interaction, which is detected by locating high force flow. The authors have achieved an AUC value of 0.96 on the UMN dataset. The authors have also experimented with pure optical flow, for which the achieved AUC value was 0.84.

S. Wu et al. [11] tried to find out crowd escape behaviour in surveillance scenes to detect an anomalous situation in time. They used potential destinations and divergent centres to characterize the presence and absence of escape events. They have proposed a Bayesian model for crowd behaviour recognition.

S. Wu et al. [12] have used particle trajectories for modelling arbitrarily complicated crowd flows. They have introduced chaotic dynamics to represent crowd motion. This chaotic feature set is used to train the probabilistic model. The reported AUC value for this method is 0.99 on the UMN dataset.

CemDirekoglu et al. [13] have considered the angle difference between the optical flow vectors in the current frame and the former frame at each pixel location which improves accuracy over pure optical flow features. This angle difference value is combined with pure optical flow magnitude. They have used one-class SVM to learn normal behaviour. If the test sample crucially varies from normal behaviour, it is considered an abnormal one. The reported accuracy is 96.46% on the UMN dataset. They have also done experiments with Bayesian Model (BM) [11], Sparse Reconstruction Cost (SRC) [14], Chaotic Invariants (CI) [12], the Social Force model (SF) [10], and the Force Field model (FF) [15].

D. Chen and P. Huang [15] used optical flow to cluster human crowds into groups in an unsupervised manner using the method of Adjacency-Matrix based Clustering (AMC). The behaviours of cluster crowd with attributes - orientation, position and crowd size are characterized by a force field model. They have achieved 94% accuracy on the UMN dataset.

Y. Cong et al. [14] have proposed the Sparse Reconstruction Cost (SRC) over the standard dictionary to measure the normalness of the testing sample. They introduced a large scale dictionary selection method using the concept of traditional convex optimization. They achieved 0.487 AUC using the UCSDPed1 dataset. Halbe M. et al. [17] used optical flow for the computation of crowd motion energy. The crowd motion energy is further modified by Crowd Motion Intensity (CMI). The peaks in the CMI characteristics are indicators of abnormal activity. In this method, the proper threshold plays a significant role in abnormal activity detection.

Hu et al. [18] have used contextual gradients between two local regions and then have constructed a Histogram of Oriented Contextual Gradient (HOCG) descriptor for abnormal event detection based on the contextual gradients. The compositional context of events is described efficiently because of the representation of the distribution of contextual gradients of subregions in different directions by the HOCG descriptor. Authors have employed online dictionary learning and a sparse reconstruction framework for abnormality detection. The event for which SRC is higher than a specific threshold is considered abnormal. As learning is online, the threshold is set based on the best previous performance of the method. The attained AUC value is 0.993 using the UMN dataset.

In the method proposed by Xuguang Zhang et al. [19], the modification of energy level distribution in an image is considered a critical factor to detect abnormal crowd behaviour. They have used optical flow to extract velocities of pixels. Motion foreground is extracted using a crowd motion segmentation method based on flow field texture representation. Linear interpolation is computed to find the distance between the camera and the pedestrian's foreground. In this method, the crowd behaviour is analyzed according to the change of the consistency, entropy and contrast - three descriptors for co-occurrence matrix. Here Threshold judgement of three descriptors is critical to get an accurate result.

Yong Yin et al. [20] generated a feature vector of crowd density and crowd dynamic information. They utilized the Local Binary Pattern Co-occurrence Matrix (LBPCM) for crowd density estimation. They have adopted high accuracy optical flow Histograms of the Orientation of Interaction Force (HOIF) to extract the crowd dynamic information. Then SVM is used to detect an abnormal event. The attained AUC value is 0.99 for the UMN dataset.

Y. Benabbas et al. [27] have used an optical flow field with block clustering to identify an unusual event in a crowd. Gao, Y. et al. [28] have used Violent Flows (ViF) and Oriented Violent Flows (OViF) to detect violence in a crowd scene. Oriented Violent Flows consider modification in motion information. The combined features are used with AdaBoost and linear SVM classifiers. The achieved AUC value is 0.9484 for the Violent Flows dataset.

Except for these features, different authors have used various traditional features like motion structure, Spatiotemporal features, Mixture of Probabilistic Principal Component Analyzers (MPPCA), Mixture of Dynamic Textures (MDT), Local Statistical Aggregates, Sparse Cuboids (SC), Local Trinary Patterns (LTP), Simplified Histogram of Oriented Tracklets (SHOT), Interaction Force (IF), Improved Fisher Vectors, Gaussian Process Regression (GPR) etc.

## 2.2 Deep Learning Based Approaches

Tahjid Ashfaque Mostafa et al. [26] have analyzed local spatial-temporal motion patterns in video frames. They have used a motion heat map to find the region of interest. After identifying the motion structure, different classifiers are used like SVM, Naive Bayes, Neural Network and CNN. The CNN classifier gives a good result to detect anomalies in a crowd scene. The achieved accuracy is 96.74% and 94.28% on UMN and UCSD datasets, respectively.

Ravanbakhsh et al. [25] employed a Fully Convolutional Network as a pre-trained model and plugged an effective binary quantization layer as the final layer to the net. The authors captured temporal CNN patterns to denote motion in a crowd. Sabokrou et al. [29] have used transfer learning. The authors have used pre-trained CNN, i.e. Alexnet.

Feng et al. [30] have extracted appearance and motion features using the PCANet. The deep Gaussian Mixture Model (GMM) is constructed with observed regular events. The authors have got a 0.925 AUC value for the UCSDPed1 dataset. Zhou et al. [31] have used a Spatial-temporal CNN to detect features like appearance and motion from spatial and temporal dimensions. The achieved AUC values are 0.9963 and 0.927 using UMN and Subway Entrance datasets, respectively.

Smeureanu, S et al. [32] have used the VGG pre-trained CNN model to extract features, and then one class SVM classifier is used to learn normal event patterns. They have got 0.85 AUC value for the UMN dataset. In [33], the authors have proposed the Deep One Class (DOC) model in which CNN is used to extract features, and One-class SVM is used to learn decision function for abnormal event detection from the given normal event

images. They have done experiments using SVM with linear kernel and RBF kernel. With linear kernel, the achieved AUC value is 0.808, and for RBF kernel, it is 0.914 using the UCSDPed1 dataset. Hinami et al. [34] have used a fast R-CNN model to detect an abnormal event. R-CNN is a Region-based Convolutional Neural Network in which region proposals are used during training and testing, so algorithm's performance slows down significantly.

In [35], authors have proposed a two-stream R-convolutional Variational Autoencoder (R-convVAE). They have done experiments with unsupervised learning-based approaches like a convolutional autoencoder, convolutional variational autoencoder, recurrent convolutional autoencoder, recurrent convolutional variational autoencoder. They have used appearance and motion features to describe the probabilistic distribution. The achieved AUC value using two-stream R-ConvVAE is 0.75 for UCSDPed1 dataset. Yong Shean Chong et al. [36] proposed a spatiotemporal autoencoder that includes two main components, one for spatial feature representation and the other for learning the temporal evolution of the spatial features. The achieved AUC values are 0.899 and 0.847 for UCSDPed1 and Subway Entrance datasets respectively.

Vu, H et al. [37] have used denoising Autoencoder and Conditional Generative Adversarial Networks to detect an anomaly from video. Federico Landi et al.[9] have used a 3D convolutional network with a regression network to find out anomaly scores. They have done two experiments. In the first one, they have used the whole frame as input. In the second experiment, the authors extracted a spatiotemporal tube that locates abnormal events and is given as input. Here UCFCrime2Local dataset is used. For the whole frame, the obtained AUC value is 0.5612, and for the spatiotemporal tube, the obtained AUC value is 0.7413.

In [38], authors have developed a UCFCrime challenging multiclass dataset and experimented with two existing approaches for events classification. In the first one, they have used a 3D convolutional network for feature extraction and the nearest neighbour classifier for classification. The achieved accuracy for classification is only 23%. In the second one, they have used Tube Convolutional Neural Network (TCNN), which includes a tube of interest pooling layer. It combines all clips' features and generates one feature vector for one video. The reported classification accuracy is 28.4%.

## 2.3 Performance Evaluation Metrics

In this research work, the performance of the proposed approaches is measured by confusion matrix, accuracy and Area Under the Curve (AUC). Different researchers have used accuracy or AUC value to represent the performance of various methods in literature.

### 2.3.1 Confusion Matrix

The confusion matrix is one of the widely used metrics for the performance evaluation of a classifier. The confusion matrix represents the ability of the classifier to correctly recognize data samples of different classes. The confusion matrix for the binary classification problem is given in Table 2.1.

**TABLE 2.1 Confusion matrix for binary classification**

| | | **Predicted class** | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **True class** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

The meaning of True Positive is that the tested sample data is a part of positive class and predicated as positive, while the term False Positive means that the sample data is a part of the negative class, but it is classified as positive. The term True Negative indicates the sample is taken from negative class and classified as negative. Similarly, the term False Negative implies the sample is a part of the positive class but classified as Negative. With many categories, it is very problematic to visualize the entire confusion matrix. Still, the details provided by it can be used for another performance measurement parameter like accuracy that better represents the performance of the classifier model.

### 2.3.2 Accuracy

Classification accuracy summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions. The classification accuracy can be calculated from the confusion matrix as follows.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$$

Where $TP$ = True Positive

$TN$ = True Negative

$FP$ = False Positive

$FN$ = False Negative

### 2.3.3 Area Under the Curve

The Receiver Operating Characteristic (ROC) curve is a graph showing the performance of a classification model at all classification thresholds. The ROC curve represents the relation between the True Positive Rate (TPR) and the False Positive Rate (FPR) of the test dataset. True Positive Rate and False Positive Rate can be calculated as follows.

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{F P+TN}}$$

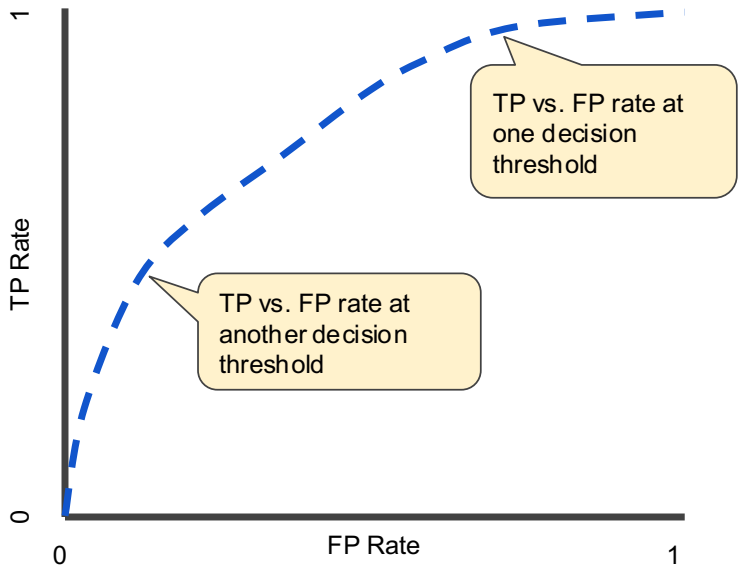



**FIGURE 2.1. TP vs FP rate at different classification thresholds [51]**

The ROC curve plots TPR vs FPR at different classification thresholds. If the classification threshold is low, more data items are classified as positive and false positives and true positives increase. Figure 2.1 represents a typical ROC curve. The curve closer to the top-left corner indicates a better performance. The curve closer to the 45-degree diagonal of the ROC space represents poor performance.

“Area Under the Curve" is shortly known as AUC. Figure 2.2 shows AUC for the ROC curve shown in Figure 2.1. It measures the entire two-dimensional area under the whole ROC curve from (0,0) to (1,1). It is the performance metric whose value lies between 0 and 1. A value less than 0.5 is the worst, and a value near 1 indicates the best performance. AUC value 0.5 suggests that the classifier cannot differentiate between positive and negative classes. AUC is widely used for performance evaluation of classifiers because of its following characteristics.

- AUC is scale-invariant. It measures how effectively predictions are categorized rather than their absolute values.
- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is set.

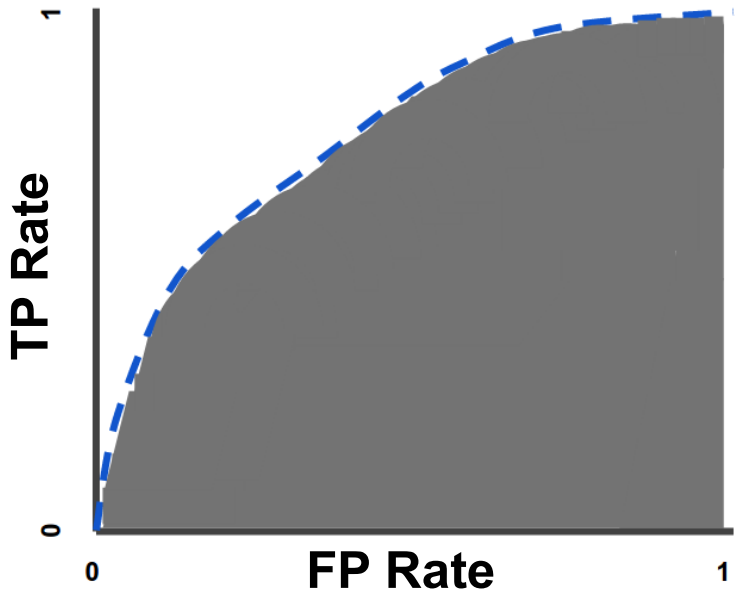



**FIGURE 2.2. Area under the ROC curve [51]**

## 2.4 Definition of the Problem

As analysis done so far of literature review, it is observed that detecting abnormal event from the video is a very challenging task because of the subjective definition of abnormality. Specifically, it becomes more complex when the crowded environment is considered because of severe occlusion and clutter. Different researchers use both supervised and unsupervised learning based approaches, but major work is done using unsupervised learning based approaches which fail when a new normal event occurs. The process for abnormal event detection still needs improvement. Existing deep learning techniques give poor performance for classifying events using challenging dataset. Hence, it is essential to develop novel algorithms for detecting and classifying abnormal events in surveillance scenes with improved performance.

# CHAPTER - 3

# Approaches Based on Machine Learning Algorithms

As discussed in the literature review, handcrafted features and deep learning features can be used to detect an abnormal event in a surveillance scene. In this chapter, two supervised learning based approaches are presented. In the first approach, Histogram of Oriented Gradients (HOG) and Optical flow handcrafted features are used with Neural Network and Support Vector Machine classifiers. To use machine learning algorithms, it is required to extract meaningful features from images, so that accurate classification can be possible. For abnormal event detection in a surveillance scene, a person and his movement information play an important role. Here, HOG features are used to detect pedestrians, optical flow is used to represent motion, and both are combined to detect abnormal events. In the second one, optical flow and features extracted by autoencoder are used with Neural Network classifier because the encoder component of autoencoder extracts essential features from the image precisely. Optical flow represents apparent motion between two consecutive frames, and it can be applied to the continuous scene, so experiments of both approaches are done using only the UMN dataset, which contains three different continuous scenes.

## 3.1 HOG and Optical Flow Based Approach

Figure 3.1 represents the block diagram of the proposed approach based on Histogram of Oriented Gradients and optical flow. A detailed explanation of every step of the proposed algorithm is given in the following subsections.

### 3.1.1 Foreground Extraction

First, RGB colour images (video frames) are converted into grayscale images. Then the background is generated by taking an average of all frames of the video. This background image is subtracted from each frame for foreground extraction. Figure 3.2 represents the extracted foreground of a sample image of each of the three scenes of the UMN dataset.

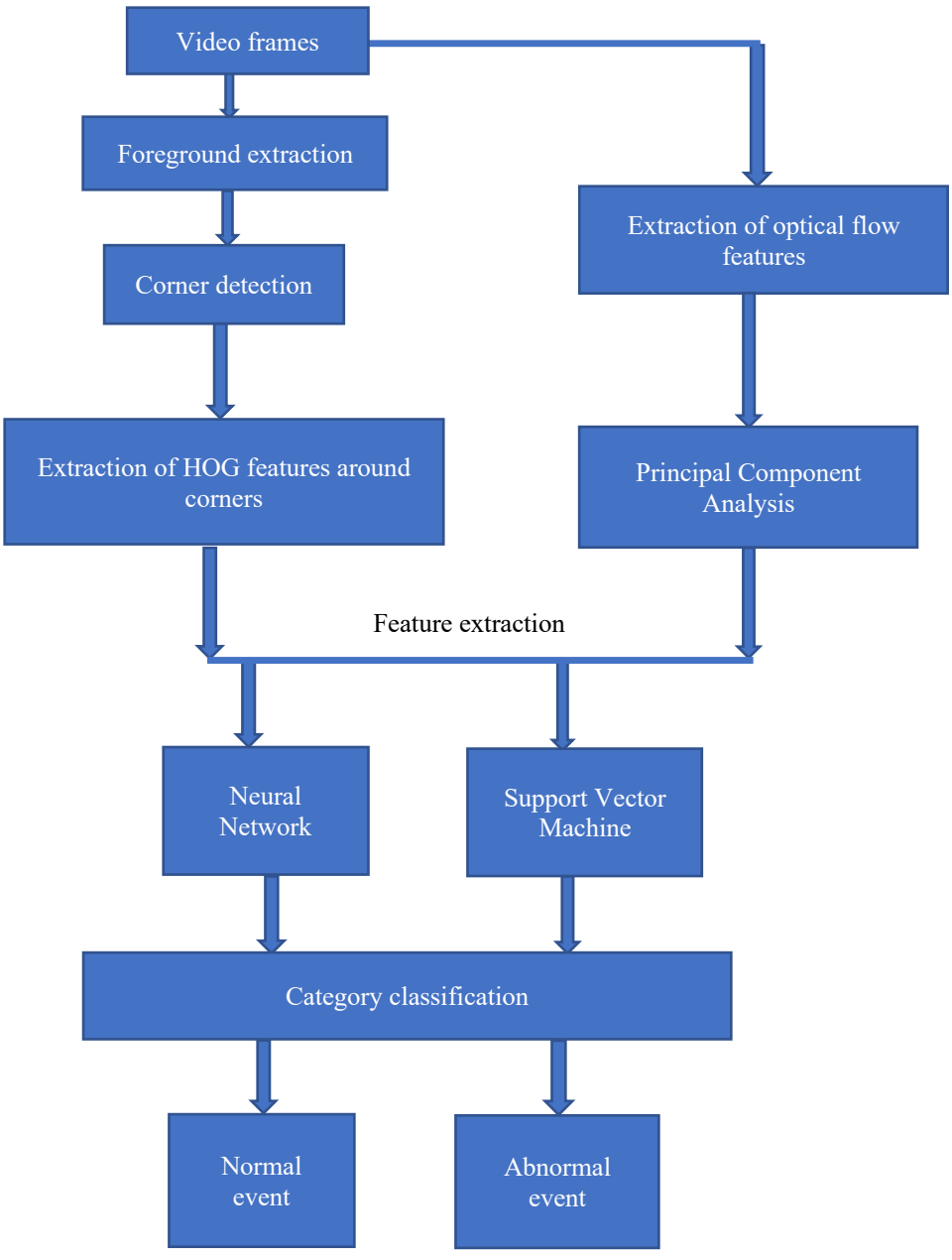
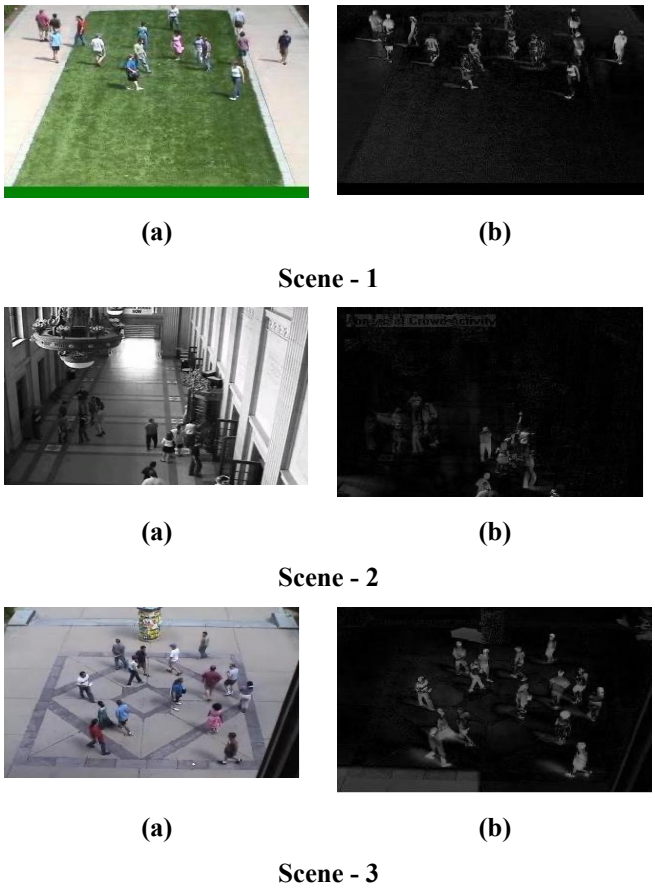



**FIGURE 3.1 HOG and optical flow based approach**



**(a)** **(b)**

**Scene - 1**

**(a)** **(b)**

**Scene - 2**

**(a)** **(b)**

**Scene - 3**

**FIGURE 3.2 (a) Input frame (b) Extracted foreground**

### 3.1.2 Corner Detection

For corner detection, Harris corner detection [55] method is used. The Harris corner detector is a mathematical operator that finds features in an image. The Harris corner detector is a mathematical way of determining which windows produce significant variations when moved in any direction. This is translated mathematically by following equation 3.1

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \qquad (3.1)$$

Where E = difference between the original and the moved window

u = window's displacement in the x-direction

v = window's displacement in the y-direction

w(x, y) = window at position (x, y), which acts as a mask

I = intensity of the image at a position (x, y)

I(x + u, y + v) = intensity of the moved window
I(x, y) = intensity of the original image

The interest is to find windows that produce a large E value, so the following term should be maximized.

$$\sum_{x,y}[I(x+u, y+v) - I(x,y)]^2$$

This term is expanded using the Taylor series:

$$\sum_{x,y}[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$\text{So, } E(u,v) \approx \sum_{x,y} u^2 I_x{}^2 + 2uvI_xI_y + v^2 I_y{}^2 \qquad (3.2)$$

Now this messy equation 3.2 can be tucked up into a neat little matrix form like following equation 3.3.

$$E(u,v) \approx [u,v] \left( \sum \begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \qquad (3.3)$$

Now, the summed matrix is renamed with M:

$$M = \sum \begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix} \qquad (3.4)$$

$$E(u,v) \approx [u\ v]\ M \begin{bmatrix} u \\ v \end{bmatrix} \qquad (3.5)$$

The matrix's eigenvalues can help to determine the suitability of a window. Score, R, is calculated for each window: $R = \det M - k(\mathrm{tr}M)^2$ , Where det M is the determinant of a matrix, trM is the trace of a matrix, k is experience value, $\lambda_1$ $\lambda_2$ are the two characteristic values of the matrix, which reflect the surface curvature of two principal axes in the image. Therefore, $\det M = \lambda_1\lambda_2$, $\mathrm{tr}M = \lambda_1+\lambda_2$. All windows with a score R greater than a certain value are corners. They are good tracking points. Figure 3.3 represents detected corners in the extracted foreground.

**Scene - 1** **Scene - 2** **Scene - 3**

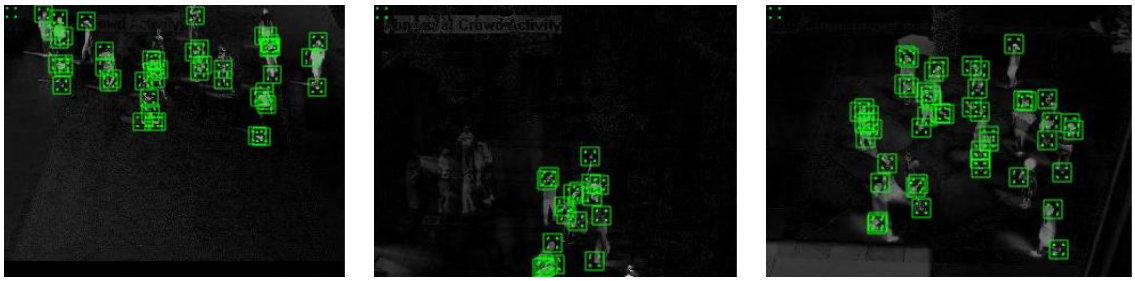
**FIGURE 3.3 Detected corners**

### 3.1.3 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) is a feature descriptor used to detect objects from an image. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image detection window or Region of Interest (ROI). By using HOG descriptor, local object appearance and shape information within an image can be defined by the distribution of intensity gradients or edge direction. Implementation of the HOG descriptor algorithm is as follows [56][59]:

- First of all, the input image is divided into 8×8 cells.
- The gradient of this patch contains two values per pixel, magnitude and direction. So, the image patch contains 64 values of magnitude and 64 values of direction.
- Now the direction is represented by 0 to 180 degrees. By a span of 20, the direction is represented using nine bins. Magnitude values are inserted into nine direction bins. So, nine bins histogram is generated for one 8*8 cell, which can be stored as a 9×1 array.
- In the previous step, a histogram is created based on the gradients of an image that are sensitive to lighting. Ideally, the descriptor should be independent of lighting variations. Therefore, the histogram should be normalized to not be affected by lighting variations. Now four 8×8 patches are combined, and Normalization is done over a block of 16×16. A 16×16 block has four histograms. These four histograms are concatenated to form a 36 x 1 vector, and these 36 x 1 elements are normalized by dividing every 36 value by its norm. Then window is moved by 8 pixels, and the 36×1 vector is calculated over this window. This process is repeated.
- To calculate the final feature vector for the whole image, all the vectors of size 36×1 are concatenated into 1xN vector where N is HOG features length.

In this approach, to represent HOG features 16x16 block is moved 15 horizontal and 15 vertical positions, So the number of HOG features for a single image is 15x15x36=8100. Instead of finding HOG features of the whole image, features are extracted around corners, so feature vector length is small and execution time can be reduced. Figure 3.4 represents the HOG features around the detected corners.



**Scene - 1** **Scene - 2** **Scene - 3**

**FIGURE 3.4 HOG features around corners**

### 3.1.4 Optical Flow

Optical flow is the distribution of the apparent velocities of objects in an image. By estimating optical flow between video frames, the velocities of objects in the video can be measured. The optical flow methods try to calculate the motion between two video frames, taken at time t and t + Δt. There are many methods available in the literature to compute optical flow. Among them, two are widely used. One is the Horn-Schunck [57] method, and the other is the Lucas-Kanade [84] method. In this approach, the Horn–Schunck method is used for optical flow computation because it is a global method and yields dense flow fields.

#### 3.1.4.1 Optical Flow Equation

Three following significant constraints are considered for optical flow estimation.

- Brightness constancy: projection of the same point looks the same in every frame.
- Small motion: points do not move very far.
- Spatial coherence: points move like their neighbours.

**2D Motion Equation**

Assume I(x, y, t) is the centre pixel in (n × n) neighbourhood and moves by $\delta_x$, $\delta_y$, in time $\delta_t$ to (x + $\delta_x$, y + $\delta_y$, t + $\delta_t$). Since I(x, y, t) and I(x + $\delta_x$, y + $\delta_y$, t + $\delta_t$) are the images of the same point so,

$$I(x, y, t) = I(x + \delta_x, y + \delta_y, t + \delta_t) \quad (3.6)$$

The provided local translations $\delta_x$, $\delta_y$, $\delta_t$ are not too large, so equation 3.6 can be expanded by Taylor Series as follows.

$$I(x + \delta_x, y + \delta_y, t+ \delta_t) = I(x, y, t) + \frac{\partial I}{\partial X}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t + \text{Higher Order Terms} \quad (3.7)$$

Higher-Order Terms are small terms so that they can be ignored. Using the equations 3.6 and 3.7, the following equations can be obtained:

$$\frac{\partial I}{\partial X}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t = 0 \quad (3.8)$$

$$\frac{\partial I}{\partial X}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \quad (3.9)$$

and finally,

$$\frac{\partial I}{\partial X}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \quad (3.10)$$

Where $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$ are the x and y components of optical flow and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial t}$ are image intensity derivatives at x, y and t, respectively. These partial derivatives can be written as: $I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$ and $I_t = \frac{\partial I}{\partial t}$

So, the optical flow equation can be written like following:

$$I_x u + I_y v + I_t = 0 \quad (3.11)$$

Where, ($I_x$, $I_y$) = Image brightness derivative and (u, v) = Optical flow or image velocity at pixel (x, y) at time t.

### 3.1.4.2 Horn–Schunck Method

Horn and Schunk have represented optical flow by the following expression.

$$\iint\{(I_x u + I_y v + I_t)^2 + \lambda (u_x^2 + u_y^2 + v_x^2 + v_y^2)\}\, dx\, dy$$

The above expression represents brightness constancy and smoothness constraint. Double integration represents optical flow for each pixel in a 2D image. To minimize the term, the second-order derivation is used as follows.

$$(I_x u + I_y v + I_t)\, I_x + \lambda (\nabla^2 u) = 0$$

$$(I_x u + I_y v + I_t) I_y + \lambda (\nabla^2 v) = 0 \quad \text{where}$$

$$\nabla^2 u = u_{xx} + u_{yy}$$

$$\nabla^2 v = v_{xx} + v_{yy}$$

The second-order derivative can be represented by Laplacian so $\nabla^2 u$ and $\nabla^2 v$ are replaced with $u - u_{av}$ and $v - v_{av}$, respectively.

$$(I_x u + I_y v + I_t)\, I_x + \lambda (u - u_{av}) = 0$$

$$(I_x u + I_y v + I_t)\, I_y + \lambda (v - v_{av}) = 0$$

So, finally u and v can be computed as

$$u = u_{av} - I_x P/D$$

$$v = v_{av} - I_y P/D$$

$$\text{Where } P = I_x u_{av} + I_y v_{av} + I_t$$

$$D = \lambda + I_x^2 + I_y^2$$

Figure 3.5 represents computed optical flow for one frame of each scene of the UMN dataset.

**Scene - 1** **Scene - 2** **Scene - 3**

**FIGURE 3.5 Optical flow representation**

### 3.1.5 Principal Component Analysis

Principal Component Analysis (PCA) [58] is a dimensionality reduction method that reduces the dimensionality of large datasets without significant loss of information. The first step for PCA is to standardize the range of continuous initial variables so that equal contribution of each variable can be considered. This can be done by following equation 3.12.

$$Z = \frac{\text{value} - \text{mean}}{\text{standard deviation}} \qquad (3.12)$$

The equation represents that, from every value, the mean is subtracted and then divided by standard deviation. In the second step, covariance matrix is computed to understand the relationship between variables. It summaries the correlations between all the possible pairs of variables. If any entry in the covariance matrix is positive, then two variables are correlated, i.e. both increase or decrease together. If any entry is negative, two variables are inversely correlated, i.e. one increases when the other decreases. Then, the eigenvectors and eigenvalues of the covariance matrix are computed to identify the principal components. Eigenvectors represent the directions of the axes where there is the most variance, i.e. Principal Components. Eigenvalues are the coefficients attached to eigenvectors, i.e. the amount of variance carried in each principal component.

The principal components are generated in such a way that the first principal component represents the largest possible variance in the dataset. Because of this type of organization, the components with low information can be discarded, and dimensionality can be reduced without losing much information. The optical flow features represent two values, magnitude and direction per pixel, so to use all features for training and testing is very time-consuming. Therefore, PCA is applied to reduce dimensionality so that training and testing become easy.

### 3.1.6 Neural Network and Support Vector Machine

#### 3.1.6.1 Neural Network

An Artificial Neural Network (ANN) is a computational model with interconnected nodes inspired by the biological nervous system. An ANN is configured through a learning process for specific applications, such as pattern recognition or data classification.

An ANN is based on a collection of connected nodes called neurons, as shown in Figure 3.6. Each connection between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then pass it to the other neurons connected to it.

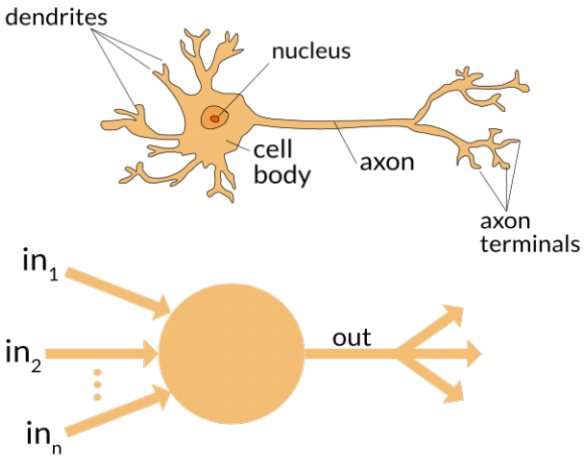



**FIGURE 3.6 Biological neuron and basic ANN architecture [60]**

Learning in biological systems involves adjustments to the connections that exist between the neurons. With their remarkable ability to derive meaning from complicated or imprecise data, neural networks can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

Various ANN architectures have been developed over the years: the simplest one is Feedforward Neural Network (FNN), where each neuron is connected to all the neurons in the previous layer, defining a directed graph without any loops. Figure 3.7 shows the simple FNN with one hidden layer between the input and output layers.

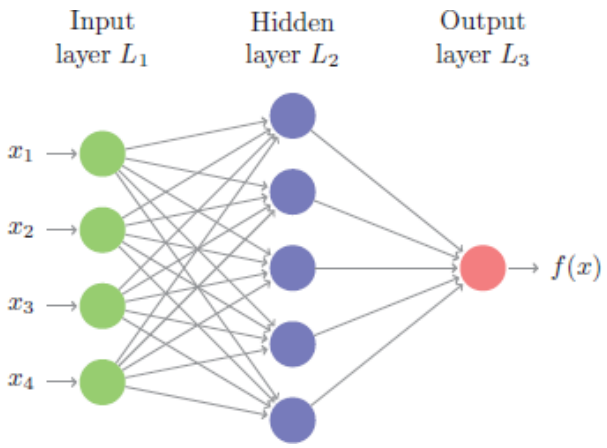



**FIGURE 3.7 Simple feedforward neural network [61]**

ANNs provide an approximation function given by the following equation 3.13 of an arbitrary complex continuous function f, where x and y correspond to input and output data, respectively.

$$y = f(x; \theta) \qquad (3.13)$$

Function f parametrized by a set of coefficients θ, i.e. matrices and biases. Matrices are generally inputs with their weights, and biases represent threshold. The creation of the predictive model thus requires the estimation of the parameters θ that better approximate the desired output; this is achieved by minimizing a cost function defined according to the output layer properties; common choices are Mean Squared Error (MSE) for regression and cross-entropy for classification [62].

### 3.1.6.2 Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm used for classification and regression problems. Each data item is plotted as a point in space. Then, SVM classifies data by finding the best hyperplane that separates all data points.

Figure 3.8 shows the plotting of 2-D data, which is linearly separable. Multiple lines could be drawn, but the best one can be found to minimize classification error. Generalizing to n-dimensional data, the best 'hyperplane' can be found. Hyperplane refers to the decision boundary. SVM approaches this problem by searching for the maximum marginal hyperplane.

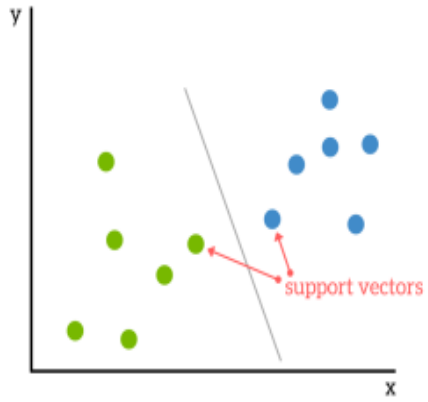



**FIGURE 3.8 Two dimensional linearly separable data [62]**

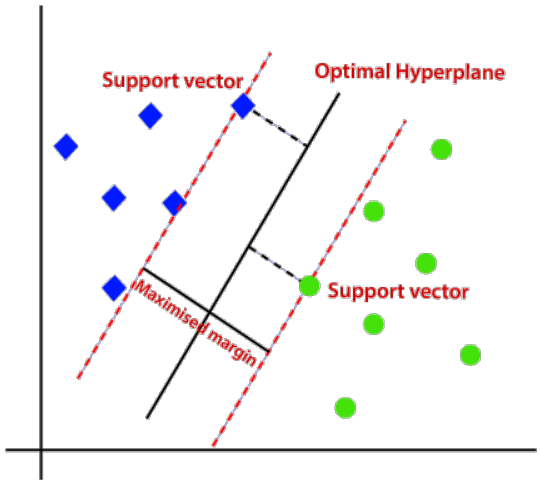



**FIGURE 3.9 Optimal hyperplane [63]**

Figure 3.9 shows the Maximum marginal hyperplane, by which the data can be differentiated. Support vectors are the data points closer to the optimal hyperplane. If there are n training data points $z_i \in R_d$, $i$ = 1,2, . . ., n, which can be divided into two different classes and their respective class labels are given by $yi \in \{-1, 1\}$, i = 1, . . ., n, Generally, a linear support vector machine creates a hyperplane in feature space Z, as represented in the following equation 3.14.

$$w.\, z + b = 0 \qquad (3.14)$$

The research [79] shows that the weights w for the optimal hyperplane in the feature space can be written as some linear combination of support vectors as given in equation 3.15.

$$w = \sum_{\text{support vectors}} \alpha_i z_i \qquad (3.15)$$

The linear decision function I in the feature space will accordingly be of the form given by equation 3.16,

$$I(z) = \text{sign}\left(\sum_{\text{support vectors}} \alpha_i z_i . z + b\right) \qquad (3.16)$$

Equation 3.16 represents the dot-product of support vectors denoted by $z_i$ and vector z in feature space [79]. SVM can also be applied to nonlinear data. SVM uses various kernel functions, which are mathematical operations. Kernel functions map low dimension samples into a high-dimension feature space, such that an optimum hyperplane can be found to deal with the nonlinear data. The various kernel functions like linear, polynomial, radial basis function, and sigmoid are used as per characteristics of data.

In SVM, prediction of unseen data is made by considering only the Support Vectors and hence the presence of outliers in the training set may not influence the generalization accuracy of SVM. It works more effectively in high dimensional spaces. SVM is considered an admirable approach for easily divisible classes. SVM performs well for binary and multiclass both types of classification.

SVM is not appropriate for the classification of large datasets because its training complexity depends on the size of the dataset. SVM doesn't give satisfactory performance in the case of overlapping classes. It is determinant to select optimal values of hyperparameters for effective performance.

### 3.1.7 Experimental Results

Experiments are done on three different scenes of the UMN dataset. In the experiment using a neural network classifier, a feed-forward neural network with one hidden layer containing ten nodes is used. The following Figure 3.10 represents the generated pattern recognition network for scene - 3.

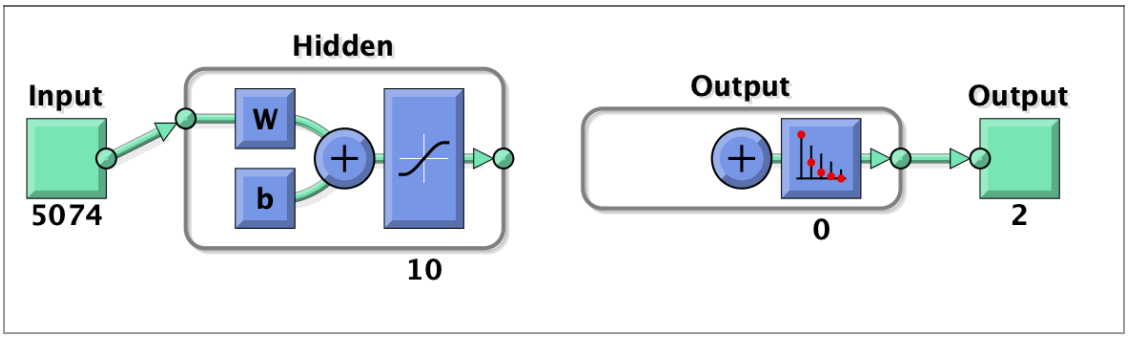



**FIGURE 3.10 Generated pattern recognition network for the UMN dataset scene - 3**

The optical flow represents the magnitude and direction for each pixel. Here, the image size is 128x128, so the number of optical flow features for a single image is 16384+16384=32718. After Applying PCA, the number of optical flow features for single image is 4282. As mentioned in section 3.1.3 the number of HOG features for a single image is 8100. The number of HOG features around detected corners is 792. The total number of features given for training is 4282+792=5074. Table 3.1 and Table 3.2 show experimental results on the UMN dataset using neural network and linear SVM classifiers, respectively with 70% training data and 30% test data. Experiments are done using only optical flow, only HOG features, and a combination of both. Whenever HOG and Optical flow both features are combined the accuracy is improved.

**TABLE 3.1 Experimental results using neural network**

| UMN dataset | Number of images | HOG | Optical flow | HOG + Optical flow |
|---|---|---|---|---|
| Scene1 | 1453 | 90.9 | 98.4 | 98.8 |
| Scene2 | 4144 | 93.24 | 93.81 | 94.13 |
| Scene3 | 2142 | 90.34 | 88.94 | 90.50 |
| Average Accuracy | | 91.49 | 93.72 | 94.48 |

**TABLE 3.2 Experimental results using SVM**

| UMN dataset | Number of images | HOG | Optical flow | HOG + Optical flow |
|---|---|---|---|---|
| Scene1 | 1453 | 79.83 | 84.36 | 98.35 |
| Scene2 | 4144 | 97.83 | 100 | 100 |
| Scene3 | 2142 | 95.79 | 99.84 | 99.69 |
| Average Accuracy | | 91.15 | 94.73 | 99.35 |

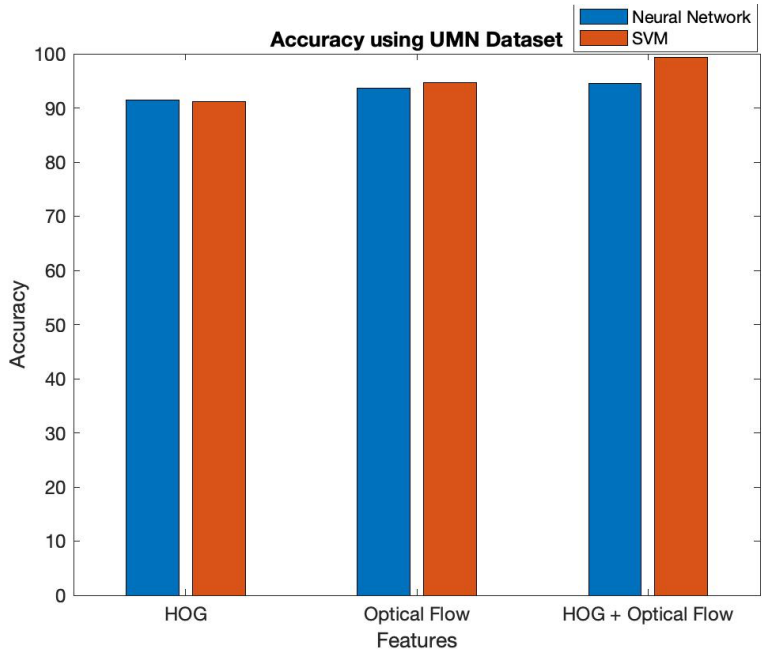



**FIGURE 3.11 Performance of HOG and optical flow features using NN and SVM classifiers**

Figure 3.11 shows the performance of HOG, optical flow and a combination both features using neural network and SVM Classifiers. In the figure, horizontal axis represents features and vertical axis represents accuracy percentage. Figure shows that the optical flow features give higher accuracy compared to HOG features using both classifiers and accuracy is increased by using both features collectively.

## 3.2 Autoencoder and Optical Flow Based Approach

### 3.2.1 Autoencoder

The Autoencoder is introduced by Rumelhart et al. [65]. A standard autoencoder architecture includes three main components, as shown in Figure 3.12. First, an encoder contains a series of layers with the number of nodes in decreasing order and eventually reduces to a latent space representation. The second component is latent space representation, also known as a bottleneck. This is the lowest possible dimension of the input data where information is preserved. The third component is a decoder, which is the mirror image of the encoder in which the number of nodes increases in every layer and ultimately outputs a similar input.

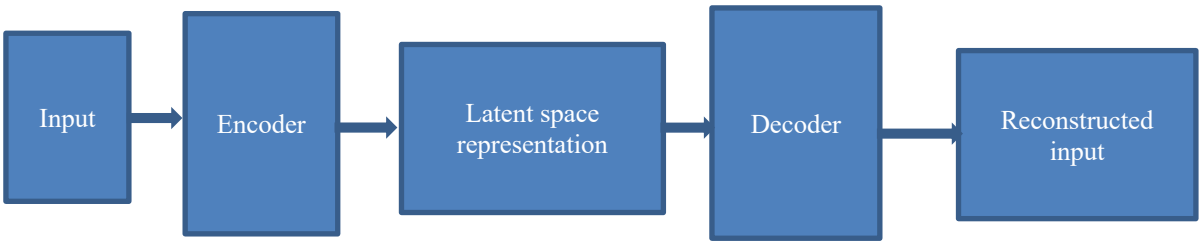



**FIGURE 3.12 Autoencoder**

An Autoencoder takes an input $\mathbf{x} \in R^d$ and first maps it to the latent representation (hidden layer) $\mathbf{h} \in R^{d'}$ using the following mapping function.

$$h = f_\theta = \sigma(w x + b) \text{ where parameters } \theta = \{ w, b\} \qquad (3.17)$$

The following reverse mapping function is used for reconstructing the input, as shown in equation 3.18.

$$f : y = f_{\theta'}(h) = \sigma(w'h + b') \text{ where parameters } \theta' = \{ w', b'\} \quad (3.18)$$

The parameters w learned from the input layer to the hidden layer create the encoder, and the parameters w' learned from the hidden layer to the output layer define the decoder. The decoder parameters are normally related to the parameters in the encoder by $w' = w^T$ [66]. Training an autoencoder does not require label information of the input data. It uses the backpropagation algorithm to minimize the cost function. The cost function, i.e. reconstruction error, is mean squared error e between each input $x_i$ and the corresponding output $y_i$ , by adjusting the parameters of the encoder w and the decoder w', as shown in equation-3.19.

$$e(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \qquad (3.19)$$

The highly tuned autoencoder model should be able to reconstruct the same input which is given in the first layer. Autoencoders are extensively used with image data for the applications like Feature Extraction, Dimensionality Reduction, Image Compression, Image Denoising, and Image Generation. Variants of autoencoders are sparse autoencoder, variational autoencoder, convolutional autoencoder, stacked autoencoders, etc. In this approach, a sparse autoencoder is used.

### Sparse Autoencoder

A sparse autoencoder is an autoencoder whose training criterion involves a sparsity penalty. The loss function is constructed by penalizing activations of hidden layers so that only a few nodes are encouraged to activate when a single sample is fed into the network [67]. By activating a few nodes at a time, the autoencoder actually learns latent representations instead of redundant information in input data. Figure 3.13 represents the sparse autoencoder architecture.

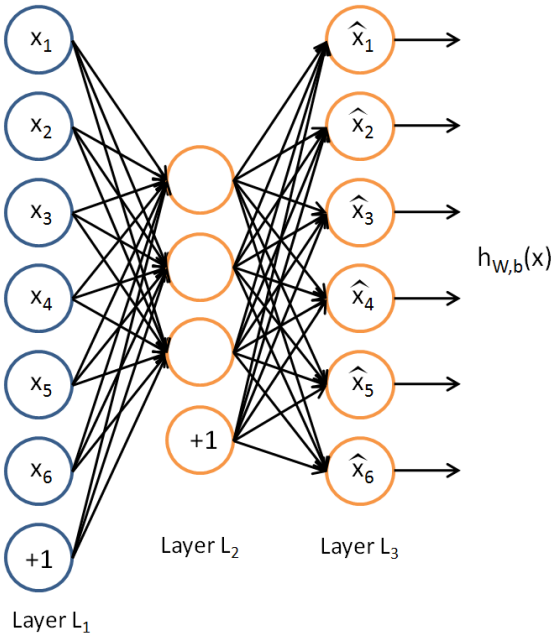



**FIGURE 3.13 Sparse autoencoder [68]**

### Parameters

**Sparsity regularization:** The autoencoder tries to learn a function $h_{w,b}(x) \approx x$. It tries to learn an approximation to the identity function, so the output is similar to the input. The identity function looks to be the insignificant function to be trying to learn, but by placing constraints on the network, such as by limiting the number of hidden units, interesting structure about the data can be found. If the number of hidden units is large, then sparsity constraint can be applied.

Consider $a_j^l$ denotes the activation of hidden unit j in the autoencoder at layer l. The average activation of hidden unit j (averaged over the training set) is represented as equation-3.20.

$$\rho_j = \frac{1}{n}\sum_{i=1}^{n} a_j^l x^i \qquad (3.20)$$

The constraint enforced $\rho_j = \rho$ where ρ is a sparsity parameter, typically a small value close to zero. To achieve this, an additional penalty term can be added, which is known as the sparsity regularization term. Many such terms give reasonable results. One such sparsity regularization term is the Kullback-Leibler divergence, as shown in equation-3.21.

$$\Omega_{sparsity} = \sum_{j=1}^{s_2} KL(\rho \,||\, \rho_j) = \sum_{j=1}^{s_2} \rho \log\frac{\rho}{\rho_j} + (1 - \rho) \log\frac{(1-\rho)}{(1-\rho_j)} \qquad (3.21)$$

Where, $s_2$ is the number of neurons in the hidden layer, and the index j is summing over the hidden units in the network. If ρ and $\rho_j$ are same, KL ($\rho \,||\, \rho_j$) is 0; otherwise it increases monotonically as $\rho_j$ diverges from ρ.

**$L_2$ weight regularization:** When training a sparse autoencoder, the weights may be increased to make the sparsity regularization term small. A regularization term on the weights to the cost function is added to prevent this. This term is called the $L_2$ regularization term and is defined by equation-3.22.

$$\Omega_{weights} = \frac{1}{2} \sum_{l}^{L} \sum_{j}^{n} \sum_{i}^{k} \left(w_{ji}^{(l)}\right)^2 \qquad (3.22)$$

where L is the number of hidden layers, n is the number of observations and k is the number of variables in the training data.

**Maximum Epochs**: It represents the maximum number of training iterations.

### 3.2.2 The Proposed Approach

As shown in the Figure 3.14, two types of features are extracted from the input image, one is features extracted by autoencoder, and the other is optical flow features. Here, only the encoder part of the sparse autoencoder is used which extracts features from the input image. As discussed in section- 3.1.4, optical flow features are extracted from the input

image and PCA is applied to reduce dimensionality. Finally, the feature vector that contains features extracted from the encoder and optical flow features is given as input to a neural network that classifies the image as normal or abnormal.

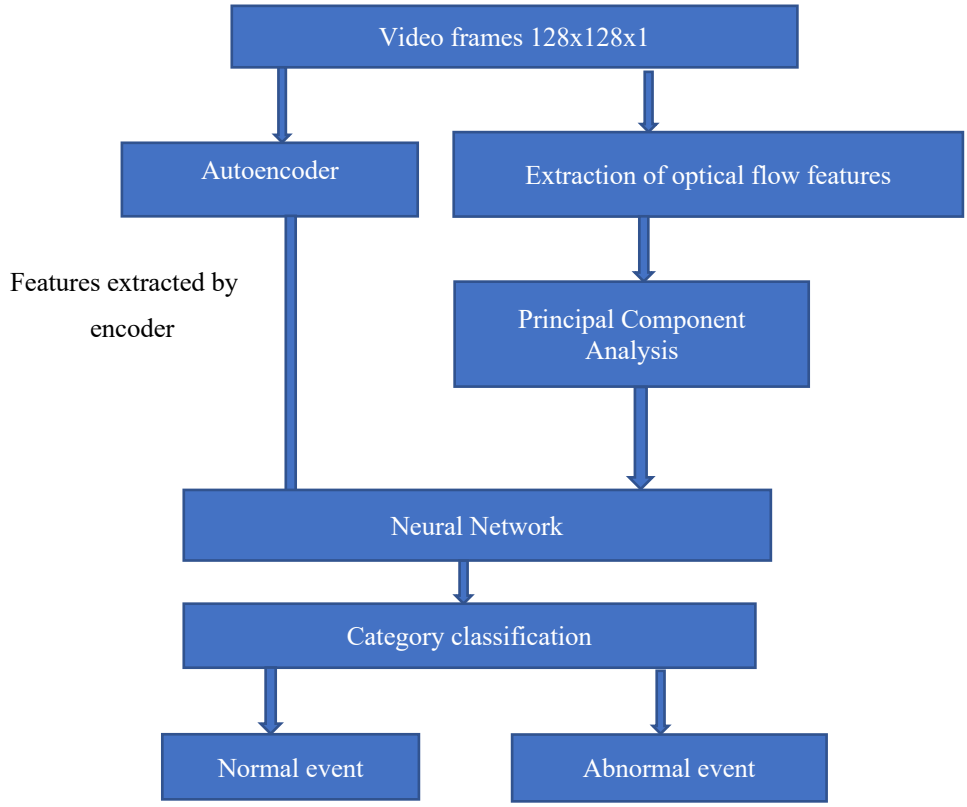



**FIGURE 3.14 Autoencoder and optical flow based approach**

### 3.2.3 Implementation Details and Experimental Results

The optical flow represents the magnitude and direction for each pixel. Here, the image size is 128x128, so the optical flow features for a single image are 16384+16384=32718. After Applying PCA, the number of optical flow features for a single image of scene - 1 is 2904. In this experiment, a sparse autoencoder with 100 nodes is used, so the number of features extracted by the encoder is 100. So final feature vector of the image contains 3004 features. The set parameter values are as follows.

Sparsity regularizer : 1

$L_2$ weight regularizer : 0.001

Maximum Epochs : 100

The experiment is performed on three different scenes of the UMN dataset for 80% training and 20% test data. Figure 3.15 shows the extracted features by the encoder from a sample image of scene - 1. The features with labels are given as input to a simple feed-forward neural network which contains one hidden layer with ten nodes. Figure 3.16 represents the generated pattern recognition network for scene - 1.

**FIGURE 3.15 Features extracted by autoencoder**

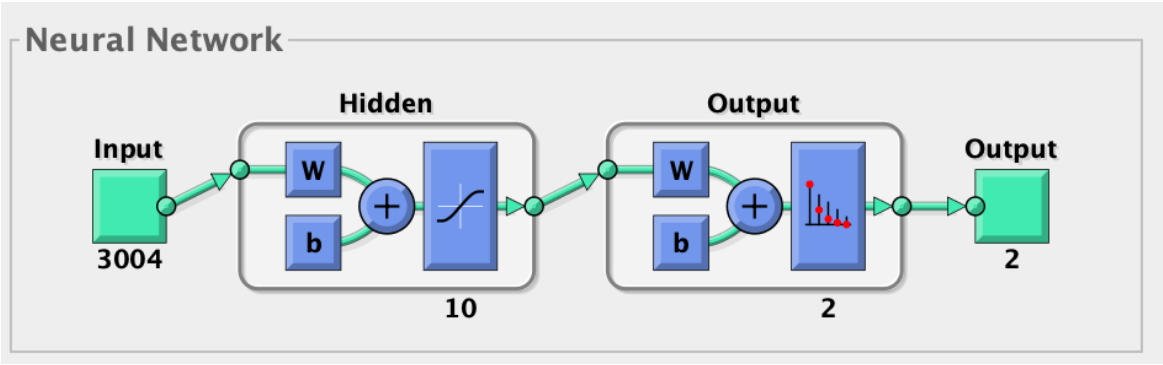



**FIGURE 3.16 Generated pattern recognition network for the UMN dataset scene - 1**

The achieved accuracy is 100%, and the AUC value is 1. Table 3.3 shows the confusion matrix for the proposed approach using the UMN dataset. To prove that the combination of both features improves performance, three experiments are done, i.e. using only optical flow, using only features extracted by an encoder and using a combination of both. Using only optical flow, the achieved average accuracy of three scenes is 96.7%. The achieved average accuracy using only features extracted by the encoder is 84.7%. Whenever both features are combined, 100% accuracy is obtained. The experiments are also carried out using 70% training and 30 % test data and for that also each image is classified correctly.

**Table 3.3 Confusion matrix for the UMN dataset**

| True class | | Predicted class | | | | | |
|---|---|---|---|---|---|---|---|
| | | Scene - 1 | | Scene - 2 | | Scene - 3 | |
| | | Abnormal | Normal | Abnormal | Normal | Abnormal | Normal |
| | Abnormal | 34 | 0 | 169 | 0 | 18 | 0 |
| | Normal | 0 | 256 | 0 | 658 | 0 | 410 |

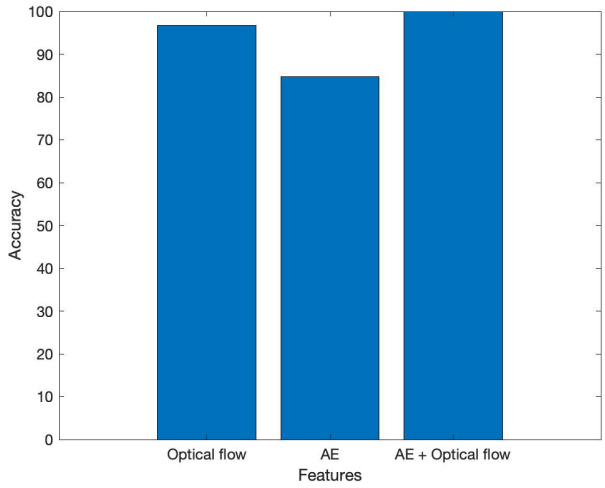



**FIGURE 3.17 Performance of optical flow and AE features using neural network classifier**

Figure 3.17 shows the performance of optical flow, autoencoder features and a combination of both features using neural network Classifier. In the figure, horizontal axis represents features and vertical axis represents the accuracy percentage. Optical flow features give higher accuracy compared to features extracted by autoencoder but when both features are used combinedly then 100% accuracy is achieved.

## 3.3 Comparison of the Proposed Approaches

Figure 3.18 shows the comparison of HOG and optical flow based approach with autoencoder and optical flow based approach using neural network classifier. In the figure horizontal axis represents approaches and vertical axis represents the accuracy percentage. Figure represents that autoencoder and optical flow based approach performs better than HOG and optical flow based approach.

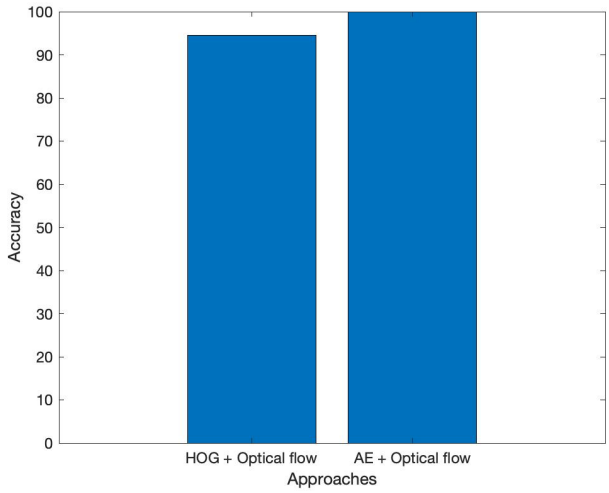



**FIGURE 3.18 Performance of the proposed approaches using neural network classifier**

## 3.4 Concluding Remarks

This chapter represents two approaches based on supervised machine learning algorithms for abnormal event detection in a crowd scene. The experiments are done using UMN dataset. In the first approach, HOG and Optical flow features are used with neural network and SVM classifiers. Using only HOG features the achieved accuracy is 91.49% with neural network classifier and 91.15% with SVM classifier. Using only optical flow features the achieved accuracy is 93.72% with neural network classifier and 94.73% with SVM classifier. A combination of HOG and optical flow features gives higher accuracy compared to the individual one using both classifiers. It gives 94.48% accuracy using neural network and 99.35% accuracy using SVM classifier.

The second approach is based on autoencoder and optical flow. The encoder component of autoencoder extracts features from the input image and optical flow represents motion information. In this approach neural network is used as a classifier. Using features extracted by encoder, the achieved accuracy is 84.7%. Using optical flow features, the achieved accuracy is 96.7%. A combination of both features gives higher accuracy compared to the individual one. It gives 100% accuracy and AUC value 1.

The comparison between these two methods is also shown in terms of accuracy. The approach based on autoencoder and optical flow performs better than the approach based on HOG and optical flow.

# CHAPTER - 4

# Approaches Based on Supervised Deep Learning Algorithms

## 4.1 Introduction

Deep learning is a subset of machine learning inspired by the working of the human brain. Human brain interprets the information, labels it and assigns it into various categories. When new information is provided to the brain, it compares with the existing information, does analysis and arrives at a conclusion. Deep learning is based on numerous layers of algorithms, i.e. artificial neural networks, each providing a different interpretation of the data that's been fed to them [69].

The word "deep" typically points out the number of hidden layers in the neural network. Usually, the traditional neural networks have only two-three hidden layers since deep neural networks can have many layers like more than 100 layers. The traditional neural network requires manual feature extraction, while the deep neural network model is trained with raw data, as various features are extracted automatically at different layers. So it requires a large amount of labelled data to learn itself.

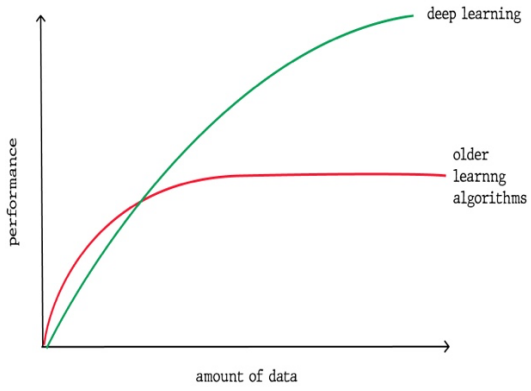



**FIGURE 4.1 Performance evaluation for deep learning and machine learning algorithms [69]**

Deep learning gives much more accuracy than other feature extraction-based classifiers for a large amount of data. So, as dataset size increases, the performance of a deep neural network also increases. Figure 4.1 shows the performance of deep learning vs older machine learning algorithms.

This chapter presents three proposed supervised deep learning-based approaches in detail. (1) Stacked Autoencoder based approach (2) Convolutional Neural Network based approach and (3) CNN-LSTM based approach.

## 4.2 Stacked Autoencoder Based Approach

### 4.2.1 Stacked Autoencoder

The stacked autoencoder is a neural network in which hidden layers are trained by an unsupervised method and then fine-tuned by a supervised manner. A stacked autoencoder consists of several layers of sparse autoencoders where the output of each hidden layer is connected to the input of the successive hidden layer. Stacked autoencoder mainly consists of following steps [67]. First, the autoencoder is trained using input data and acquiring the learned data. Second, the learned data from the previous layer is used as an input for the next layer and this continues until the training is completed. Finally, once all the hidden layers are trained, the backpropagation algorithm is used to minimize the cost function, and weights are updated with the training set to achieve fine-tuning.

The contemporary advancement in stacked autoencoder is that it provides a version of raw data with much detailed and useful feature information, which is used to train a classifier with a determined context and find better accuracy than training with raw data.

### 4.2.2 The Proposed Approach

The main objective of using a stacked autoencoder is to train a deep neural network by training one layer at a time. Figure 4.2 represents the proposed approach. Here, two sparse autoencoders are used, one with 100 nodes and the other with 50 nodes. Like the approach

discussed in section 3.2, in this approach also, only encoder components of autoencoders are used.

First, training images are given to autoencoder1, which contains 100 nodes. The features extracted by encoder1 are provided as input to autoencoder2, including 50 nodes. The number of nodes in the second autoencoder is decreased so that its encoder learns a smaller representation of the input data. The features extracted by encoder2 are given as input to the softmax layer with labels. Then both encoders are stacked with a softmax layer to form a stacked network. This network is fine-tuned by retraining it on the training data in a supervised manner as shown in Figure 4.2.

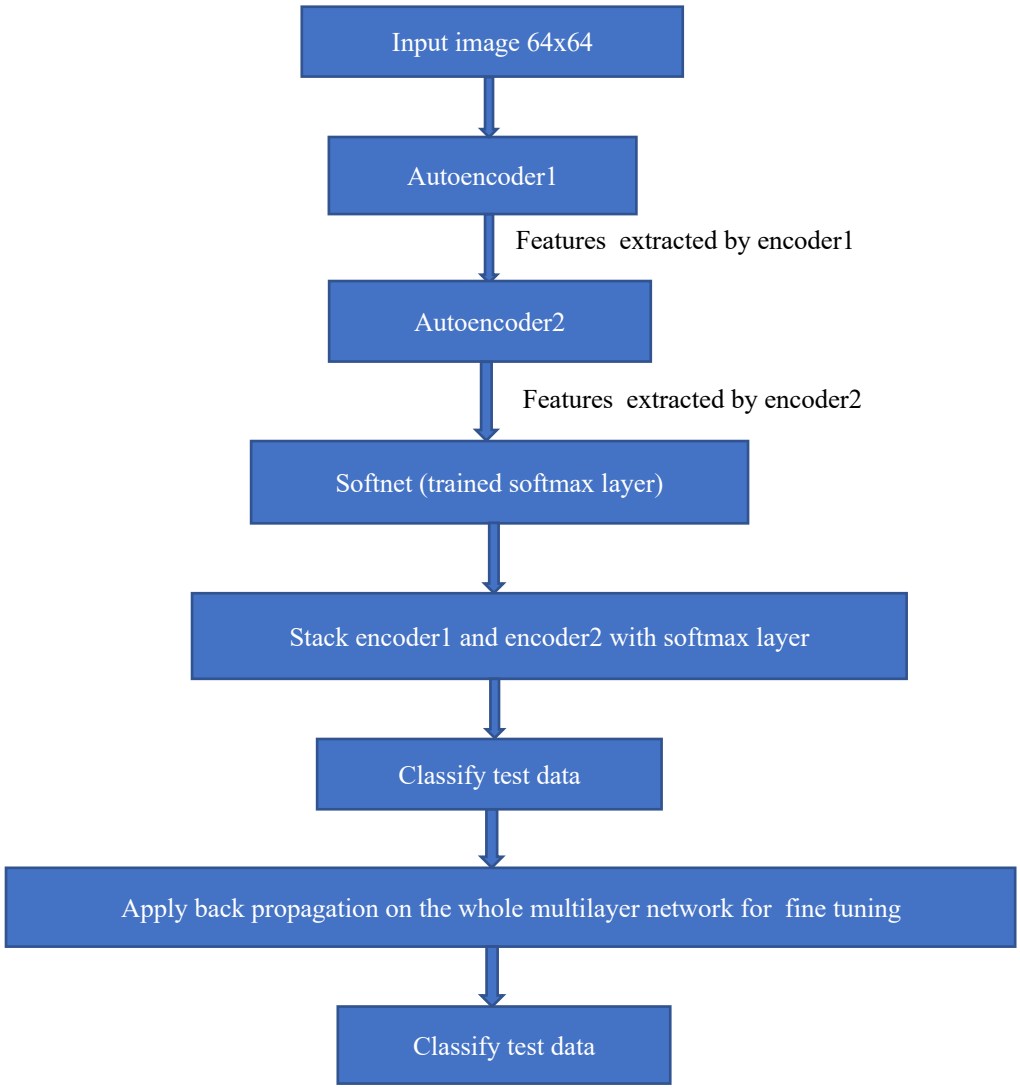



**FIGURE 4.2 Stacked autoencoder based approach**

### 4.2.3 Experimental Results

Figure 4.3 shows the generated stacked network diagram. Here the input image size is 64x64=4096. The features extracted by encoder1 and encoder2 are 100 and 50, respectively. Table 4.1 shows experimental results on various datasets with 80% training and 20% test data except for the UCFCrime2Local dataset. For the UCFCrime2Local dataset, experiments are done as per the given train-test split and for abnormality detection, all six events except for Normal are considered abnormal events. Table 4.2 represents the confusion matrices for the same. This method gives adequate performance on all datasets except for UCFCrime2Local dataset as it is weakly labelled complex dataset.

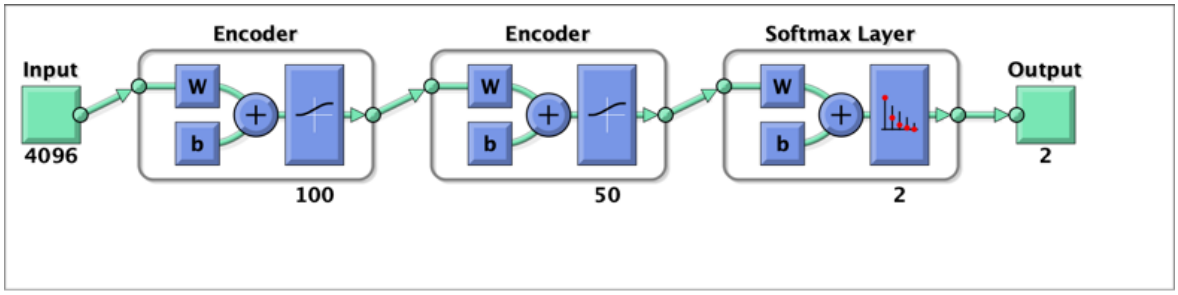



**FIGURE 4.3 Generated stacked network**

**TABLE 4.1 Experimental results for abnormal event detection using stacked autoencoder**

| Dataset | Number of images | Accuracy (%) | Area Under Curve |
|---|---|---|---|
| UMN | 7003 | 96.9 | 0.9079 |
| UCSDPed1 | 14000 | 98.6 | 0.9811 |
| Violent Flows | 21800 | 93.1 | 0.9269 |
| Subway Entrance | 143996 | 99.6 | 0.9586 |
| UCFCrime2Local | 251151 | 81.4 | 0.5229 |

**TABLE 4.2 Confusion matrices for various datasets using stacked autoencoder**

| Predicted class | | Actual class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | UCFCrime2-Local | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 195 | 0 | 785 | 16 | 2364 | 197 | 691 | 49 | 884 | 1083 |
| | N | 44 | 1161 | 24 | 1975 | 104 | 1695 | 61 | 27998 | 13132 | 61379 |

## 4.3 Convolutional Neural Network Based Approach

### 4.3.1 Convolutional Neural Network

Convolutional neural network (CNN) is one of the variants of traditional artificial neural networks mostly used for feature extraction and classification of image and video data. It is a deep learning algorithm that can take an image as an input, assign importance in terms of learnable weights and biases to various aspects/objects in the image, and differentiate one from the other. The convolutional neural network is also known as ConvNet. The basic idea of convolutional neural network was first proposed in 1989s by Yann LeCun [70], then after various researchers further improved it.

Figure 4.4 shows the basic architecture of CNN for object recognition. The architecture is divided into two parts, feature learning and classification. Layers used in feature learning are repeated combinations of the convolutional, activation, and pooling layers. In the classification layer, the output of the previous layers is mostly flattened and fed into the fully connected layer. A fully connected layer has the number of neurons equal to the number of classes or categories into which data is classified at the end.

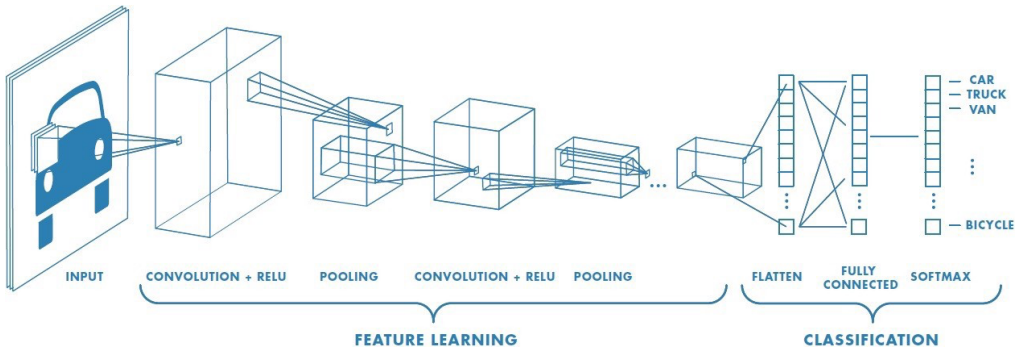



**FIGURE 4.4 CNN architecture for object classification [81]**

CNN contains a sequence of various layers with tuned parameters and hyperparameters. The main layers of CNN architecture are as follows.

- Input Layer: It is designed to input raw data to the model. Colour or grayscale images are given as input.
- Convolution Layer: It computes a dot product between the input image patch and the filter and gives the output volume based on input. It extracts various features

from the image like edges, colour, gradient orientation etc. This operation provides two types of results, one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. To get increased or the same dimensionality, padding is applied.

- Activation function layer: The output of the convolution layer is fed to this layer which applies a specific activation function to input. There are different activation functions like ReLU (Rectified Linear Unit), tanh, sigmoid etc. ReLU is the most widely used activation function in CNN.
- Pooling layer: This layer reduces the spatial size of the convolved features so that less computational power is required to process the data. It is used to extract dominant features, which are rotational and positional invariant. There are two types of pooling: Max pooling and Average pooling. Max pooling returns the maximum value from the image portion covered by the filter. Average pooling produces the average of all the values from the image portion covered by the filter.
- Fully connected layer: It gives computed 1-D array class scores from the input received, which are used to classify data using the softmax classification technique.
- Batch normalization layer: Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini batch.
- Softmax layer: The softmax function calculates relative probability. The input values can be positive, negative, zero, or greater than one. The softmax transforms them into values between 0 and 1 to be interpreted as probabilities.
- Dropout layer: It is a method to avoid a few randomly chosen neurons in the training of CNN. These neurons do not contribute to developing the model. This approach is used to reduce overfitting and make the model generalized.

In this research work, two types of experiments are performed. One in which transfer learning is used and the other in which a new CNN model is developed from scratch. Both are discussed in detail as follows.

### 4.3.2 Transfer Learning Approach

The reuse of a previously learned model for solving a new problem is known as transfer learning. The significant advantages of transfer learning are less training time and the absence of a large amount of data. As the model has been pre-trained, a small amount of data is required in training, so it takes less time. Pre-trained network can be used in two ways. One is last few layers are retrained as per requirements, and the other is it can be used for only feature extraction, and those features are given as input to another classifier.

Although it has achieved good results in many cases, it is not appropriate for all applications because it is hard to know how much the previous training process helps. In addition, only retraining the last few layers cannot guarantee the best results because categories for source training data are generally different and the semantic representations in the higher layer are relative to the training categories. Therefore, using the highly specific semantic representation in the new recognition task is not worthy.

Transfer learning is typically used in Computer Vision and Natural Language Processing tasks. Pretrained networks are trained on large datasets like the ImageNet, which consists of more than a million images classified into many classes. The various pre-trained CNN architectures like LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet etc. are used by different researchers for transfer learning.

#### 4.3.2.1 GoogLeNet Architecture

In this work, GoogLeNet [71] architecture is used. GoogLeNet was proposed by research at Google in 2014. It was a winner at the ILSVRC 2014 image classification challenge. It has reported less error rate compared to AlexNet and ZF-Net. The GoogLeNet network is trained on the ImageNet dataset and classifies 1000 objects such as keyboard, mouse, pencil, and many animals. It contains an inception module in which 1x1, 3x3, 5x5 convolution and 3x3 max pooling are performed in a parallel way at the input, and its outputs are stacked together to generate the final output. Because of different sizes of convolution filters, the model can handle multiple scale objects preciously. Table 4.3 represents GoogLeNet architecture which is 22 layers deep. It takes 224x224 RGB colour images as input.

**TABLE 4.3 GoogLeNet architecture**

| Operation | Filter size | Number of filters | Stride | Output size |
|---|---|---|---|---|
| convolution | 7x7 | 64 | 2 | 112x112x64 |
| max pool | 3x3 | | 2 | 56x56x64 |
| convolution | 3x3 | 192 | 1 | 56x56x192 |
| max pool | 3x3 | | 2 | 28x28x192 |
| inception (3a) | | | | 28x28x256 |
| inception (3b) | | | | 28x28x480 |
| max pool | 3x3 | | 2 | 14x14x480 |
| inception (4a) | | | | 14x14x512 |
| inception (4b) | | | | 14x14x512 |
| inception (4c) | | | | 14x14x512 |
| inception (4d) | | | | 14x14x528 |
| inception (4e) | | | | 14x14x832 |
| max pool | 3x3 | | 2 | 7x7x832 |
| inception (5a) | | | | 7x7x832 |
| inception (5b) | | | | 7x7x1024 |
| Avg pool | 7x7 | | 1 | 1x1x1024 |
| dropout (40%) | | | | 1x1x1024 |
| fully connected | | | | 1x1x1000 |
| softmax | | | | 1x1x1000 |
| classification output | | | | 1000 |

#### 4.3.2.2 Implementation Details and Experimental Results

In a convolutional neural network, lower layers represent low-level features such as edges, textures, colours etc., whereas higher layers represent high-level semantic features such as objects or parts of objects. Most of the time, low level features represent similar information in many cases. Compared to low-level features, the high-level features are not the same for different categories of objects. Due to this fact, any good method used for transfer learning should have the ability to learn features that are common and tune the high-level features according to the categories of the target. The last few layers must be altered as per requirements, like number of classes.

The transfer learning approach using a pre-trained network is a significantly faster and easy way of training. To carry out the transfer learning approach, all the pre-trained

GoogLeNet network layers were kept as they are, except the last three layers: fully connected layer, softmax layer and classification output layer. These three layers are altered because in our experiments, only two categories are required Normal and Abnormal instead of 1000. The set parameters are as follows.

Optimizer: SGDM

Initial Learn Rate: 0.01

Maximum epochs: 50

Batch size: 64

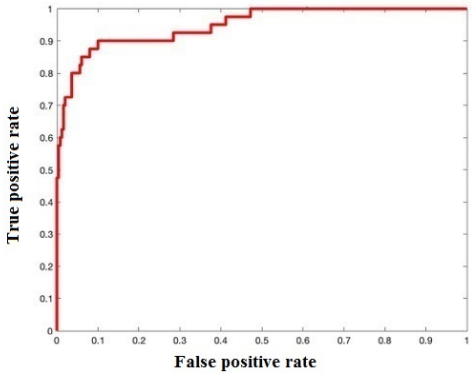



**FIGURE 4.5 ROC curve for the UMN dataset using GoogLeNet architecture**

Here, the experiment is done using the UMN dataset, for which 86.21% accuracy and 0.9487 AUC value is achieved for 80% training and 20% test data. Figure 4.5 shows the ROC curve for the same.

### 4.3.3 The Proposed CNN Architectures

The performance of transfer learning was not sufficient for the small and simple dataset, so there is a need to develop a new CNN model from scratch to detect and classify abnormal events in surveillance scenes. To develop a new model, the experiments were started with small architecture, and by deep analysis of the performance, the final model is proposed. Two CNN models are developed as a result of various experiments using different datasets. The first one is a small architecture that performs well on small datasets, but it does not give acceptable results for complex and large datasets. So, another architecture is developed, which is deeper than the proposed first architecture and smaller than GoogLeNet. It gives excellent results on all the mentioned datasets.

#### 4.3.3.1 CNN Architecture1

To develop a new CNN model, initial experiments were started with small architecture containing fewer layers and filters. In the proposed CNN Architecture1, Fourteen layers are used as described in Table 4.4. The proposed Arhitecture1 is shown in Figure 4.6.

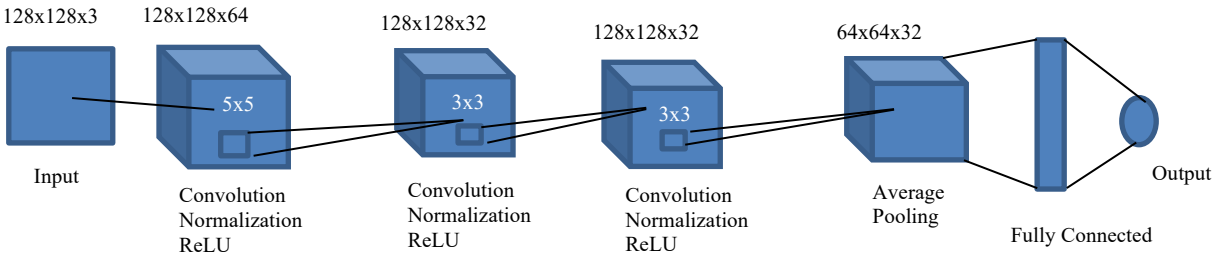



**FIGURE 4.6 The proposed CNN Architecture1**

**TABLE 4.4 CNN Architecture1**

| Sr No | Layer name | Operation | Description |
|---|---|---|---|
| 1. | 'input' | Image input | 128×128×3 images with 'zerocentre' normalization |
| 2. | ‘conv_1' | Convolution | 64 5×5×3 convolutions with stride [1 1] and padding 'same' |
| 3. | 'bn_1' | Batch normalization | Batch normalization with 64 channels |
| 4. | 'relu_1' | ReLU | ReLU |
| 5. | 'conv_2' | Convolution | 32 3×3×64 convolutions with stride [1 1] and padding 'same' |
| 6. | 'bn_2' | Batch normalization | Batch normalization with 32 channels |
| 7. | 'relu_2' | ReLU | ReLU |
| 8. | 'conv_3' | Convolution | 32 3×3×32 convolutions with stride [1 1] and padding 'same' |
| 9. | 'bn_3' | Batch normalization | Batch normalization with 32 channels |
| 10. | 'relu_3' | ReLU | ReLU |
| 11. | 'avgpool' | Average pooling | 2×2 average pooling with stride [2 2] and padding [0 0 0 0] |
| 12. | 'fc' | Fully connected | 2 fully connected layer |
| 13. | 'softmax' | Softmax | softmax |
| 14. | 'classOutput' | Classification output | crossentropyex with classes 'Abnormal' and 'Normal' |

In CNN, features are extracted by convolution operation on the input image. The size of the input image is taken 128x128x3. To extract large objects, filter size should be large, and to detect small objects, filter size should be small. So initially, filter size is taken 5x5, then decreased to 3x3 in deeper convolution operation.

Here three convolution layers are used. For each convolution layer, zero padding is set such that the output size of the image remains the same as input size. For each convolution layer, biases and weights are initialized with the Glorot initializer [72]. Glorot et al. [72] studied how activations and gradients varied across layers and during training and proposed a new initialization scheme that brings substantially faster convergence. So, as per [72], biases are initialized to be 0 and the weights $W_{ij}$ with the following commonly used heuristic: $W_{ij} \sim U\left[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ where U[−a, a] is the uniform distribution in the interval (−a, a) and n is the size of the previous layer.

The particular neuron would fire, or not that can be decided by the use of activation function. Here, Rectified Linear Unit (ReLU) is used as an activation function. ReLU is a piecewise linear function that will output the input directly if it is positive. Otherwise, it will output zero. In CNN, ReLU is a commonly used activation function because it does not activate all neurons simultaneously. It is defined as y = max (0, x). It converts all negative inputs to zero. This makes it mathematical operative as few neurons are activated every time. ReLU converges faster than other activation functions. It doesn't use complex math; therefore, the model takes less time to be trained. Convolutional layers encapsulate the presence of features in an input image. The output feature maps are easily affected by the position of the features in the input. They can be more robust to changes in the location of the feature in the image by down-sampling. Pooling layers come up with a technique to down-sampling feature maps by sum up the appearance of features in patches of the feature map. In the proposed Architecture1, one average-pooling layer is used, which summarizes the average presence of a feature. Average pooling layer is used with pool size 2 and stride (Step size for traversing the input vertically and horizontally) 2.

To train a deep neural network is a very challenging task because the input from previous layers can alter after each mini-batch, when weights are updated. Batch normalization is a

technique to standardize the inputs to a layer for each mini-batch. Batch normalization normalizes the input values to have a mean of zero and a variance of one by the following formula. $\hat{u} = \frac{u-mean}{\sqrt{Variance}} = \frac{u-E[u]}{\sqrt{Var[u]}}$. Because of it, the learning process becomes stabilized, and the required number of training epochs can be reduced. Lower learning rate slows the model training. By use of batch normalization layer, the learning rate can be initialized with high value. Batch normalization reduces generalization error and fulfils the goal of using the dropout layer so that dropout can be removed or its occurrences can be reduced. Like this, it accelerates training. So here batch normalization layer is used after each convolution layer.

A non-adaptive optimizer can outperform adaptive optimizers, but only at the cost of a significant amount of training time to perform hyperparameter tuning [73]. So, here a non-adaptive optimizer, Stochastic Gradient Descent (SGD) is used with momentum 0.9, L2Regularization 1.0000e-04 and Gradient Threshold Method L2norm.

To generate a successful model, various parameters are tuned like Number of layers, Order of layers, Number of filters, Filter size, Initial learning rate, Learning rate drop factor, Learning rate drop period, Epochs and Batch Size. Learning rate is the most crucial parameter for model building. It specifies how fast the network learns. Generally, its value lies between 0 to 1.

**Experimental Results**

In the proposed Architecture1, three convolution layers are used with a different number of filters. For convolution layer one, two and three, 64, 32 and 32 filters are applied for each input image, respectively. Initial filter size is taken 5x5, then decreased to 3x3. The features learned by the network can be seen by comparing areas of activation with the original image.

For the Architecture1, experiments are done using the UMN and Airport-WrongDir datasets. The extracted features are shown in Figure 4.7. The Figure represents input image and the largest activation at each convolution layer. The learning way of the network can be recognized by identifying features in this manner. In the experiments, the parameters are tuned as follows.

Optimizer : SGDM

Initial Learning Rate : 0.01

Learning Rate Drop Factor : 0.2

Learning Rate Drop Period : 5

Number of Epochs : 20

Batch size : 100.

(a) UMN dataset (b) Airport Wrong-Dir dataset

**Convolution layer – 1**

(a) UMN dataset (b) Airport Wrong-Dir dataset

**Convolution layer – 2**

(a) UMN dataset (b) Airport Wrong-Dir dataset

**Convolution layer - 3**

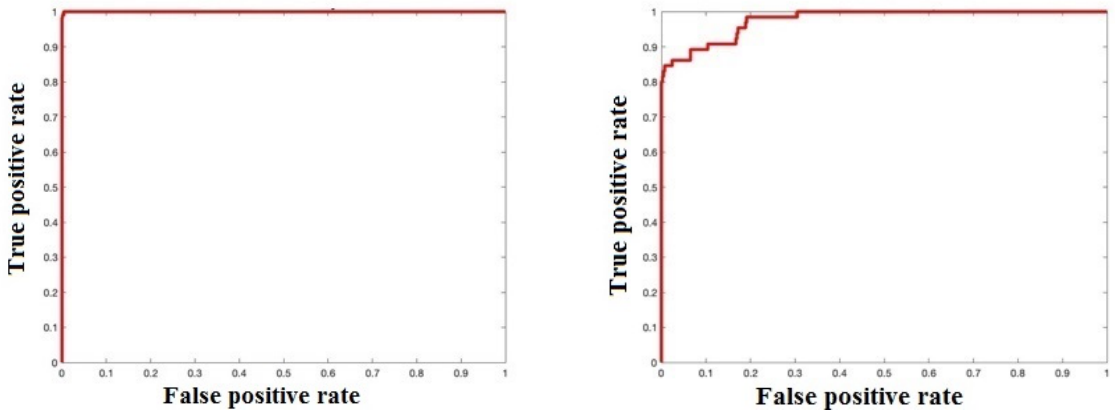
**FIGURE 4.7 The Largest activation at each convolution layer using CNN Architecture1**

**TABLE 4.5 Experimental results using CNN Architecture1**

| Dataset | Number of Images | Accuracy (%) | Area Under Curve |
|---|---|---|---|
| UMN | 7003 | 99.64 | 0.9999 |
| Airport-WrongDir | 2392 | 97.28 | 0.9774 |

For the UMN dataset, accuracy and loss for training become stable after two epochs, and for the Airport-WrongDir dataset, they become stable after four epochs. The experiments

are done for 80% training and 20% test data. Table 4.5 shows experimental results. Figure 4.8 shows the ROC curves using both datasets.




**(a) ROC curve for UMN dataset** **(b) ROC curve for Airport-WrongDir dataset**

**FIGURE 4.8 ROC curves using CNN Architecture1**

As these two datasets are simple and have few images, the proposed Architecture1 gives good results. For other complex datasets, the achieved results were not satisfactory. So the other model is developed i.e. CNN Architecture2, as follows.

#### 4.3.3.2 CNN Architecture2

Figure 4.9 represents the proposed CNN Architecture2. In this model, twenty layers are used as described in Table 4.6. The size of the input image is taken 128x128. For colour images, the input layer size is 128x128x3, and for grayscale images, the size is 128x128x1. In the proposed model, four convolution layers are used. For each convolution layer, zero padding is set such that the output size of the image remains the same as the input size. Biases and weights are initialized with the Glorot initializer for each convolution layer.

For initial convolution operation, fewer large size filters are used to extract coarse details. Then sequentially, the filter size is decreased, and the number of filters is increased to extract fine details. Initially, 16 filters of size 11x11 are taken for the first convolution layer. ReLU is used as an activation function. After applying the activation function, the max-pooling layer is used, which summarizes the maximum presence of a feature. The max pooling layer is used with pool size 2 and stride 2, So now the image size is 64x64.

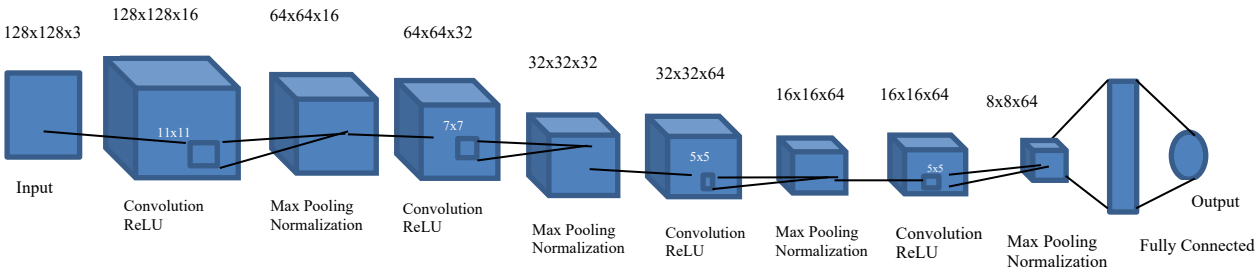



**FIGURE 4.9 The proposed CNN Architecture2**

During the study of CNN, it was found that few researchers [53] use batch normalization before the non-linearity, i.e. ReLU layer, and few researchers reported that adding batch normalization after the non-linearity improves accuracy [54, 55]. Here experiments are done using both architectures, and almost identical results are obtained on all datasets. So, the batch normalization layer is used after the max pooling layer in the proposed architecture. The same process is repeated three times with 32 filters of size 7x7, 64 filters of size 5x5 and again 64 filters of size 5x5, respectively.

Different researchers have various opinions about the use of the dropout layer. In [73], authors have reported that accuracy is decreased significantly by using the dropout layer. It should be used carefully to design CNN, as It cannot give better results in each application. At the same time, [74] shows dropout helps accuracy. As dropout is the one way to remove overfitting of the model, In this work, various experiments with the proposed Architecture2 are done like (1) No dropout, (2) single dropout before fully connected layer and (3) one dropout layer after each batch normalization layer. Significant variation is not found in all three cases for the experiments using the reported datasets. As batch normalization fulfils the goal of using dropout, it is not included in the proposed architecture.

Like Architecture1, here also a non-adaptive optimizer, Stochastic Gradient Descent (SGD), is used with momentum 0.9, L2Regularization 1.0000e-04 and gradient threshold method L2norm. The initial learning rate is taken 0.01.

**TABLE 4.6 CNN Architecture2**

| Sr No | Layer name | Operation | Description |
|---|---|---|---|
| 1. | 'input' | Image input | 128×128×3 images with 'zerocentre' normalization |
| 2. | 'conv_1' | Convolution | 16 11×11×3 convolutions with stride [1 1] and padding 'same' |
| 3. | 'relu_1' | ReLU | ReLU |
| 4. | 'maxpool_1' | Max pooling | 2×2 max pooling with stride [2 2] and padding [0 0 0 0] |
| 5. | 'bn_1' | Batch normalization | Batch normalization with 16 channels |
| 6. | 'conv_2' | Convolution | 32 7×7×16 convolutions with stride [1 1] and padding 'same' |
| 7. | 'relu_2' | ReLU | ReLU |
| 8. | 'maxpool_2' | Max pooling | 2×2 max pooling with stride [2 2] and padding [0 0 0 0] |
| 9. | 'bn_2' | Batch normalization | Batch normalization with 32 channels |
| 10. | 'conv_3' | Convolution | 64 5×5×32 convolutions with stride [1 1] and padding 'same' |
| 11. | 'relu_3' | ReLU | ReLU |
| 12. | 'maxpool_3' | Max pooling | 2×2 max pooling with stride [2 2] and padding [0 0 0 0] |
| 13. | 'bn_3' | Batch normalization | Batch normalization with 64 channels |
| 14. | 'conv_4' | Convolution | 64 5×5×64 convolutions with stride [1 1] and padding 'same' |
| 15. | 'relu_4' | ReLU | ReLU |
| 16. | 'maxpool_4' | Max pooling | 2×2 max pooling with stride [2 2] and padding [0 0 0 0] |
| 17. | 'bn_4' | Batch normalization | Batch normalization with 64 channels |
| 18. | 'fc' | Fully connected | 2 fully connected layer |
| 19. | 'softmax' | Softmax | softmax |
| 20. | 'classOutput' | Classification output | crossentropyex with classes 'Abnormal' and 'Normal' |

In the proposed Architecture2, four convolution layers are used with different channels. For convolution layer - 1, 2, 3 and 4, there are 16, 32, 64 and 64 activation channels for each input image, respectively. The features learned by the network can be seen by comparing areas of activation with the original image. The simple features like colour, edges etc., can be learned by the earlier layer channels, while the deeper layer channels can

learn the complex features. The learning way of the network can be recognized by identifying features in this manner.

**Experimental Results**

The crucial experiments are done using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local datasets with different partitions of data. The Airport-WrongDir dataset is not widely used by different researchers. To compare the performance with Architecture1, experiment is done using this dataset with 20% test data. The experiments are also done using the combined dataset containing all images of four different datasets, i.e. UMN, UCSDPed1, Violent Flows and Subway Entrance.

(a) UMN (b) UCSDPed1 (c) Violent Flows (d) Subway Entrance

**Convolution layer - 1**

(a) UMN (b) UCSDPed1 (c) Violent Flows (d) Subway Entrance

**Convolution layer - 2**

(a) UMN (b) UCSDPed1 (c) Violent Flows (d) Subway Entrance

**Convolution layer - 3**

(a) UMN (b) UCSDPed1 (c) Violent Flows (d) Subway Entrance

**Convolution layer - 4**

**FIGURE 4.10 The largest activation at each convolution layer using CNN Architecture2**

For the four mentioned datasets UMN, UCSDPed1, Violent Flows, and Subway Entrance, experiments are done using non-stratified holdout validation with two combinations, (i) 80% training data and 20% test data (ii) 70% training data and 30% test data. The experiments are also carried out with k fold cross-validation, where the value of k is taken 10. The datasets have enough video frames for a deep learning based approach and are also easily handled with a CPU.

The images of all datasets are resized to 128 x 128. The same proposed model with the same tuning parameters gives excellent performance on UMN, UCSDPed1, Violent Flows, and Subway Entrance datasets. It also provides outstanding results on the combined dataset. As here, the features are extracted by the CNN model, the activation of convolution and pooling layers represent somewhat information about the learned features like edges, corners, intensity etc. Figure 4.10 represents the most significant activation of each of the four convolution layers for an image of various datasets. The Figure represents input image and the largest activation at each convolution layer. In the activation channel image, strong positive and negative activations are represented by white and black pixels, respectively. The pixel position in the activation of a channel correlates with the pixel position in the original image.

Figure 4.11 shows the training progress of the CNN model for the UMN dataset with 80% training data. This Figure is helpful to observe the progress of the model training. By this Figure, improvement in network training accuracy at each epoch and underfitting or overfitting (if exists) can be monitored. Here 80% of data is taken for training, and 20% of data is taken for validation to observe the correlation between training and validation data at each iteration. If training accuracy is higher than validation/test accuracy, then overfitting can be considered, and if training accuracy is lower than validation/test accuracy, then underfitting can be considered. The Figure represents two graphs, Accuracy vs Iterations and Loss vs Iterations. The Figure displays training and validation metrics at every iteration with solid and dotted lines, respectively. Each iteration is an estimation of the gradient and an update of the network parameters. The curves of accuracy and loss for training data correlate with the curves of accuracy and loss for validation data, respectively. This Figure is represented here just to show that the proposed model neither overfits nor underfits; otherwise the validation set is not used in other experiments.

Figure 4.12 shows the training progress of the proposed model using the combined dataset with 80% training data. For all experiments, the number of epochs is taken 5. For all datasets, test accuracy is similar to train accuracy, and after two epochs, accuracy and loss both become stable. So, it can be concluded that the proposed model works perfectly.

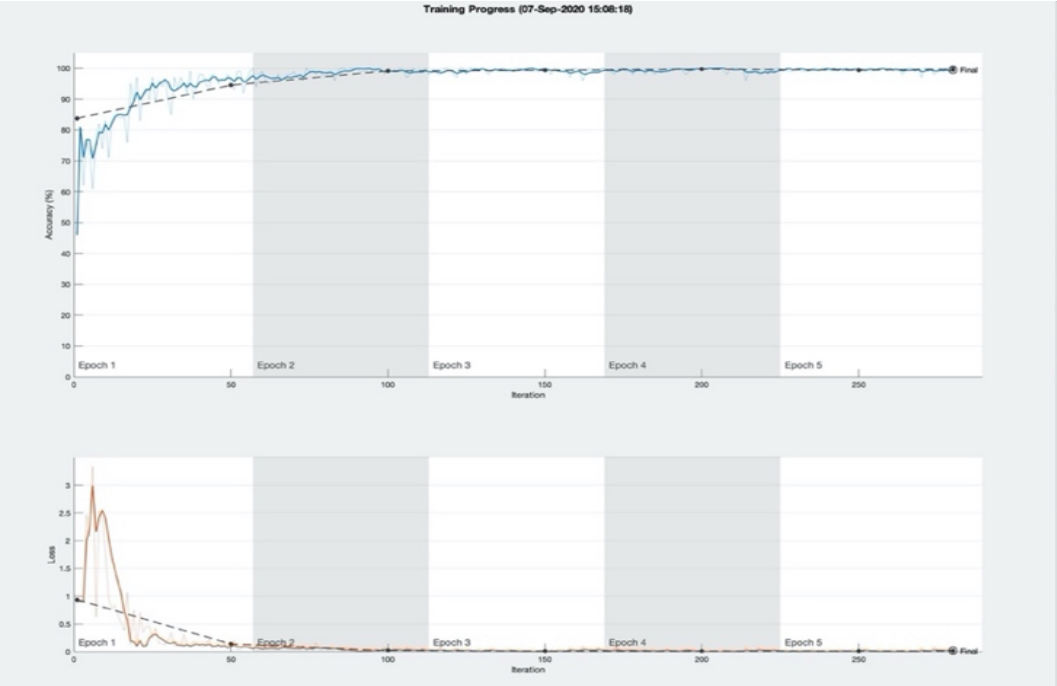



**FIGURE 4.11 Training progress of CNN Architecture2 using the UMN dataset**

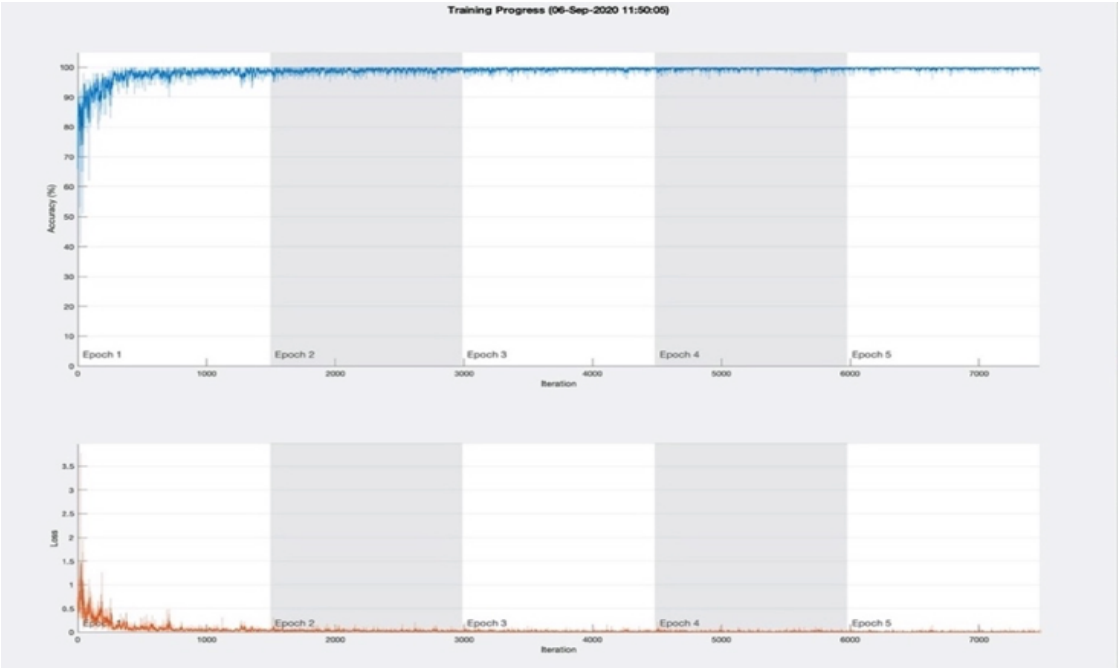



**FIGURE 4.12 Training progress of CNN Architecture2 using the combined dataset**

The proposed Architecture2 is used in two ways. In the first method, the CNN model is used for feature extraction as well as for classification. So after the fully connected layer, the classification layer is used to classify an image as normal or abnormal. In the second method, the CNN model is used to extract features, and SVM is used for classification. The extracted features by the CNN model are used to train the linear SVM classifier. CNN and SVM both classifiers almost give the identical results. Table 4.7 represents experimental results with 80% training data and 20% test data. Table 4.8 represents experimental results with 70% training data and 30% test data.

**TABLE 4.7 Experimental results using CNN Architecture2 with 20% test data**

| Dataset | Number of images | Accuracy (%) | | Area Under Curve | |
|---|---|---|---|---|---|
| | | Classification by CNN | Feature extraction by CNN & classification by SVM | Classification by CNN | Feature extraction by CNN & classification by SVM |
| UMN | 7003 | 99.50 | 99.43 | 0.9999 | 0.9991 |
| UCSDPed1 | 14000 | 99.89 | 99.89 | 1 | 0.9995 |
| Violent Flows | 21800 | 99.86 | 99.95 | 1 | 1 |
| Subway Entrance | 143996 | 99.99 | 99.99 | 1 | 1 |
| Combined dataset | 186799 | 99.79 | 99.88 | 1 | 0.9999 |
| Airport-Wrong Dir | 2393 | 97.70 | 98.33 | 0.9909 | 0.9909 |

**TABLE 4.8 Experimental results using CNN Architecture2 with 30% test data**

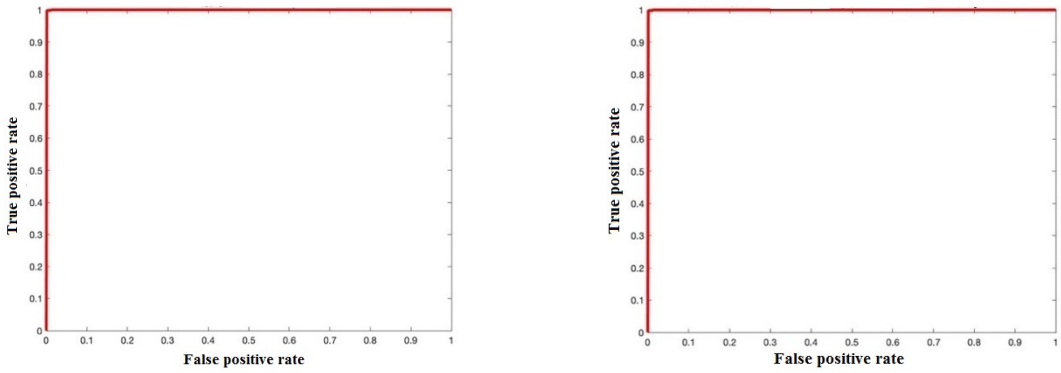
| Dataset | Number of images | Accuracy (%) | | Area Under Curve | |
|---|---|---|---|---|---|
| | | Classification by CNN | Feature extraction by CNN & classification by SVM | Classification by CNN | Feature extraction by CNN & classification by SVM |
| UMN | 7003 | 99.29 | 99.0 | 1 | 1 |
| UCSDPed1 | 14000 | 99.81 | 99.81 | 0.9997 | 0.9993 |
| Violent Flows | 21800 | 85.47 | 83.64 | 0.9413 | 0.9244 |
| Subway Entrance | 143996 | 75.23 | 74.61 | 0.6319 | 0.6436 |
| Combined dataset | 186799 | 94.48 | 93.47 | 0.9531 | 0.9501 |

Whenever the dataset is split into train and test set, the model's performance depends on the images which are used for training and testing. K-fold cross-validation gives accurate results, as all combinations of training and test set are considered for experiments. For 10-fold cross-validation, the whole dataset is split into ten groups. One by one, each group is taken as test data, and the remaining nine groups are taken as training data. So the whole execution process is done ten times. Then the average performance is considered as the final performance. This method gives actual performance rather than simply partition of data as 90% training and 10% testing. Table 4.9 represents experimental results with 10-fold cross-validation for each dataset.

**TABLE 4.9 Experimental results using CNN Architecture2 with 10-fold cross-validation**

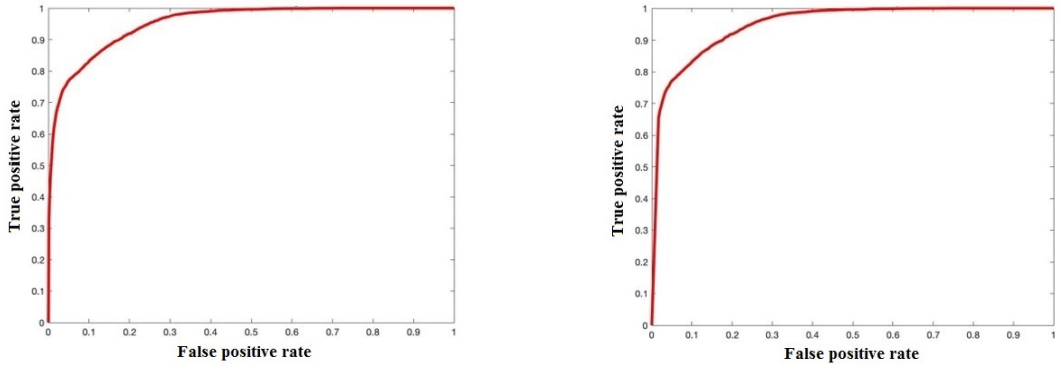
| Dataset | Number of images | Accuracy (%) | | Area Under Curve | |
|---|---|---|---|---|---|
| | | Classification by CNN | Feature extraction by CNN & classification by SVM | Classification by CNN | Feature extraction by CNN & classification by SVM |
| UMN | 7003 | 99.1860 | 99.4573 | 0.9991 | 0.9994 |
| UCSDPed1 | 14000 | 86.4929 | 88.9571 | 0.9455 | 0.9437 |
| Violent Flows | 21800 | 99.50 | 99.5459 | 0.9996 | 0.9986 |
| Subway Entrance | 143996 | 99.3291 | 99.5055 | 0.9967 | 0.9969 |
| Combined dataset | 186799 | 98.78 | 98.9122 | 0.9976 | 0.9978 |

Figure 4.13 and Figure 4.14 represent the ROC curves for the combined dataset using CNN and SVM classifiers with 20% test data and 30% test data, respectively. Figure 4.15 represents the ROC curves for the combined dataset using CNN and SVM classifiers with 10-fold cross-validation.



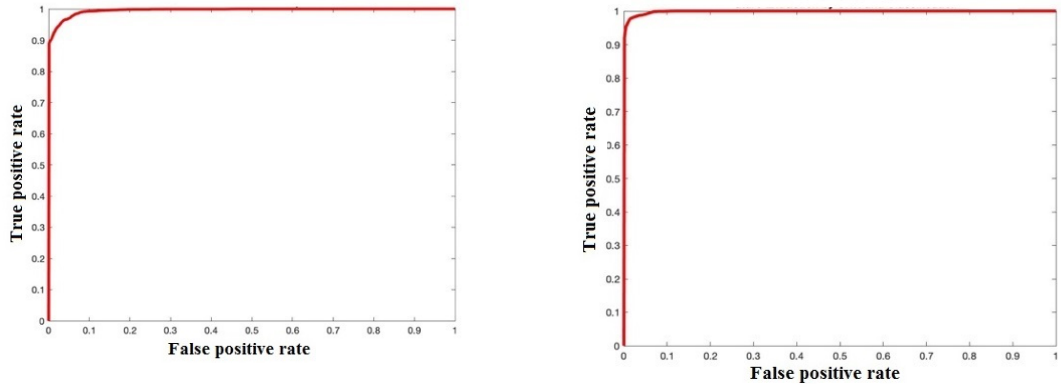

**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.13 ROC curves for the combined dataset with 20% test data**




**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.14 ROC curves for the combined dataset with 30% test data**




**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.15 ROC curves for the combined dataset with 10-fold cross-validation**

Confusion matrix illustrates the comparison of the values like true positive, false positive, true negative and false negative. Table 4.10 and Table 4.11 represent the confusion matrices for the four datasets using CNN and SVM classifiers, respectively, with 20% test data. Similarly, Table 4.12 and Table 4.13 represent the confusion matrices with 30% test data, and Table 4.14, and Table 4.15 represent the confusion matrices with 10-fold cross-validation.

Here Ab and N represent the abnormal and the normal class labels, respectively. For 20% and 30% test data, the confusion matrix represents the number of test data images. For 10-fold cross-validation, the experiment is done 10 times and the cumulative matrix is the final confusion matrix so it represents the total number of images of the dataset.

**TABLE 4.10 Confusion matrices for various datasets using CNN classifier with 20% test data**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 224 | 3 | 808 | 1 | 2463 | 5 | 751 | 1 | 4243 | 13 |
| | N | 4 | 1169 | 2 | 1989 | 1 | 1891 | 3 | 28044 | 65 | 33038 |

**TABLE 4.11 Confusion matrices for various datasets using SVM classifier with 20% test data**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 224 | 3 | 808 | 1 | 2466 | 2 | 751 | 1 | 4229 | 27 |
| | N | 5 | 1168 | 2 | 1989 | - | 1892 | 1 | 28046 | 18 | 33085 |

**TABLE 4.12 Confusion matrices for various datasets using CNN classifier with 30% test data**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 341 | - | 1212 | 2 | 3126 | 576 | 285 | 843 | 4141 | 2243 |
| | N | 15 | 1745 | 6 | 2980 | 374 | 2464 | 9857 | 32214 | 852 | 48803 |

**TABLE 4.13 Confusion matrices for various datasets using SVM classifier with 30% test data**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 341 | - | 1212 | 2 | 2992 | 710 | 329 | 799 | 4817 | 1567 |
| | N | 21 | 1739 | 6 | 2980 | 360 | 2478 | 10171 | 31900 | 2029 | 47563 |

**TABLE 4.14 Confusion matrices for various datasets using CNN classifier with 10-fold cross-validation**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 1092 | 45 | 2925 | 1120 | 12279 | 60 | 2984 | 776 | 19487 | 1794 |
| | N | 12 | 5854 | 771 | 9184 | 49 | 9412 | 190 | 140046 | 485 | 165033 |

**TABLE 4.15 Confusion matrices for various datasets using SVM classifier with 10-fold cross-validation**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | Combined dataset | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 1110 | 27 | 3192 | 853 | 12278 | 61 | 3118 | 642 | 19652 | 1629 |
| | N | 11 | 5855 | 693 | 9262 | 38 | 9423 | 70 | 140166 | 403 | 165115 |

The various confusion matrices represent that very few images are misclassified. The reason for getting such accurate results is the tuning of parameters and hyperparameters of the model. If the number of layers is taken less or more than the number described in the proposed architecture, performance decreases. The number of filters, filter size and learning rate value also affect much to the model performance. Many experiments are done using the various parameter and hyperparameter values and best values are selected to propose the perfect architecture. Therefore the same model gives the best performance on all the datasets without altering parameter or hyperparameter values. The set parameters are as follows.

Optimizer : SGDM
Initial Learning Rate : 0.01
Learning Rate Drop Factor : 0.2
Learning Rate Drop Period : 5
Number of Epochs : 5
Batch size : 64

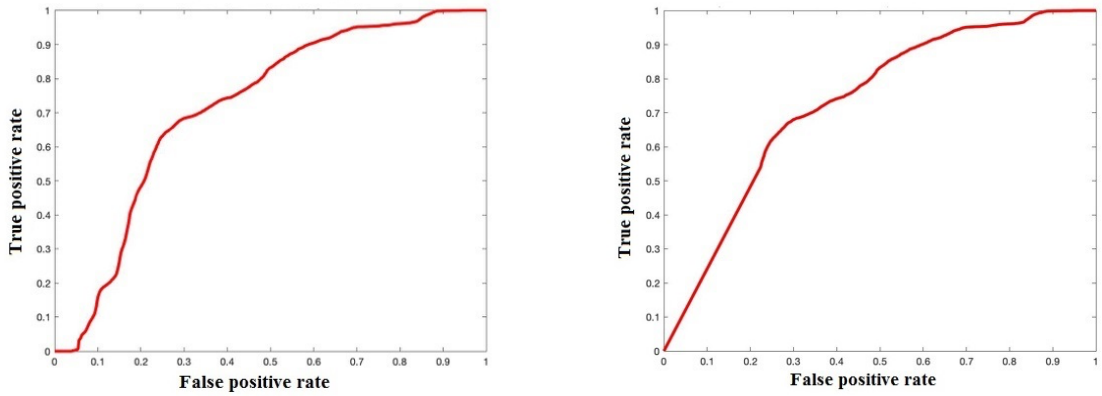
For abnormal event detection using the UCFCrime2Local dataset, all images of various categories except for ‘Normal’ are considered abnormal event images. The achieved accuracy for detection as per the given train-test split is 74.38%, and the AUC value is 0.7163 for CNN classifier and accuracy is 71.16% and AUC value is 0.7303 for SVM classifier. Table 4.16 and Table 4.17 represent the confusion matrices for the UCFCRime2Local dataset using the proposed CNN Architecture2 with CNN and SVM classifiers respectively. Figure 4.16 shows the ROC curves for the same.

**TABLE 4.16 Confusion matrix for the UCFCrim2Local dataset using CNN classifier**

| True class | | Predicted class | |
|---|---|---|---|
| | | **Ab** | **N** |
| | **Ab** | 4180 | 9836 |
| | **N** | 9754 | 52708 |

**TABLE 4.17 Confusion matrix for the UCFCrim2Local dataset using SVM classifier**

| True class | | Predicted class | |
|---|---|---|---|
| | | **Ab** | **N** |
| | **Ab** | 9171 | 4845 |
| | **N** | 17208 | 45254 |




**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.16 ROC curves for abnormal event detection using the UCFCrime2Local dataset**

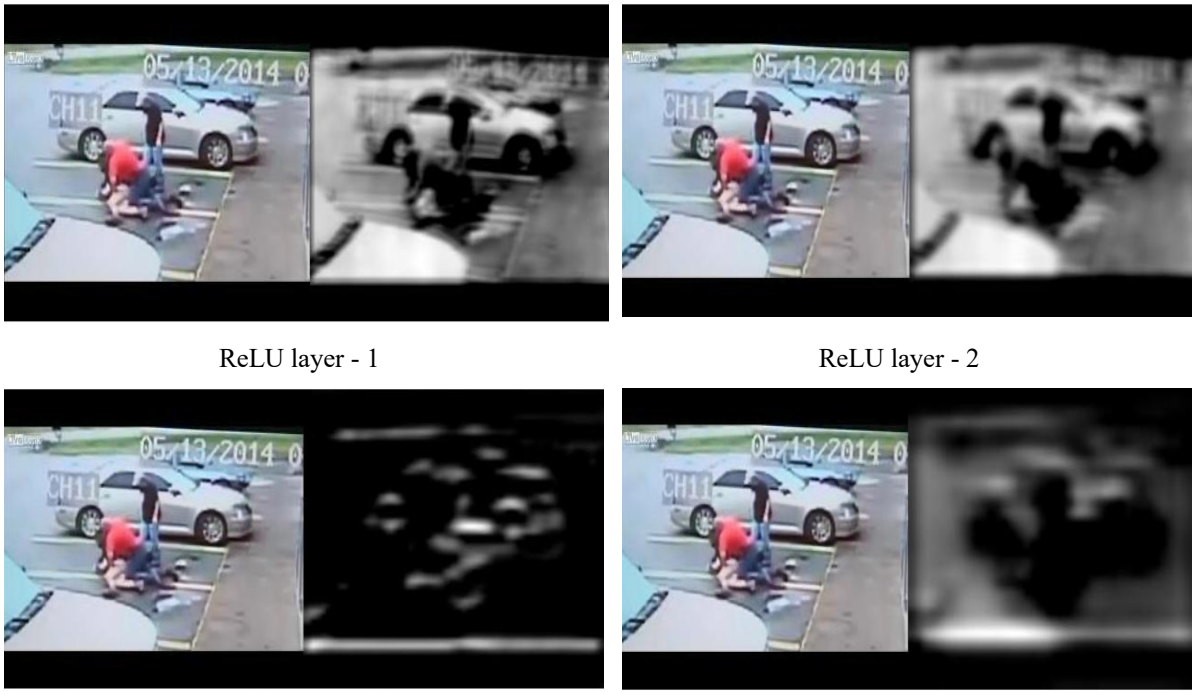
The same model is used for classification of various abnormal events. All datasets mentioned in section 1.4 except for the UCFCrime2Local, have only two types of events: normal or abnormal. UCFCrime2Local dataset contains seven different events: Arrest, Assault, Burglary, Normal, Robbery, Stealing and Vandalism. So classification of abnormality is done using only this dataset. Experiments for classification are done in three ways. In the first one, video frames are taken as per the given train-test split. For the other two ways, all images of a particular event's videos are combined according to class labels. Now using all images, two types of experiments are done (1) Non-stratified holdout validation with 80% training and 20% test data and (2) 5-fold cross-validation. Figure 4.17, Figure 4.18, Figure 4.19 and Figure 4.20 show the largest activation at four

convolution layers, RELU layers, Max pooling layers and Batch normalization layers, respectively for the sample image of the arrest event.


Convolution layer - 1 Convolution layer - 2

Convolution layer - 3 Convolution layer - 4

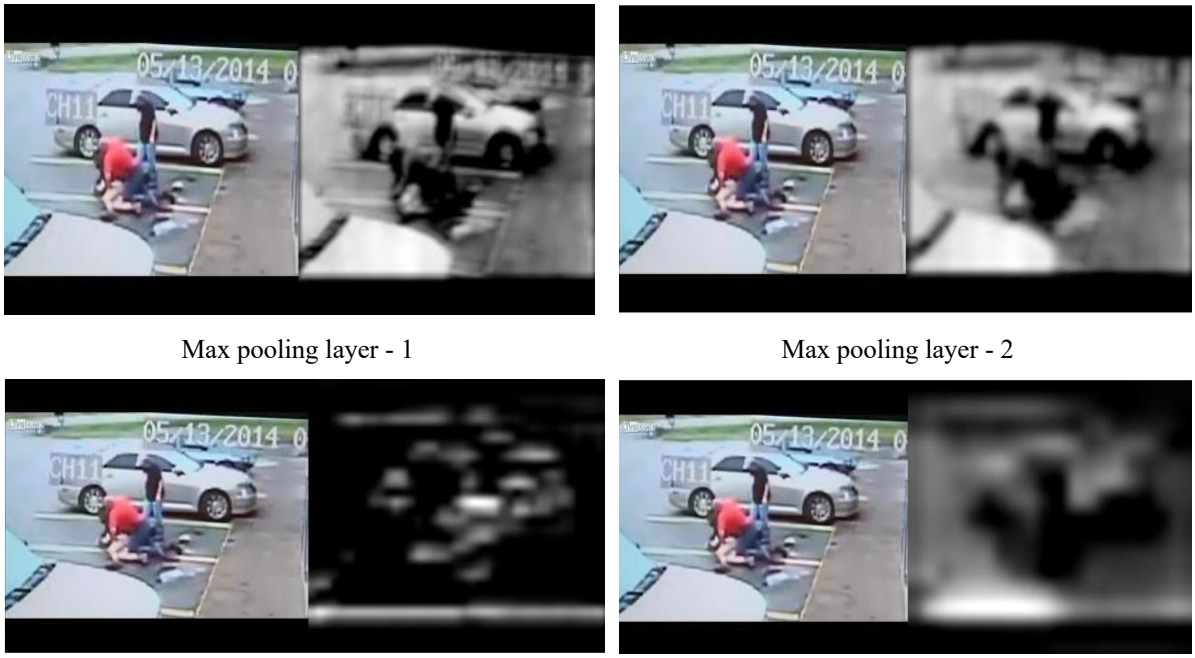
**FIGURE 4.17 The largest activation at each convolution layer for the arrest event**




ReLU layer - 1 ReLU layer - 2

ReLU layer - 3 ReLU layer - 4

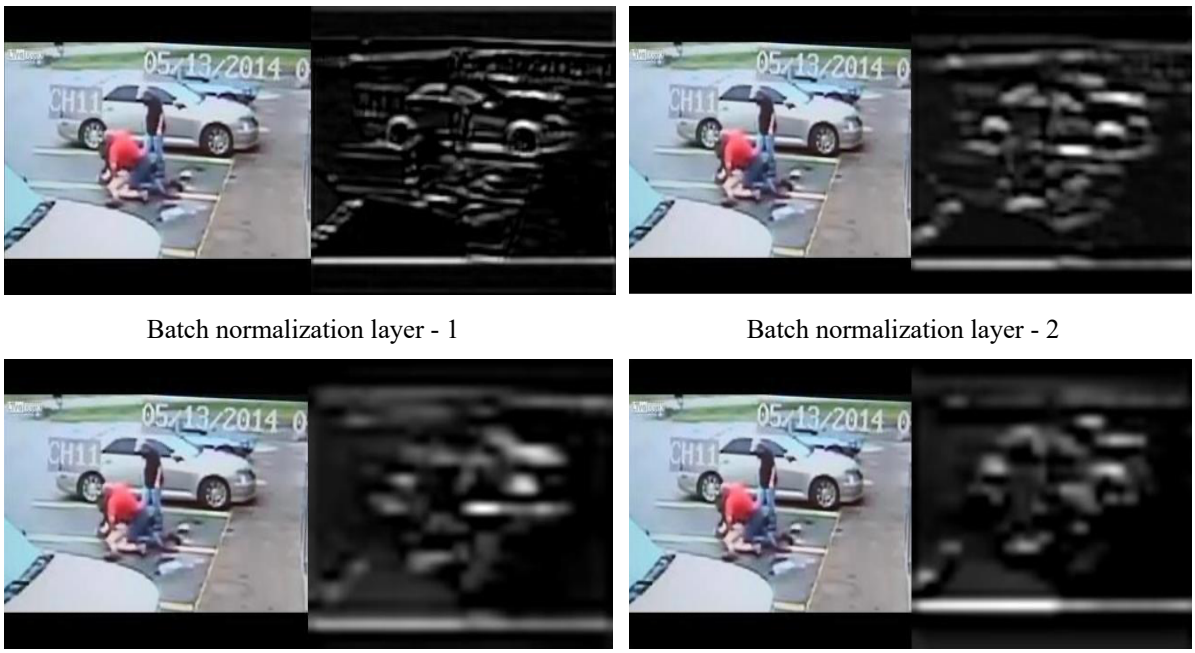
**FIGURE 4.18 The largest activation at each ReLU layer for the arrest event**



Max pooling layer - 1 Max pooling layer - 2

Max pooling layer - 3 Max pooling layer - 4

**FIGURE 4.19 The largest activation at each max pooling layer for the arrest event**



Batch normalization layer - 1 Batch normalization layer - 2

Batch Normalization layer - 3 Batch normalization layer - 4

**FIGURE 4.20 The largest activation at each batch normalization layer for the arrest event**

Table 4.18, Table 4.19 and Table 4.20 represent the achieved AUC values for classification using the given train-test split, 80% training data and 5 fold cross-validation, respectively. Table 4.21 shows overall classification accuracy for each experiment. For 80% training data and 5-fold cross-validation performance is better compared to the given train-test split

because few images of each specific event's video are present in the training set. So it can be concluded that if the model is used for a particular place and is trained by standard background images, it gives outstanding classification results.

**Table 4.18 Classification results using the UCFCrime2Local dataset as per given train-test split**

| Event | Number of images | Area Under Curve | |
|---|---|---|---|
| | | Classification by CNN | Classification by SVM |
| Arrest | 8739 | 0.7061 | 0.6755 |
| Assault | 5560 | 0.7276 | 0.6928 |
| Burglary | 7841 | 0.5160 | 0.3161 |
| Normal | 203272 | 0.7115 | 0.5955 |
| Robbery | 8650 | 0.6263 | 0.6717 |
| Stealing | 8668 | 0.6323 | 0.4976 |
| Vandalism | 8421 | 0.5845 | 0.5427 |

**Table 4.19 Classification results using the UCFCrime2Local dataset with 20% test data**

| Event | Number of images | Area Under Curve | |
|---|---|---|---|
| | | Classification by CNN | Classification by SVM |
| Arrest | 8739 | 0.9876 | 0.9949 |
| Assault | 5560 | 0.9991 | 0.9991 |
| Burglary | 7841 | 0.8335 | 0.8130 |
| Normal | 203272 | 0.8852 | 0.8423 |
| Robbery | 8650 | 0.8943 | 0.8712 |
| Stealing | 8668 | 0.9687 | 0.9158 |
| Vandalism | 8421 | 0.7346 | 0.7602 |

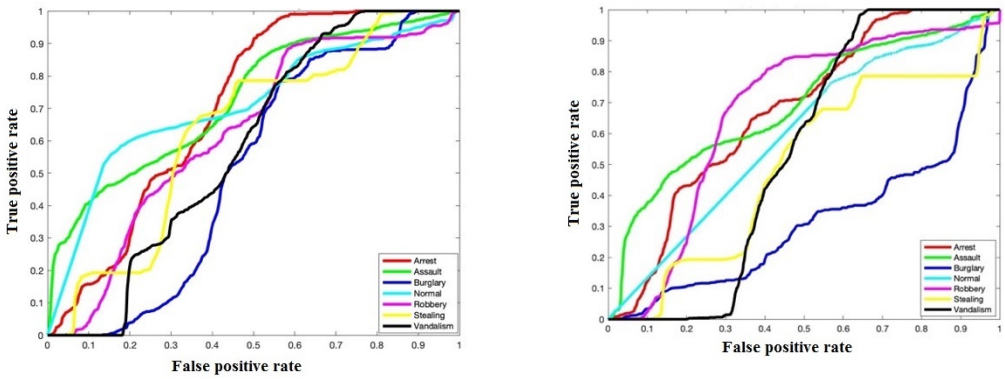
Figure 4.21, Figure 4.22 and Figure 4.23 represent the ROC curves for classification using the given train-test split, 80% training data and 5-fold cross-validation, respectively. In Figure 4.23, only one curve is visible because for each category, the achieved AUC value is 1.

**Table 4.20 Classification results using the UCFCrime2Local dataset with 5-fold cross-validation**

| Event | Number of Images | Area Under Curve | |
|---|---|---|---|
| | | Classification by CNN | Classification by SVM |
| Arrest | 8739 | 1 | 1 |
| Assault | 5560 | 1 | 1 |
| Burglary | 7841 | 1 | 1 |
| Normal | 203272 | 1 | 1 |
| Robbery | 8650 | 1 | 1 |
| Stealing | 8668 | 1 | 1 |
| Vandalism | 8421 | 1 | 1 |

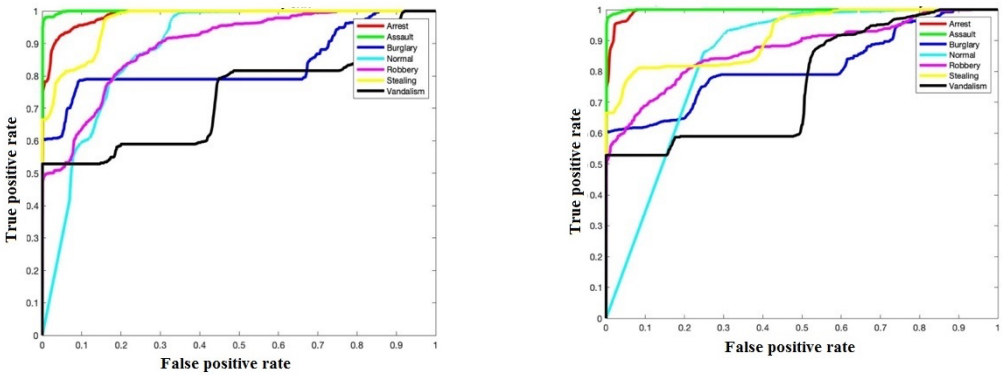
**Table 4.21 Overall classification accuracy**

| Dataset partition | Accuracy (%) | |
|---|---|---|
| | Classification by CNN | Classification by SVM |
| As per the train-test split | 71.79 | 70.20 |
| 80% training data | 87.29 | 88.13 |
| 5 fold cross-validation | 99.9924 | 99.9944 |



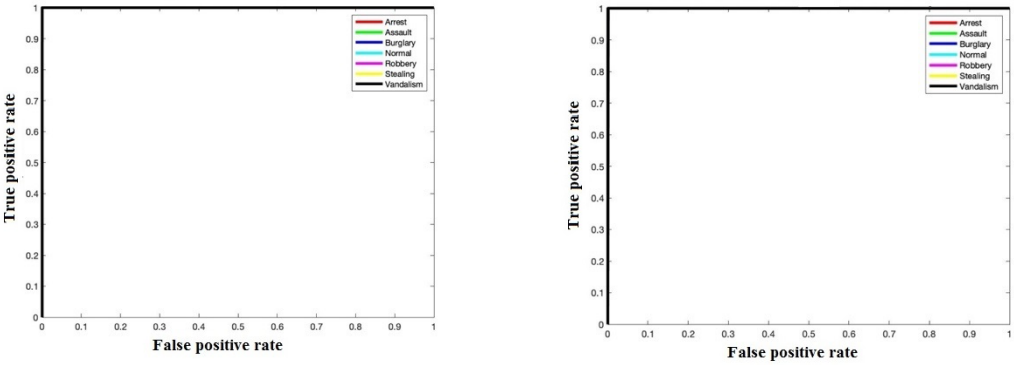

**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.21 ROC curves for events classification using the UCFCrime2Local dataset as per given train-test split**




**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.22 ROC curves for events classification using the UCFCrime2Local dataset with 20% test data**




**(a) Using CNN classifier** **(b) Using SVM classifier**

**FIGURE 4.23 ROC curves for events classification using the UCFCrime2Local dataset with 5-fold cross-validation**

## 4.4 CNN-LSTM Based Approach

### 4.4.1 LSTM Network

LSTM is a Recurrent Neural Network (RNN) architecture introduced by Sepp Hochreiter et al. [75]. It remembers values over arbitrary intervals. RNN is used to classify, process and predict time series data. Traditional RNN can remember short duration past sequence. If output depends on the past long duration sequence, it fails as it forgets the starting point. In contrast, the LSTM is insensitive to gap length. It can remember past long sequence. The following Figure 4.24 shows the basic architecture of the LSTM cell.

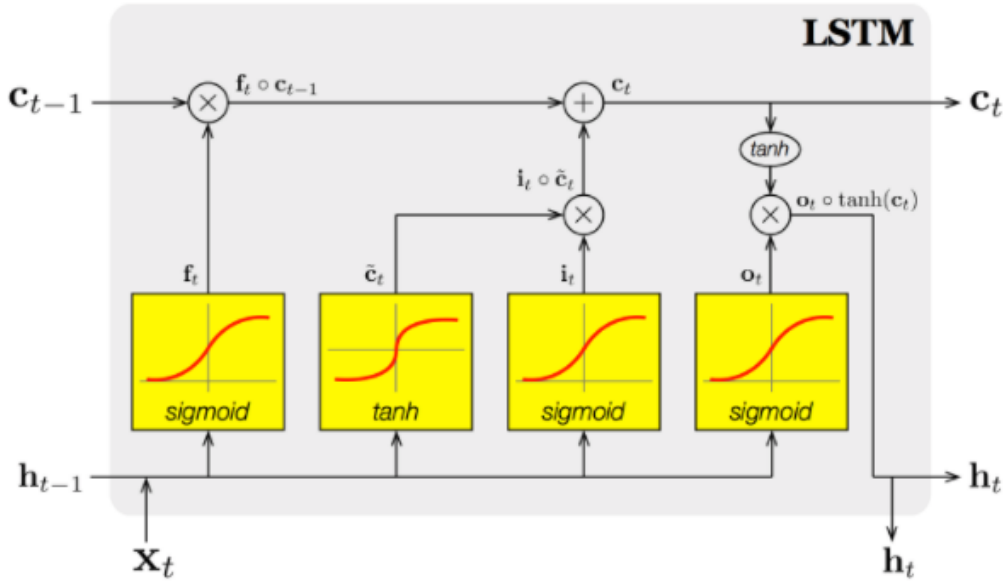



**FIGURE 4.24 LSTM architecture [76]**

The cell is a repeated module that contains three different gate structures, forget gate, input gate and output gate. In the shown Figure, c, h and x represent cell state, hidden state and input, respectively. The first sigmoid activation function is the forget gate. It defines which information should be omitted from the previous cell state ($C_{t-1}$). Sigmoid function outputs a number between 0 and 1. If output is 1 then the whole information is passed, and if the output is 0, then nothing is passed from the previous cell state to the current cell state. The second sigmoid and first tanh activation function represents the input gate. The sigmoid function decides which values will be updated, and the tanh function creates a new candidate value that could be added to the state. The last sigmoid is the output gate and highlights which information should be going to the next hidden state. As the tanh function outputs values between -1 and 1, the cell state is represented by tanh activation function and multiplied with the output of the last sigmoid function, so the final result of the output gate contains decided information.

### 4.4.2 BiLSTM Network

Bi-directional long short term memory network is known as BiLSTM network. It flows the sequence information in both directions forward and backward. Figure 4.25 shows BiLSTM Network. It preserves the future and the past information, therefore it is usually

employed where the sequence to sequence tasks are needed. This kind of network can be used in text classification, speech recognition and forecasting models.

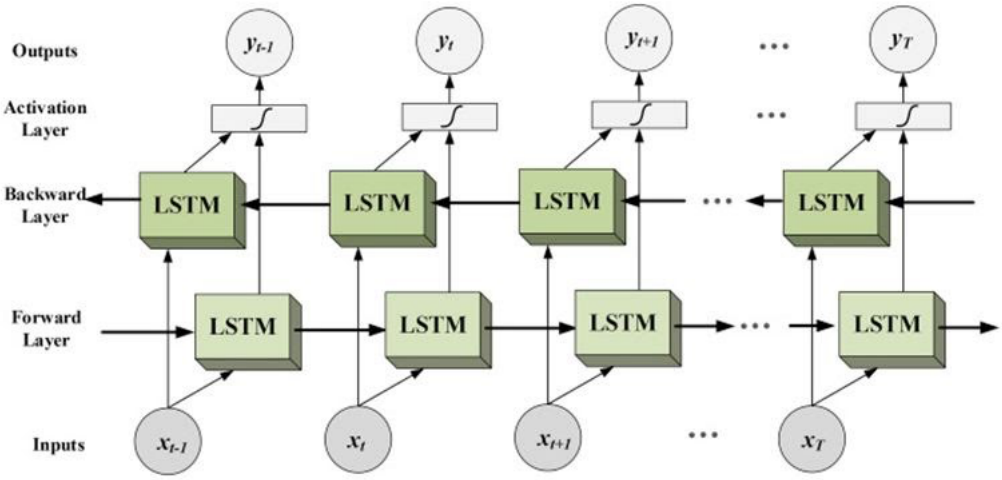



**FIGURE 4.25 BiLSTM network [85]**

### 4.4.3 CNN-LSTM Architecture

The CNN based approach works only in the spatial domain, i.e. images. To consider the temporal domain with spatial domain, the CNN-LSTM architecture is used, as shown in Figure 4.26.

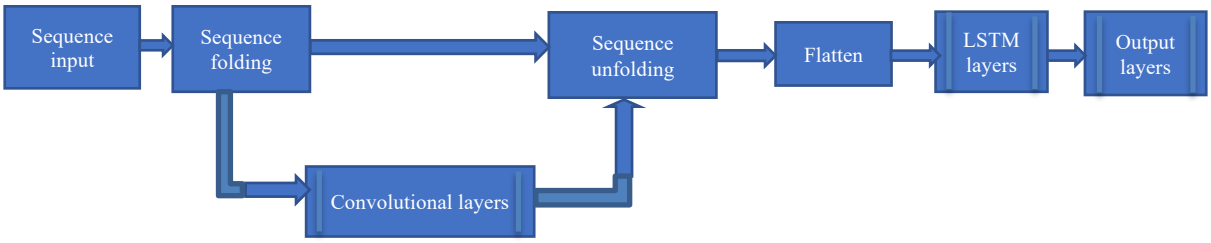



**FIGURE 4.26 CNN-LSTM architecture**

The sequences of feature vectors are generated from the input video sequences using the proposed Convolutional Neural Network Architecture2 (As shown in section 4.3.3.2). Here BiLSTM model is used to consider input for both forward and backward directions. The detailed description of the proposed LSTM architecture is given in the Table 4.22. The LSTM network is trained using the feature sequences with tuned parameters as follows.

Number of Nodes : 500

Optimizer: SGDM

Initial Learning Rate : 0.01

Batch size : 8

Number of Epochs : 5

**TABLE 4.22 The proposed LSTM architecture**

| Sr No | Layer name | Operation | Description |
|---|---|---|---|
| 1. | 'sequence' | Sequence input | Sequence input with 1024 dimensions |
| 2. | 'bilstm' | BiLSTM | BiLSTM with 500 hidden units |
| 3. | 'fc' | Fully connected | 2 fully connected layer |
| 4. | 'softmax' | Softmax | softmax |
| 5. | 'classification' | Classification output | crossentropyex with classes 'Abnormal' and 'Normal' |

### 4.4.4 Experimental Results

Table 4.23 represents the abnormal event detection results on various datasets using 80% training data except for the UCFCrime2Local dataset. For the UCFCrime2Local dataset, the experiment is performed as per the given train-test split.

**TABLE 4.23 Experimental results for abnormal event detection using the CNN-LSTM architecture**

| Dataset | Number of video sequences | Accuracy (%) | Area Under Curve |
|---|---|---|---|
| UMN | 22 | 100 | 1 |
| UCSDPed1 | 70 | 100 | 1 |
| Violent-Flows | 246 | 98 | 1 |
| Subway Entrance | 152 | 96.77 | 0.9846 |
| UCFCrime2Local | 300 | 73.33 | 0.79 |

Table 4.24 represents confusion matrices for abnormal event detection using various datasets. Table 4.25 represents classification results on the UCFCime2Local dataset in which the AUC value for each event is represented. Figure 4.27 shows the ROC curves for the same. The achieved overall classification accuracy is 73.86%. The UCFCrime2Local is a recently published complex dataset, and fewer experiments are done using it by researchers. Available state-of-the-art methods, using this dataset, are for abnormality

detection, not for classification. Although it is a complex dataset, the proposed model performs well on it.

**TABLE 4.24 Confusion matrices for various datasets using the CNN-LSTM architecture**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | UCFCrim2Local | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 2 | - | 5 | - | 27 | 1 | 2 | 1 | 18 | 13 |
| | N | - | 2 | - | 9 | - | 22 | 2 | 26 | 11 | 48 |

**TABLE 4.25 Experimental results for events classification using the CNN-LSTM architecture**

| Event | Area Under Curve |
|---|---|
| Arrest | 0.7471 |
| Assault | 0.7153 |
| Burglary | 0.6024 |
| Normal | 0.8387 |
| Robbery | 0.7349 |
| Stealing | 0.7674 |
| Vandalism | 0.7929 |

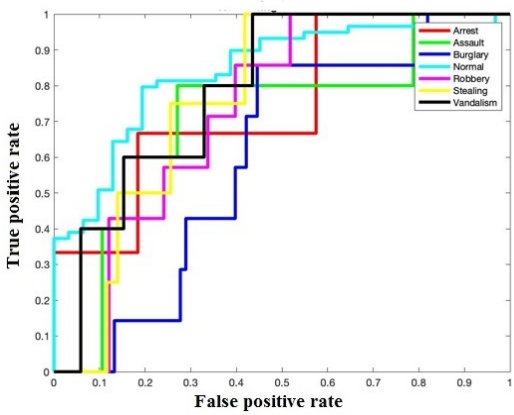



**FIGURE 4.27 ROC curves for classification results using the CNN-LSTM architecture**

## 4.5 Comparison of the Proposed Approaches

This section represents the comparison of the proposed approaches for abnormal event detection and classification. Figure 4.28 shows the performance of the proposed supervised deep learning based approaches for abnormal event detection. Horizontal axis represents

datasets and vertical axis represents the AUC values. Figure represents that the CNN and CNN-LSTM based approaches provides excellent results using various datasets.

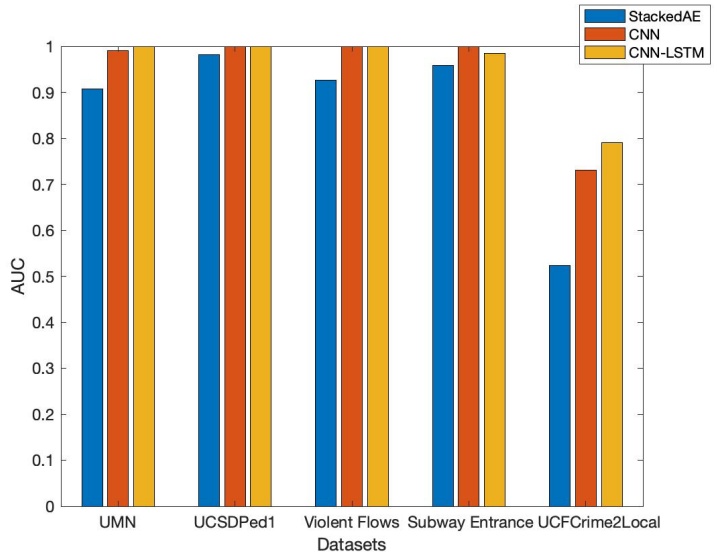



**FIGURE 4.28 Performance of the proposed approaches for abnormal event detection**

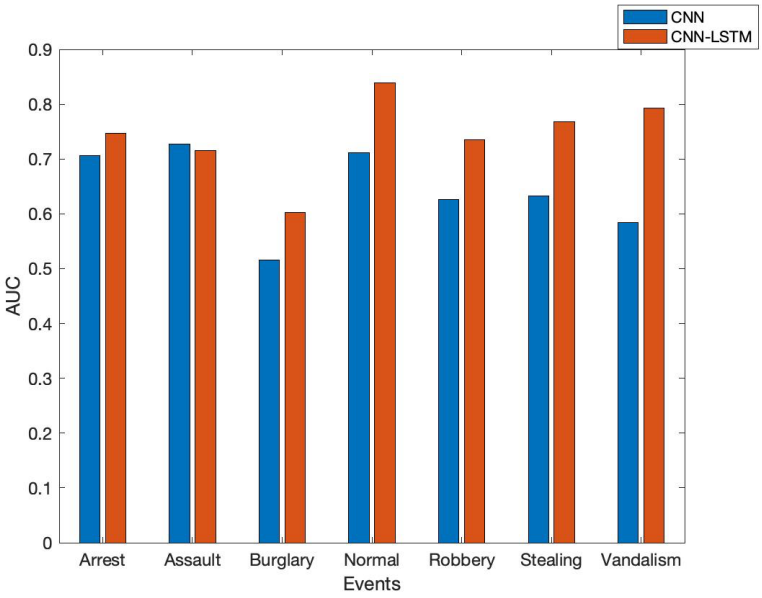



**FIGURE 4.29 Performance of the proposed approaches for events classification**

Figure 4.29 represents the performance of the CNN and CNN-LSTM based approaches for events classification. Horizontal axis represents events and vertical axis represents the AUC values. The Figure represents that the CNN-LSTM based approach provides higher AUC values for various events except for assault event.

## 4.6 Concluding Remarks

This chapter represents three approaches based on supervised deep learning algorithms. Stacked autoencoder based approach for abnormal event detection while, CNN and CNN-LSTM based approaches for abnormal event detection and classification. In the first approach the stacked autoencoder architecture is proposed with two encoders and one softmax layer. Backpropagation algorithm is used for fine tuning. This approach works on spatial domain. The achieved accuracy is 96.9% and AUC value is 0.9079 for UMN dataset. The achieved accuracy is 98.6% and AUC value is 0.9811 for UCSDPed1 dataset. The achieved accuracy is 93.1% and AUC value is 0.9269 for Violent Flows dataset. The achieved accuracy is 99.6% and AUC value is 0.9586 for Subway Entrance dataset. The achieved accuracy is 81.4% and AUC value is 0.5229 for UCFCrime2Local dataset.

For the CNN based approach, image classification network GoogLeNet is retrained which provides 86.21% Accuracy and AUC value 0.9487 for the UMN dataset. Even for small and simple dataset, transfer learning doesn't provide sufficient performance. So, a new CNN architecture is developed from scratch i.e. CNN Architecture1 which includes 14 layers. This architecture provides 99.64% accuracy and AUC value 0.9999 for UMN dataset. It provides 97.28% accuracy and AUC value 0.9774 for Airport-WrongDir dataset.

The CNN Architecture1 provides adequate results for small and simple datasets but not for complex datasets. Therefore the other new architecture is developed i.e. CNN Architecture2. This approach is used by two ways. In the first one the CNN is used for feature extraction and classification. In the second one, the CNN is used for feature extraction and SVM is used for classification. The experiments are done using various partitions of data and also with combined dataset which includes all images of UMN, UCSDPed1, Violent Flows and Subway Entrance datasets. The achieved accuracy is 99.79% and AUC value is 1 using the combined dataset with CNN classifier for 20% test data. The achieved accuracy is 99.88% and AUC value is 0.9999 using the combined dataset with SVM classifier for 20% test data.

The proposed CNN Architecture2 provides appropriate results using UCFCrime2Local

dataset for detection and classification of various events. The achieved accuracy for detection as per the given train-test split is 74.38%, and the AUC value is 0.7163 using CNN classifier and accuracy is 71.16% and AUC value is 0.7303 using SVM classifier. For classification, the experiments are done by three different ways. In the first one, the dataset is used as per the given train test split which includes different event videos with different background for training and testing. The achieved accuracy as per train test split is 71.79 % using CNN classifier and 70.20% using SVM classifier. For the other two ways, all images of a particular event's videos are combined according to class labels. Then experiments are done with 80% training and 20% test data and 5-fold cross-validation. The achieved accuracy with 20% test data is 87.29% using CNN classifier and 88.13 % using SVM classifier. The achieved accuracy with 5-fold cross-validation is 99.9924% using CNN classifier and 99.9944 % using SVM classifier.

To consider temporal features with spatial features, the CNN-LSTM architecture is proposed which is formed using the proposed CNN Architecture2. The achieved accuracy for abnormal event detection is 100%, 100% 98%, 96.77% and 73.33% using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively. The achieved AUC values for abnormal event detection are 1, 1, 1, 0.9846 and 0.79 using UMN, UCSDPed1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets respectively.

For classification, the achieved AUC values using CNN-LSTM architecture for Arrest, Assault, Burglary, Normal, Robbery, Stealing and Vandalism are 0.75, 0.72, 0.60, 0.84, 0.73, 0.77 and 0.79, respectively. The achieved overall classification accuracy is 73.86%.

The CNN-LSTM approach gives adequate results on the weakly labelled challenging dataset UCFCrime2Local. For classification, this approach provides higher AUC values for various events except for assault event.

# CHAPTER - 5

# Approaches Based on Unsupervised Deep Learning Algorithms

In chapter 3 and chapter 4, supervised learning based approaches are discussed for abnormal events detection and classification. In real world applications, it may not possible to generate labels of each type of abnormal event. For generalized abnormal event detection, unsupervised learning based approach can be applied. As discussed in section 3.2.1, autoencoders are used to reconstruct the input. So here autoencoders are trained using only normal events. For test data normal and abnormal both types of events are used. In testing phase, reconstruction error for abnormal event is higher compared to normal event and based on this error value, abnormality can be detected. Here, experiments are done for spatial domain i.e. images and spatiotemporal domain i.e. videos. In this work two architectures are proposed : Convolutional Autoencoder for image data and LSTM Autoencoder for video data.

## 5.1 Convolutional Autoencoder Based Approach

### 5.1.1 Convolutional Autoencoder

The major drawback of simple sparse autoencoder is that it cannot extract 2D structure from image sequences, which is the key point to detect abnormal event. To address this issue convolutional autoencoder architecture seems to be more appropriate [64]. Convolutional autoencoders are widely used for image data because of their excellent performance. It can capture the 2D structure in image sequences. It can pull out the local information from the images, which is extremely appropriate in the abnormality detection context. In Convolutional autoencoder, the weights are shared among all locations in the input, preserving the spatial locality, similar to CNN [78].

The loss function is similar to the simple autoencoder, as presented in equation-5.1.

$$e(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 + \lambda (w)^2 \qquad (5.1)$$

where $\lambda$ is the regularization parameter for the regularization term $(w)^2$, which is used during the training procedure. The convolution filter extracts useful features as well as compress the input image. In convolution, multiple input activations within the fixed receptive field of a filter are connected to a single activation output in the feature map. For the input x, the hidden layer mapping of the $k^{th}$ feature map is given by equation 5.2.

$$h_k = \sigma( x * w_k + b_k ) \qquad (5.2)$$

where b and $\sigma$ are bias and an activation function respectively, while symbol $*$ represents the 2D convolution. The deconvolutional layer performs an inverse operation of the convolutional layer. The learned filters in the deconvolutional layers work as base to reconstruct the input image. The reconstruction is obtained using equation 5.3

$$y = \sigma [\sum_{k \in H} h_k * w'_k + c] \qquad (5.3)$$

where c is bias per input channel and H represents the group of latent feature maps. $w'$correlates with the flip operation over both dimensions of the weights w. Convolutional and deconvolutional layers can be assembled to build deep architectures. The initial layers extract low level features and later layers extract high-level features like motion and appearance.

### 5.1.2 The Proposed Architecture

The algorithm is as follows.

- Train convolutional autoencoder with normal event images of the training set.
- Calculate the mean squared error between the input images and reconstructed images as per following equation 5.4

$$MSE = \frac{1}{MN}\sum_{M,N}[I_i(m, n) - I_r(m, n)]^2 \qquad (5.4)$$

Where $I_i$ is input image and $I_r$ is reconstructed image with M and N number of rows and columns respectively.

- Calculate average mean squared error and set it as a threshold.
- For test data, use both normal and abnormal event images and calculate reconstruction error for each image.
- Consider the observations which have greater error value than the set threshold, as abnormal event images.
- To find AUC value, calculate abnormality score as following equation 5.5.

$$\text{Abnormality score} = (e(x)-\min e(x))/ \max e(x) \qquad (5.5)$$

where x is the output reconstructed image and e(x) is the reconstruction error.

Various complex autoencoder architectures are proposed by different researchers. In this work, as an encoder part convolutional layers and as a decoder part transposed convolution layers are used. As an activation function ReLU is used and weights and biases are initialized by Glorot initializer as discussed in section 4.3.3.1. The detailed description of the proposed architecture is shown in Table 5.1.

**TABLE 5.1 The proposed convolutional autoencoder architecture**

| Sr No | Layer name | Operation | Description |
|---|---|---|---|
| 1. | 'input' | Image input | 128×128×1 images with 'zerocenter' normalization |
| 2. | 'conv_1' | Convolution | 64 3×3×1 convolutions with stride [1 1] and padding 'same' |
| 3. | 'relu_1' | ReLU | ReLU |
| 4. | 'conv_2' | Convolution | 32 3×3×64 convolutions with stride [1 1] and padding 'same' |
| 5. | 'relu_2' | ReLU | ReLU |
| 6. | 'transposed conv_1' | Transposed convolution | 64 3×3×32 transposed convolutions with stride [1 1] and cropping 'same' |
| 7. | 'relu_3 | ReLU | ReLU |
| 8. | 'transposed-conv_2' | Transposed convolution | 1 3×3×64 transposed convolutions with stride [1 1] and cropping 'same' |
| 9. | 'relu_4' | ReLU | ReLU |
| 10. | 'regressionoutput' | Regression output | mean-squared-error with response 'Response' |

### 5.1.3 Experimental Results

Input image Reconstructed image Input image Reconstructed image

(a) UMN dataset (b) UCSDPed1 dataset

**FIGURE 5.1 Input and reconstructed images**

**TABLE 5.2 Reconstruction error for various datasets using ConvAE**

| Images | Reconstruction error | | |
|---|---|---|---|
| | **Min** | **Mean** | **Max** |
| **UMN** | | | |
| Normal images of training set | 282.9357 | 1.4786e+3 | 2.3981e+3 |
| Normal images of test set | 282.9458 | 1.4281e+3 | 2.2886e+3 |
| Abnormal images of test set | 677.8004 | 2.0092e+3 | 2.3996e+3 |
| **UCSDPed1** | | | |
| Normal images of training set | 29.1127 | 61.6919 | 195.1543 |
| Normal images of test set | 29.6090 | 60.2274 | 193.5565 |
| Abnormal images of test set | 32.7486 | 75.75 | 165.0304 |
| **Violent Flows** | | | |
| Normal images of training set | 15.5072 | 339.1884 | 1.1688e+03 |
| Normal images of test set | 15.5072 | 334.5054 | 1.2877e+03 |
| Abnormal images of test set | 107.9339 | 446.5740 | 1.5290e+3 |
| **Subway Entrance** | | | |
| Normal images of training set | 0.6123 | 1.2918 | 14.3529 |
| Normal images of test set | 0.6187 | 1.2855 | 13.6877 |
| Abnormal images of test set | 0.7980 | 1.9244 | 4.4140 |
| **UCFCrime2Local** | | | |
| Normal images of training set | 27.4413 | 117.1915 | 585.0745 |
| Normal images of test set | 32.1611 | 110.9258 | 531.8320 |
| Abnormal images of test set | 46.3153 | 115.9604 | 235.93 |

Figure 5.1 shows the sample input and reconstructed images of the UMN and UCSDPed1 datasets. Table 5.2 shows the minimum, mean and maximum values of reconstruction

errors for various datasets. In the Table, it can be seen that mean error of abnormal images is greater than the mean error of normal images for each dataset except for the UCFCrime2Local. This dataset is weakly labelled and each video frame of abnormal video does not contain abnormal event. So the performance of the proposed architecture is lower on this dataset compared to others.

Table 5.3 shows the results of the proposed model using various datasets. For all datasets except for the UCFCrime2Local dataset, 80% data is used for training, and 20% data is used for testing. UCFCrime2Local dataset is used as per the given train-test split. For this algorithm, any modification in the set threshold value causes variation in the accuracy. Still, the AUC remains constant for all threshold values, so AUC can be considered the preferred performance measurement.

**TABLE 5.3 Experimental results for abnormal event detection using ConvAE**

| Dataset | Accuracy (%) | Area Under Curve |
|---|---|---|
| UMN | 85.79 | 0.8635 |
| UCSDPed1 | 72.18 | 0.7045 |
| Violent-Flows | 63.72 | 0.6661 |
| Subway Entrance | 95.49 | 0.8233 |
| UCFCrime2Local | 78.03 | 0.5967 |

## 5.2 LSTM Autoencoder Based Approach

The convolutional autoencoder based approach works only in the spatial domain. To consider the temporal domain with the spatial domain, a spatiotemporal autoencoder can be used. But it requires convolution in the spatial domain, convolution in the temporal domain, then deconvolution in the spatial domain and again deconvolution in the temporal domain. So it becomes complex. As discussed in section 4.4, the LSTM network performs well on sequence data i.e. videos. In this research, the approach based on the LSTM autoencoder is proposed.

Sequence prediction problems are challenging because the length of the input sequence can vary for different videos. An LSTM autoencoder is the autoencoder for sequence data. It uses Encoder-Decoder LSTM architecture and reconstructs the input sequence. Like convolutional autoencoder based approach, here also the training set contains only normal event feature sequences and test set contains both normal and abnormal event feature sequences.

### 5.2.1 The Proposed Approach

The CNN model extracts the best features from images. So, the feature sequences of input videos that the CNN model extracts are given as input to the LSTM autoencoder. The proposed algorithm is as follows.

- Extract features from video sequences using the CNN model and train LSTM autoencoder with the training set's normal event's feature sequences.
- Calculate mean squared error between the input feature sequences and reconstructed feature sequences as following equation 5.6

$$e = \frac{1}{xy}\sum_{k=1}^{x}\sum_{i=1}^{y}(B'_{ki} - B_{ki})^2 \qquad (5.6)$$

  where B' is the reconstructed feature sequence and B is the input feature sequence with x and y number of rows and columns respectively.
- Calculate average mean squared error and set it as the threshold.
- For test data, use both normal and abnormal event sequences and calculate the reconstruction error for each sequence.
- Consider the observations which have greater error value than the set threshold, as abnormal video sequences.
- The abnormality score which is used to find the AUC value is calculated as follows

$$\text{Abnormality score} = (e(x)-\min e(x))/ \max e(x) \qquad (5.7)$$

  where x is the reconstructed feature sequence and e(x) is the reconstruction error.

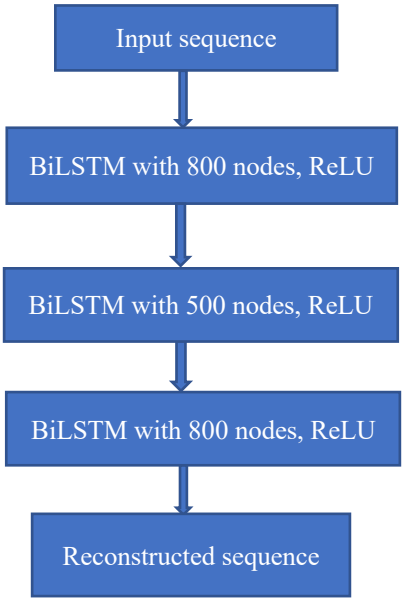



**FIGURE 5.2 The proposed LSTM autoencoder architecture**

Figure 5.2 shows the proposed LSTM autoencoder architecture. In this model, nine layers are used as described in Table 5.4. Here also bidirectional LSTM architecture is used. For feature extraction, two CNN models are used (1) Pretrained GoogLeNet CNN architecture and (2) the proposed CNN Architecture2 (as shown in Figure 4.9). Feature sequences are 2D arrays in which number of rows represents number of features in each video frame and number of columns represents number of video frames.

**TABLE 5.4 The proposed LSTM autoencoder architecture**

| Sr No | Layer name | Operation | Description |
|---|---|---|---|
| 1. | 'input' | Sequence Input | Sequence input with 1024 dimensions |
| 2. | 'bilstm_1' | BiLSTM | BiLSTM with 800 hidden units |
| 3. | 'relu_1' | ReLU | ReLU |
| 4. | 'bilstm_2' | BiLSTM | BiLSTM with 500 hidden units |
| 5. | 'relu_2' | ReLU | ReLU |
| 6. | 'bilstm_3' | BiLSTM | BiLSTM with 800 hidden units |
| 7. | 'relu_3' | ReLU | ReLU |
| 8. | 'fc' | Fully Connected | 1024 fully connected layer |
| 9. | 'output' | Regression Output | mean-squared-error with response 'Response' |

### 5.2.2 Experimental Results

The experiments are done using UMN, UCSDPed1, Violent Flows, Subway Entrance and UCFCrime2local datasets. For all datasets except for the UCFCrime2Local dataset, 80% of data is used as training, and 20% is used for testing. UCFCrime2Local dataset is used as per the given train-test split. As LSTM autoencoder is trained using normal video sequences, for the given abnormal video sequences in test data, the reported reconstruction error is higher compared to normal video sequences. Table 5.5 shows each dataset's minimum, mean, and maximum values of reconstruction errors.

**TABLE 5.5 Reconstruction error for various datasets using LSTMAE**

| Feature sequences | Reconstruction error | | | | | |
|---|---|---|---|---|---|---|
| | Features extracted by pre-trained GoogLeNet CNN Architecture | | | Features extracted by the proposed CNN Architecture2 | | |
| | Min | Mean | Max | Min | Mean | Max |
| **UMN** | | | | | | |
| Normal sequences of training set | 0.3875 | 0.5805 | 1.0184 | 0.0566 | 0.0980 | 0.1561 |
| Normal sequences of test set | 0.4252 | 0.4322 | 0.4392 | 0.0779 | 0.0819 | 0.0892 |
| Abnormal sequences of test set | 0.6503 | 0.7586 | 0.9624 | 0.1152 | 0.1813 | 0.2281 |
| **UCSDPed1** | | | | | | |
| Normal sequences of training set | 0.1845 | 0.2743 | 0.5525 | 0.0276 | 0.0713 | 0.2152 |
| Normal sequences of test set | 0.2168 | 0.2547 | 0.3298 | 0.0377 | 0.0764 | 0.1409 |
| Abnormal sequences of test set | 0.2967 | 0.3335 | 0.3999 | 0.0702 | 0.0929 | 0.1616 |
| **Violent Flows** | | | | | | |
| Normal sequences of training set | 0.4493 | 0.6809 | 1.1549 | 0.2194 | 0.4524 | 0.9963 |
| Normal sequences of test set | 0.5231 | 0.7619 | 0.9744 | 0.2621 | 0.5218 | 0.9721 |
| Abnormal sequences of test set | 0.4668 | 0.6694 | 1.1046 | 0.3009 | 0.4460 | 0.6869 |
| **Subway Entrance** | | | | | | |
| Normal sequences of training set | 0.0728 | 0.1636 | 0.4856 | 0.0209 | 0.0776 | 0.4563 |
| Normal sequences of test set | 0.0779 | 0.1643 | 0.5569 | 0.0270 | 0.1234 | 0.9656 |
| Abnormal sequences of test set | 0.1087 | 0.2568 | 0.3714 | 0.0700 | 0.1582 | 0.2946 |
| **UCFCrime2Local** | | | | | | |
| Normal sequences of training set | 0.3182 | 0.6590 | 1.4801 | 0.1733 | 0.3982 | 0.7534 |
| Normal sequences of test set | 0.32 | 0.6426 | 1.2631 | 0.2989 | 0.5079 | 0.8395 |
| Abnormal sequences of test set | 0.4469 | 0.7173 | 1.4669 | 0.3001 | 0.5168 | 0.8986 |

The reconstruction errors are calculated between feature values. They are less compared to the error between pixel intensity values of two images as represented in Table 5.2. It can be seen in Table 5.5 that every mean error of abnormal videos is greater than the mean error of normal videos for each dataset except for Violent Flows. The Violent Flows dataset contains short videos of the dense crowd, so it is difficult to differentiate normal and abnormal event sequences.

Table 5.6 shows experimental results. As discussed in section 5.1, for this approach also, AUC can be considered preferred performance measurement metric because the AUC remains constant for all threshold values. The performance of the proposed approach is good except for the Violent-Flows dataset, which contains a dense crowd. Table 5.7 and Table 5.8 represent confusion matrices for various datasets using the GoogLeNet and the proposed CNN Architecture2, respectively.

**Table 5.6 Experimental results for abnormal event detection using LSTMAE**

| Dataset | Accuracy (%) | | Area Under Curve | |
|---|---|---|---|---|
| | The proposed CNN Architecture2 | GoogLeNet CNN Architecture | The proposed CNN Architecture2 | GoogLeNet CNN Architecture |
| UMN | 100 | 100 | 1 | 1 |
| UCSDPed1 | 78.57 | 85.71 | 0.75 | 0.9394 |
| Violent-Flows | 54 | 54 | 0.4852 | 0.5 |
| Subway Entrance | 77.42 | 80.65 | 0.8274 | 0.8182 |
| UCFCrime2Local | 66.67 | 66.6667 | 0.5161 | 0.6288 |

**TABLE 5.7 Confusion matrices for various datasets using LSTMAE with GoogLeNet Architecture**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | UCFCrim2-Local | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 3 | - | 1 | 2 | 27 | - | 8 | 3 | 1 | 30 |
| | N | - | 2 | - | 11 | 23 | - | 3 | 17 | - | 59 |

**TABLE 5.8 Confusion matrices for various datasets using LSTMAE with the proposed CNN Architecture2**

| True class | | Predicted class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UMN | | UCSDPed1 | | Violent Flows | | Subway Entrance | | UCFCrim2-Local | |
| | | Ab | N | Ab | N | Ab | N | Ab | N | Ab | N |
| | Ab | 3 | - | 8 | - | 3 | 18 | 5 | 2 | 1 | 30 |
| | N | - | 2 | 3 | 3 | 5 | 24 | 5 | 19 | - | 59 |

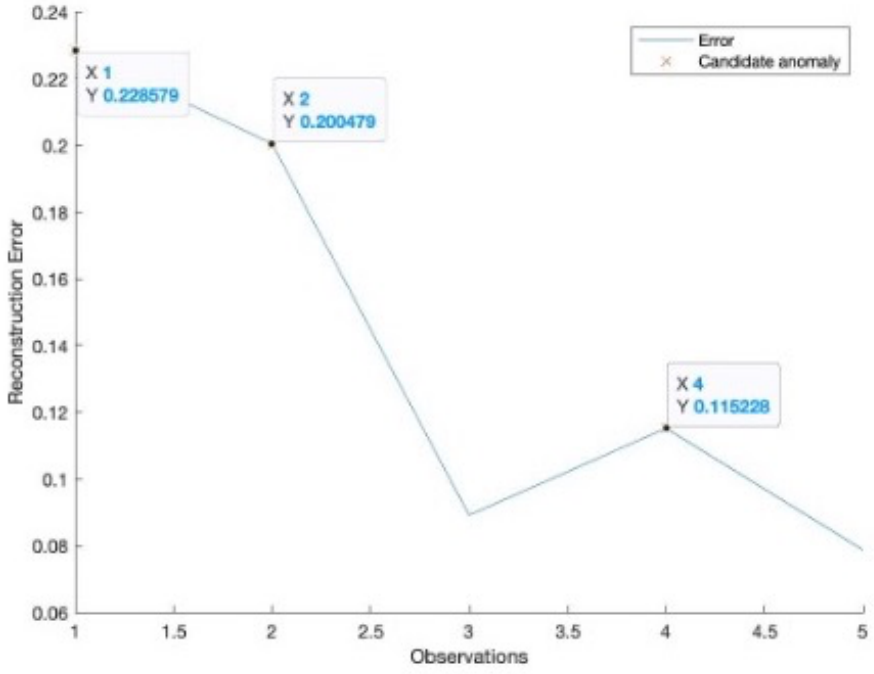



**FIGURE 5.3 Reconstruction error over the UMN test data using LSTMAE**

Figure 5.3 represents reconstruction error over the entire test data of the UMN dataset using the proposed CNN Architecture2. Here horizontal axis represents observations and vertical axis represents reconstruction error. As illustrated in Table 5.5 the average mean error for the UMN dataset's normal video sequences using the proposed CNN Architecture2 is 0.098. Figure 5.3 represents the detected abnormal events' reconstruction errors, which are greater than 0.098.

## 5.3 Comparison of the Proposed Approaches

Figure 5.4 represents the performance of the proposed unsupervised learning based approaches using various datasets. Horizontal axis represents datasets and vertical axis represents the AUC values. The LSTM autoencoder based approach performs better than the convolutional autoencoder based approach except for the Violent Flows dataset. Because of short videos of dense crowd the temporal domain consideration can't improve performance for Violent Flows dataset.

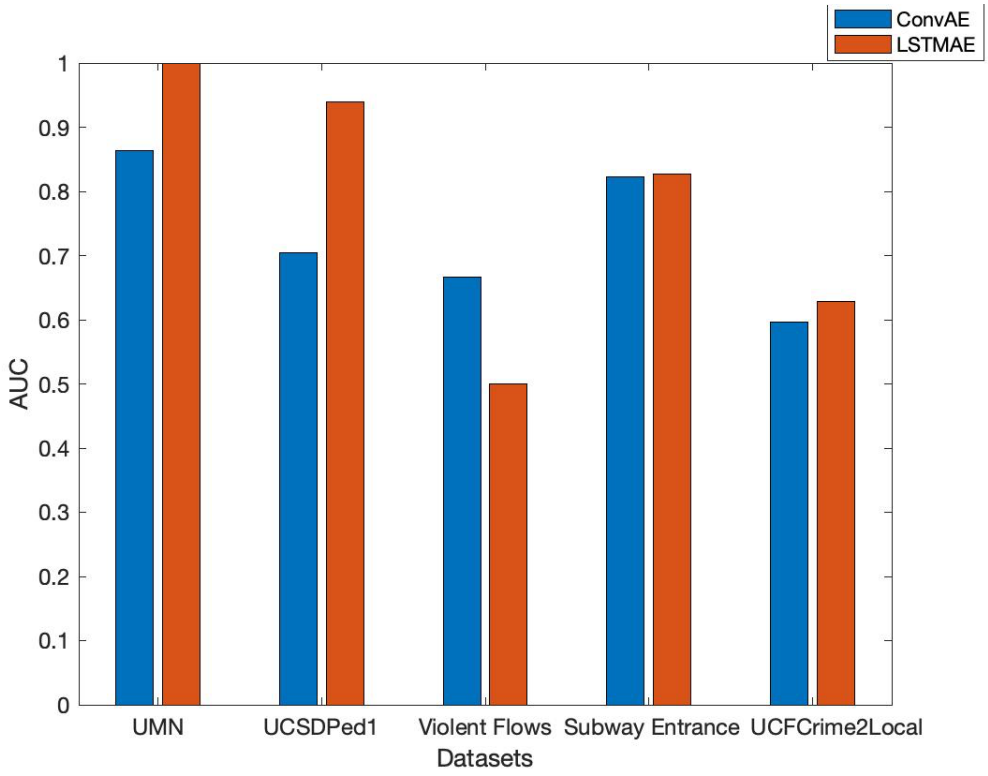



**FIGURE 5.4 Performance of the proposed approaches using various datasets**

## 5.4 Comparison of All the Proposed Approaches with State of the Art Methods

To evaluate the performance of the proposed methods, a comparison with state-of-the-art methods using the specific dataset is represented in this section. As mentioned in chapter 1,

six datasets are used in this research work. Table 5.9, Table 5.10, Table 5.11, Table 5.12 and Table 5.13 show the performance of the proposed and state-of-the-art methods using UMN, UCSDPed1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets respectively. Observable research work is not done using the Airport-WrongDir dataset so comparison is not possible.

**TABLE 5.9 Performance of various methods on the UMN dataset**

| Approach | Accuracy (%) | AUC |
|---|---|---|
| Pure optical flow [10] | - | 0.84 |
| Social force & bag of words [10] [13] | 85.09 | 0.96 |
| Bayesian model [11][13] | 96.40 | - |
| Force field model [13][15] | 81.04 | - |
| Chaotic invariants [13] [12] | 87.91 | 0.994 |
| Sparse reconstruction cost [13] [14] | 84.70 | 0.996 |
| Novel optical flow with one class SVM [13] | 96.46 | - |
| Optical flow + Bayes classification [21] | 91.57 | - |
| High-Frequency and Spatiotemporal (HFST) features + Hidden Markov model [21] | - | 0.90 |
| Behaviour entropy model[21] | - | 0.893 |
| Motion structure + CNN classifier [26] | 96.74 | - |
| Local binary pattern co-occurrence matrix and Histograms of the orientation of interaction force [20] | - | 0.99 |
| Energy model [9] | 91.66 | - |
| Histogram of oriented contextual gradient [18] | - | 0.993 |
| Commotion [23] | - | 0.98 |
| Plug and play CNN[25] | - | 0.98 |
| Generative Adversarial Nets (GAN)[22] | - | 0.99 |
| Abnormal event detection network [54] | - | 0.997 |
| Unmasking [39] | - | 0.951 |
| The proposed HOG and optical flow based method | 99.35 | - |
| The proposed autoencoder and optical flow based method | 100 | 1 |
| The proposed stacked autoencoder based method | 96.9 | 0.9079 |
| The proposed CNN architecture | 99.5 | 0.9999 |
| The proposed CNN LSTM architecture | 100 | 1 |
| The proposed convolutional autoencoder architecture | 85.79 | 0.8635 |
| The proposed LSTM autoencoder architecture | 100 | 1 |

**TABLE 5.10 Performance of various methods on the UCSDPed1 dataset**

| Approach | Accuracy (%) | AUC |
|---|---|---|
| Motion structure + CNN classifier [26] | 94.28 | - |
| Sparse reconstruction cost [14] | - | 0.487 |
| Mixture of Probabilistic Principal Component Analyzers (MPPCA) [41][25] | - | 0.59 |
| Social force[10][25] | - | 0.675 |
| SF+MPPCA [25] | - | 0.688 |
| Mixture of Dynamic Textures (MDT) [42] | - | 0.818 |
| Local statistical aggregates [43] | - | 0.927 |
| Video parsing [44] | - | 0.91 |
| Subspace [45] | - | 0.684 |
| Plug and play CNN [25] | - | 0.957 |
| Appearance and Motion DeepNet (AMDN) [46] | - | 0.921 |
| Generative Adversarial Nets (GAN) [22] | - | 0.974 |
| Sparse combination learning [24] | - | 0.918 |
| Gaussian Process Regression (GPR) [47] | - | 0.838 |
| Sparse cuboids [47] | - | 0.722 |
| Sparse inference based classifier [47] | - | 0.808 |
| Sparse spatiotemporal contextual information [47] | - | 0.779 |
| Dense spatiotemporal contextual information [47] | - | 0.899 |
| Unmasking [39] | - | 0.684 |
| Ensemble random projection [40] | - | 0.809 |
| Spatiotemporal autoencoder [36] | - | 0.899 |
| Convolutional LSTM autoencoder [35] | - | 0.43 |
| Variational autoencoder [35] | - | 0.63 |
| Recurrent convolutional autoencoder [35] | - | 0.694 |
| Two stream R-ConvVAE [35] | - | 0.75 |
| Histogram of orientation of optical flow [35] | - | 0.727 |
| Convolutional LSTM [35] | - | 0.67 |
| Adam[35] | - | 0.77 |
| The proposed stacked autoencoder based method | 98.6 | 0.9811 |
| The proposed CNN architecture | 99.89 | 1 |
| The proposed CNN LSTM architecture | 100 | 1 |
| The proposed convolutional autoencoder architecture | 72.18 | 0.7045 |
| The proposed LSTM autoencoder architecture | 85.71 | 0.9394 |

**TABLE 5.11 Performance of various methods on the Violent Flows dataset**

| Approach | Accuracy (%) | AUC |
|---|---|---|
| Local Trinary Patterns (LTP) [6] | 71.53 | 71.53 |
| Histogram of Oriented Gradients (HOG) [6] | 57.43 | 0.6182 |
| Histogram of Optical Flow (HOF) [6] | 58.53 | 0.5760 |
| HNF (Combination of HOG and HOF) [6] | 56.52 | 0.5994 |
| Violent Flows (ViF) [6] | 81.30 | 0.85 |
| ViF, OViF (Oriented Violent Flows), AdaBoost and SVM [14] | 88 | - |
| Simplified Histogram of Oriented Tracklets (SHOT) [48] | 82.2 | - |
| Jerk [49] | 74.2 | - |
| Interaction Force [49] | 74.5 | - |
| Histogram of Oriented Tracklets (HOT) [49] | 82.3 | - |
| Improved fisher vectors [49] | 96.4 | - |
| 3D CNN [50] | 98 | 0.98 |
| Convolutional LSTM [52] | 94.57 | |
| The proposed stacked autoencoder based method | 93.1 | 0.9269 |
| The proposed CNN architecture | 99.95 | 1 |
| The proposed CNN LSTM architecture | 98 | 1 |
| The proposed convolutional autoencoder architecture | 63.72 | 0.6661 |
| The proposed LSTM autoencoder architecture | 54 | 0.5 |

**TABLE 5.12 Performance of various methods on the Subway Entrance dataset**

| Approach | Accuracy (%) | AUC |
|---|---|---|
| Unmasking [39] | - | 0.706 |
| Sparse reconstruction cost [29] | - | 0.833 |
| Mixture of Dynamic Textures (MDT) [29] | - | 0.908 |
| Fully convolutional neural network [29] | - | 0.904 |
| Gaussian Process Regression (GPR) [47] | - | 0.927 |
| Sparse cuboids [47] | - | 0.889 |
| Sparse inference based classifier [47] | - | 0.91 |
| Sparse spatiotemporal contextual Information [47] | - | 0.917 |
| Dense spatiotemporal contextual Information [47] | - | 0.911 |
| Recurrent convolutional autoencoder (R-ConvAE) [35] | - | 0.821 |
| Multilevel anomaly detector [37] | - | 0.8234 |
| The proposed stacked autoencoder based method | 99.6 | 0.9586 |
| The proposed CNN architecture | 99.99 | 1 |
| The proposed CNN LSTM architecture | 96.77 | 0.9846 |
| The proposed convolutional autoencoder architecture | 95.49 | 0.8233 |
| The proposed LSTM autoencoder architecture | 77.42 | 0.8274 |

**TABLE 5.13 Performance of various methods on the UCFCrime2Local dataset**

| Approach | Accuracy(%) | AUC |
|---|---|---|
| Inflated 3D ConvNet (I3D) + Regression for whole frame [9] | - | 0.5612 |
| I3D + Regression for spatiotemporal tube [9] | - | 0.7473 |
| The proposed stacked autoencoder based method | 81.4 | 0.5229 |
| The proposed CNN architecture | 71.16 | 0.7303 |
| The proposed CNN-LSTM architecture | 73.33 | 0.79 |
| The proposed convolutional autoencoder architecture | 78.03 | 0.5967 |
| The proposed LSTM autoencoder architecture | 66.67 | 0.6288 |

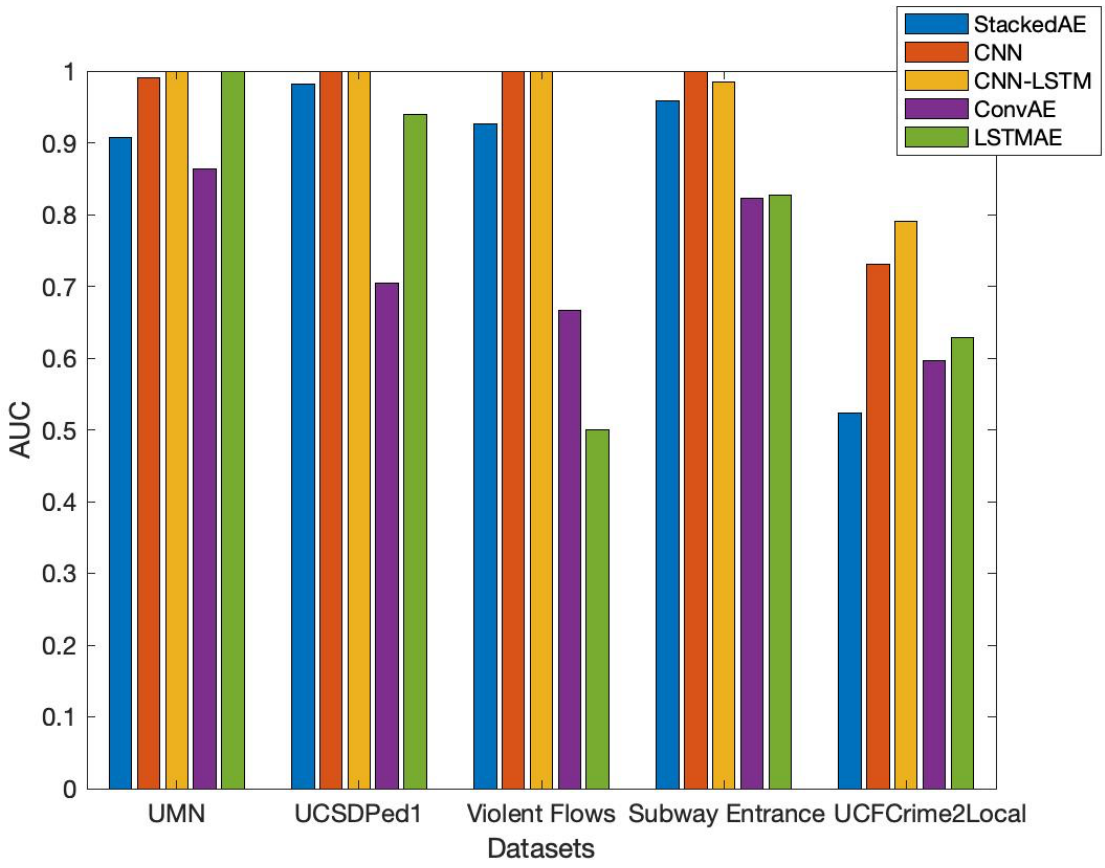



**FIGURE 5.5 Performance of the proposed deep learning based approaches**

Figure 5.5 shows the overall performance of the proposed deep learning based approaches using various datasets for abnormal event detection. In the Figure, horizontal axis represents datasets and vertical axis represents the AUC values. The Figure represents that the CNN and CNN-LSTM based approaches give higher AUC value than other approaches using all datasets.

## 5.5 Concluding Remarks

In this chapter two unsupervised learning based architectures are proposed to detect abnormal events in a crowd scene i.e. convolutional autoencoder for spatial domain and LSTM autoencoder for spatiotemporal domain. The achieved AUC values using convolutional autoencoder are 0.86, 0.70, 0.67, 0.82 and 0.6 for UMN, UCSDPed1,Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively. For LSTM autoencoder based approach, feature extraction task is done by CNN architecture. Here experiments are done using two different models: The proposed CNN Architecture2 and pretrained GoogLeNet architecture. The achieved AUC values using CNN Architecture2 are 1, 0.75, 0.48, 0.83 and 0.52 for UMN, UCSDPed1,Violent Flows, Subway Entrance and UCFCrime2Local datasets, respectively while the achieved AUC values using GoogLeNet architecture are 1, 0.94, 0.5, 0.82 and 0.63 respectively.

Because of using temporal features with spatial features, LSTM autoencoder performs better than convolutional autoencoder on various benchmark datasets except for Violent-Flows. As the Violent Flows dataset contains short video clips of the dense crowd, the temporal features cannot improve performance. The comparison of all the proposed approaches with state of the art methods is also shown in this chapter.

# CHAPTER - 6
# Conclusion and Future Scope

This chapter describes the summary of the thesis and points out the probable expansion for future work. The chapter includes two sections. The section 6.1 concludes the thesis by mentioning the significant contributions and the section 6.2 presents future directions for further research.

## 6.1 Conclusion

In this research work, supervised and unsupervised learning based approaches are proposed to detect abnormal events in surveillance scenes. Handcrafted and deep learning both types of features are used for supervised learning based approaches, while for unsupervised learning based approaches, deep learning features are used. To classify events, supervised deep learning architectures are used.

Using handcrafted features, two approaches are proposed. One is based on HOG and optical flow features, and the other is based on autoencoder and optical flow. Combining HOG and optical flow features gives better performance than only HOG or optical flow features using neural network and SVM classifiers. It provides 94.48% accuracy using neural network and 99.35% accuracy using the SVM classifier for the UMN dataset. A combination of features extracted by autoencoder and optical flow performs better than only autoencoder features or optical flow features. Autoencoder and optical flow based approach gives 100% accuracy and AUC value 1 using neural network classifier for the UMN dataset.

For abnormal event detection using supervised deep learning based approaches, three methods are proposed based on Stacked Autoencoder, CNN and CNN-LSTM, respectively. CNN and CNN-LSTM based approaches give outstanding performance on

various benchmark datasets. For the CNN based approach, the proposed architecture performs better than the pre-trained GoogleNet architecture. The proposed CNN architecture provides AUC values 0.99, 1, 1, 1 and 0.73 using UMN, UCSDPed1, Violent-Flows, Subway Entrance and UCFCrime2Local datasets, respectively while the CNN-LSTM architecture provides AUC values 1, 1, 1, 0.9846 and 0.79 respectively. Because of consideration of both spatial and temporal domain, the CNN-LSTM architecture performs better than CNN architecture using complex weakly labelled UCFCrime2Local dataset.

The proposed approaches based on CNN and CNN-LSTM perform efficiently for multiclass events classification. The CNN-LSTM model gives better AUC value than the CNN model for various events using UCFCRime2Local dataset except for assault event. It provides 73.86% overall classification accuracy.

Two architectures are proposed for abnormal event detection using unsupervised deep learning based approaches, i.e. convolutional autoencoder and LSTM autoencoder. The convolutional autoencoder based approach provides AUC values 0.86, 0.70, 0.67, 0.82 and 0.6 for UMN, UCSDPed1,Violent Flows, Subway Entrance and UCFCrime2Local datasets respectively while the LSTM autoencoder provides AUC values 1, 0.94, 0.5, 0.83 and 0.63 respectively. As LSTM autoencoder uses both spatial and temporal domain features, it performs better than convolutional autoencoder on various benchmark datasets except for Violent Flows. As the Violent Flows dataset contains short video clips of the dense crowd, the temporal features cannot improve performance.

Supervised learning based approaches perform better than unsupervised learning based approaches for abnormality detection.

## 6.2 Future Scope

Public and private sectors demand accurate solutions for automatic detection and recognition of anomalies in surveillance scenes, and there is still a large room for improvement. The methods presented in this thesis have been experimented with specific types of abnormal events. Further, more events can be considered with large datasets. The events especially related to the Covid-19 pandemic, like violating social distancing policies

and specific rules of behaviour in public spaces can be recognized. For real-world applications, more events can be handled with a particular cluster of deep learning GPUs to reduce the overall training time for the various proposed models. The feature extraction techniques can be improved to reduce misclassification. For further research, emotional aspects can also be considered to classify abnormal events because changes in people's emotions are usually a precursor of abnormal events.

# List of References

[1] N. Sjarif, S. Shamsuddin, S. Hashim and S. Yuhaniz, "Crowd Analysis and Its Applications" In *Proc. International Conference on Software Engineering and Computer Systems,*2011.

[2] M. Zitouni, H. Bhaskar, J. Dias and M. Al-Mualla, "Advances and Trends in Visual Crowd Analysis: A Systematic Survey and Evaluation of Crowd Modelling Techniques", *Neurocomputing,* Vol. 186, pp.139-159, 2016.

[3] G. Tripathi, K. Singh and D. Vishwakarma, "Convolutional Neural Networks for Crowd Behaviour Analysis: A Survey" The *Visual Computer International Journal of Computer Graphics,* Vol. 35, pp.753–776, 2019.

[4] Unusual Crowd Activity Dataset by University of Minnesota, http://mha.cs.umn.edu./movies/crowdactivity-all.avi/

[5] UCSD Anomaly Detection Dataset by University of California and San Diego, http://www.svcl.ucsd.edu/projects/anomaly/dataset.html

[6] T. Hassner, Y. Itcher, and O. Kliper-Gross, "Violent Flows: Real-Time Detection of Violent Crowd Behavior", In *Proc. 3rd IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.

[7] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. "Robust Real-Time Unusual Event Detection Using Multiple Fixed- Location Monitors", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 30, No.3, pp. 555–560, 2008.

[8] A. Zaharescu and R. Wildes, "Anomalous Behaviour Detection Using Spatiotemporal Oriented Energies, Subset Inclusion Histogram Comparison and Event-Driven Processing", *Computer Vision – ECCV,* 2010.

[9] F. Landi, C. Snoek and R. Cucchiara, "Anomaly Locality in Video Surveillance", arXiv:1901.10364, 2019

[10] University of Central Florida (2012), Computing Optical Flow, [online] Available: https://www.youtube.com/watch?v=5VyLAH8BhF8

[11] S. Wu, H. Wong and Z. Yu, "A Bayesian Model for Crowd Escape Behavior Detection" *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 24, No.1, pp.85– 98, January 2014.

[12] S. Wu, B. Moore and M. Shah, " Chaotic Invariants of Lagrangian Particle Trajectories for Anomaly Detection in Crowded Scenes", In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2054-2060, 2010.

[13] C. Direkoglu, M. Sah, and N. O'Connor, "Abnormal Crowd Behavior Detection Using Novel Optical Flow-Based Features", In *Proc. IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2017.

[14] Y. Cong, J. Yuan and J. Liu, "Abnormal Event Detection in Crowded Scenes using Sparse Representation", *Pattern Recognition*, Vol. 46, No.7, pp.1851-1864, 2013.

[15] D. Chen and P. Huang. "Motion Based Unusual Event Detection in Human Crowds", *Journal of Visual Communication and Image Representation*, Vol.22, No.2, pp.178-186, 2011.

[16] W. Li, V. Mahadevan and N. Vasconcelos , "Anomaly Detection and Localization In Crowded Scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, June - 2013.

[17] M. Halbe, V. Vyas, Y. Vaidya, "Abnormal Crowd Behavior Detection Based on Combined Approach of Energy Model and Threshold", In *Proc. International Conference on Pattern Recognition and Machine Intelligence,* 2017.

[18] X. Hu, Y. Huang, Q. Duan, C. Wenyan, D. Jian and Y. Haima, " Abnormal Event Detection in Crowded Scenes Using Histogram of Oriented Contextual Gradient Descriptor", *EURASIP Journal on Advances in Signal Processing*, 2018.

[19] X. Zhang, Q. Zhang, S. Hu, C. Guo and H. Yu, "Energy Level Based Abnormal Crowd Behavior Detection", *Sensors* MDPI, 2018.

[20] Y. Yong, L. Qiannan and M. Shibiao, "Global Anomaly Crowd Behavior Detection Using Crowd Behavior Feature Vector", *International Journal of Smart Home (SERSC)* Vol. 9, No. 12, pp.149-160, 2015.

[21] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, "Crowded Scene Analysis: A Survey", *IEEE Transactions on Circuits and Systems for Video Technology*, 2015.

[22] M. Ravanbakhsh, M. Nabi, E. Sangineto, L. Marcenaro, C. Regazzoni, N. Sebe, "Abnormal Event Detection in Videos Using Generative Adversarial Nets", In *Proc. IEEE International Conference on Image Processing,* 2017.

[23] H. Mousavi, M. Nabi, H. Kiani, A. Perina, and V. Murino, "Crowd Motion Monitoring Using Tracklet Based Commotion Measure" In *Proc. IEEE International Conference on Image Processing*, 2015.

[24] C. Lu, J. Shi and J. Jia, “Abnormal Event Detection at 150 FPS in MATLAB”, In *Proc. IEEE International Conference on Computer Vision*, pp.2720-2727, 2013.

[25] M. Ravanbakhsh, M. Nabi, H. Mousavi, E. Sangineto and N. Sebe, "Plug-and-Play CNN for Crowd Motion Analysis: An Application in Abnormal Event Detection”, In *Proc. IEEE Winter Conference on Applications of Computer Vision*, pp. 1689-1698, 2018.

[26] T. Mostafa, J. Uddin and Md. Haider Ali, “Abnormal Event Detection in Crowded Scenario”, In *Proc. 3rd International Conference on Electrical Information and Communication Technology*, 2017.

[27] Y. Benabbas, N. Ihaddadene, and C. Djeraba, “Motion Pattern Extraction and Event Detection for Automatic Visual Surveillance”, *Journal on Image and Video Processing*, vol. 7, pp. 1–15, 2011.

[28] Y. Gao, H. Liu, X. Sun, C. Wang, and Y. Liu, "Violence Detection Using Oriented Violent Flows", *Image and Vision Computing*, Elsevier ,Vol.48, pp.37–41, 2016.

[29] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayedd, and R. Klette, “Deep-Anomaly : Fully Convolutional Neural Network for Fast Anomaly Detection in Crowded Scenes”, *Computer Vision and Image Understanding*, 2018.

[30] Y. Feng, Y. Yuan, and X. Lu, “Learning Deep Event Models for Crowd Anomaly Detection”, *Neurocomputing,* Vol. 219, pp.548–556 , 2017.

[31] S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, and Z. Zhang, “Spatial-temporal Convolutional Neural Networks for Anomaly Detection and Localization in Crowded Scenes”, *Signal Processing Image Communication*, Vol. 47, pp 358–36 , 2016.

[32] S. Smeureanu, R. Ionescu, M. Popescu and B. Alexe, “Deep Appearance Features for Abnormal Behavior Detection in Video”, *Image Analysis and Processing—ICIAP* 2017.

[33] J. Sun, J. Shao and C. He, “Abnormal Event Detection for Video Surveillance Using Deep One-Class Learning”, *Multimedia Tools and Application*, 2017.

[34] R. Hinami, T. Mei, and S. Satoh, “Joint Detection and Recounting of Abnormal Events by Learning Deep Generic Knowledge”, *In* Proc. *International Conference on Computer Vision*, 2017.

[35] S. Yan, J. Smith, W. Lu and B. Zhang, "Abnormal Event Detection from Videos Using a Two-stream Recurrent Variational Autoencoder", *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[36] Y. Chong, and Y. Tay “Abnormal Event Detection in Videos Using Spatiotemporal Autoencoder”, In *Proc. 14th International Symposium on Advances in Neural Networks,* 2017.

[37] H. Vu, T. Nguyen, T. Le, W. Luo, and D. Phung, “Robust Anomaly Detection in Videos Using Multilevel Representations”, In *Proc. AAAI Conference on Artificial Intelligence*, Vol. 33, No.1, pp. 5216-5223, 2019.

[38] W. Sultani, C. Chen, M. Shah,” Real-world Anomaly Detection in Surveillance Videos”, In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp.6479-6488, 2018.

[39] R. Ionescu, S. Smeureanu, B. Alexe and M. Popescu, “Unmasking the Abnormal Events in Video”, In *Proc. IEEE International Conference on Computer Vision,* pp. 2914-2922, 2017.

[40] J. Hu, E. Zhu, S. Wang, X. Liu, X. Guo, and J. Yin, “An Efficient and Robust Unsupervised Anomaly Detection Method Using Ensemble Random Projection in Surveillance Videos”, *Sensors*, Vol. 19, No. 19, 2019.

[41] J. Kim and K. Grauman, “Observe Locally, Infer Globally: A Space-time MRF for Detecting Abnormal Activities with Incremental Updates”, In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[42] V. Mahadevan, W. Li, and N. Vasconcelos, "Anomaly Detection In Crowded Scenes," In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[43] V. Saligrama and Z. Chen, “Video Anomaly Detection Based on Local Statistical Aggregates”, In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2112–2119, 2012.

[44] B. Antic and B. Ommer, “Video Parsing for Abnormality Detection”, In *Proc. International Conference on Computer Vision*, pp. 2415–2422, 2011.

[45] E. Ehsan and R. Vidal. “Sparse Subspace Clustering” In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[46] D. Xu, Y. Yan, E. Ricci, and N. Sebe, "Detecting Anomalous Events in Videos by Learning Deep Representations of Appearance and Motion," *Computer Vision and Image Understanding*, Vol 219, issue C, pp- 548-556, 2017.

[47] K. Cheng and W. Fang, "Video Anomaly Detection and Localization Using Hierarchical Feature Representation and Gaussian Process Regression", In *Proc. IEEE Conference on Computer Vision and Pattern Recognition,* 2015.

[48] H. Rabiee, H. Mousavi, M. Nabi, and M. Ravanbakhsh, "Detection and localization of crowd behavior using a novel tracklet-based model", *International Journal of Machine Learning and Cybernetics*, Vol. 12, pp.1999–2010, 2018.

[49] P. Bilinski and F. Bremond " Human Violence Recognition and Detection in Surveillance Videos", In *Proc. 13th IEEE International Conference on Advanced Video and Signal Based Surveillance,* pp. 30–36, 2016.

[50] F. Ullah, A. Ullah, K. Muhammad, I. Haq and S. Baik, "Violence Detection Using Spatiotemporal Features with 3D Convolutional Neural Network", *Sensors,* MDPI, Vol. 19, No. 10, 2019.

[51]https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[52] S. Sudhakaran and O. Lanz, "Learning to Detect Violent Videos using Convolutional Long Short-Term Memory" *14th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp.1-6,2017

[53] J. Feng, Y. Liang and L. Li, "Anomaly Detection in Videos Using Two-Stream Autoencoder with Post-hoc Interpretability", *Computational Intelligence and Neuroscience*, Vol. 2021, 2021.

[54] Tian Wang, Zichen Miao, Yuxin Chen, Yi Zhou, Guangcun Shan, Hichem Snoussi, "AED-Net: An Abnormal Event Detection Network", *Engineering*, Vol. 5, No.5, pp 930-939, 2019.

[55] W. Peng, X. Hongling, L. Wenlin and S. Wenlong, "Harris Scale Invariant Corner Detection Algorithm Based on the Significant Region", *International Journal of Signal Processing, Image Processing and Pattern Recognition,* Vol.9, No.3, pp.413-420, 2016.

[56] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pp. 886- 893, Vol. 1, 2005.

[57] B. Horn and B. Schunck, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, pp. 185-203, 1981.

[58] S. Mishra, U. Sarkar, S. Taraphder, S. Datta, D. Swain, R. Saikhom et al. "Multivariate Statistical Data Analysis-Principal Component Analysis (PCA)", *International Journal of Livestock Research*, Vol. 7, No 5, pp. 60-78.

[59] M. Satya," Histogram of Oriented Gradients Explained Using Opencv", December 6, 2016https://www.learnopencv.com/histogram-of-oriented-gradients/

[60] https://appliedgo.net/perceptron/

[61] http://uc-r.github.io/feedforward_DNN

[62] https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

[63] V. Salunkhe, "Support Vector Machine", 23 July, 2021 https://medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38

[64] M. Ribeiro, A. Lazzaretti and H. Lopes, "A Study of Deep Convolutional Autoencoders for Anomaly Detection in Videos", *Pattern Recognition Letters*, 2017.

[65] D. Rumelhart, G. Hinton and R. Williams, "Learning Representations by Back-propagating Errors", *Nature*, Vol. 323, pp. 533–536, 1986.

[66] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber , " Stacked Convolutional Auto-encoders for Hierarchical Feature Extraction", In *Proc. 21st International Conference on Artificial Neural Networks*, pp. 52-59, 2011.

[67] V. Jonnalagadda, "Sparse, Stacked and Variational Autoencoder", December 6, 2018 https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64

[68] Ng. Andrew, "Sparse autoencoder" CS294A Lecture notes

[69] M. Vorontsova, "Deep Learning vs Machine Learning: Overview & Comparison", September 12, 2019 https://soshace.com/deep-learning-vs-machine-learning-overview-comparison/

[70] Y. LeCun et al., "Handwritten Digit Recognition with a Back-propagation Network", *Advances in Neural Information Processing Systems*, vol. 2, pp. 396–404, 1989.

[71] S. Christian et al, " Going Deeper with Convolutions ", arXiv, 2014.

[72] X. Glorot and Y. Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In *Proc. 13th International Conference on Artificial Intelligence and Statistics*, pp. 249-256, 2010.

[73] G. Christian, Z. Xingquan and M. Oge, " Dropout vs. Batch Normalization: An Empirical Study of Their Impact to Deep Learning" *Multimedia Tools and Applications*, January, 2020.

[74] X. Li, S. Chen, X. Hu and J. Yang, “Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift”, In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2019.

[75] S. Hochreiter and J. Schmidhuber, “Long Short Term Memory ”, *Neural Computation*, Vol. 9, No.8, pp.1735-1780,1997.

[76] K. Eugine, “Long Short-Term Memory (LSTM): Concept”, September 2, 2017 https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359

[77] A. Bruhn, J. Weickert and C. Schnörr, “Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods”, *International Journal of Computer Vision*, Vol. 61**,** pp. 211–231, 2005.

[78] A. Krizhevsky and G. E. Hinton, “Using Very Deep Autoencoders for Content Based Image Retrieval”, *In Proc. 19th European Symposium on Artificial Neural Networks*, 2011.

[79] C. Cortes and V. Vapnik, “Support-Vector Networks”, *Machine Learning,* Vol. 20, pp. 273–297, 1995.

[80] J. C. S. Jacques Junior, S. R. Musse and C. R. Jung, "Crowd Analysis Using Computer Vision Techniques," *IEEE Signal Processing Magazine*, Vol. 27, No. 5, pp. 66-77, Sept. 2010.

[81] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”, Towards data Science, 2018 https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[82] J. Zakaria, “A Step by Step Explanation of Principal Component Analysis (PCA)” April 1, 2021 https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[83] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision” *Imaging Understanding Workshop*, pp. 121—130,1981.

[84]https://www.mathworks.com/help/deeplearning/ref/trainautoencoder.html;jsessionid=03073da04e64f4c7a2343229745f#buyr01q-1

[85] https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/

# List of Publications

- K. Joshi, N. Patel and M. Shah, , "A Review On Crowd Analysis Techniques", *Journal of Artificial Intelligence Research & Advances* Vol.6, No.2, pp. 119-126, 2019.
- K. Joshi and N. Patel, "A CNN Based Approach for crowd Anomaly Detection", *International Journal of Next-Generation Computing,* Vol.12, No. 1, pp. 1-11, 2021.
- K. Joshi, N. Patel, “Abnormal Event Detection in a Surveillance Scene Using Convolutional Neural Network”, *International Journal of Computer Vision and Image Processing,* Vol. 11, No.4, pp. 1-20, 2021.