ABNORMAL EVENT DETECTION

Submitted to the

Department of Master of Computer Applications

in partial fulfilment of the requirements

for the Mini Project (MCAP1)

by

Ashok Mahala
1MS21MC010

Sharat Krishna Gouda
1MS21MC047

Under the guidance of

Assistant Professor
Nithya BN

Department of Master of Compute Applications

RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)
Accredited by National Board of Accreditation & NAAC with 'A+' Grade
MSR Nagar, MSRIT Post, Bangalore-560054
www.msrit.edu

2023

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project entitled **ABNORMAL EVENT DETECTION** is carried out by **Ashok Mahala** bearing USN **1MS21MC010** and **Sharat Krishna Gouda** bearing USN **1MS21MC047** students of 3rd semester, in partial fulfillment for the Mini Project (MCAP1), during the academic year 2022-2023.

Guide **Head of the Department**

Nithya BN

Name of Examiners **Signature with Date**

1.

2.

DECLARATION

I hereby declare that the project report entitled "**ABNORMAL EVENT DETECTION**" based on study undertaken by me, towards the partial fulfilment for the Mini Project (MCAP1) carried out during the 3rd semester, has been compiled purely from the academic point of view and is, therefore, presented in a true and sincere academic spirit. Contents of this report are based on my original study and findings in relation there to are neither copied nor manipulated from other reports or similar documents, either in part or in full, and it has not been submitted earlier to any University/College/Academic institution for the award of any Degree/Diploma/Fellowship or similar titles or prizes and that the work has not been published in any specific or popular magazines.

Place: Bangalore

Date:

Name: Ashok Mahala

USN: 1MS21MC010

Name: Sharat Krishna Gouda

USN: 1MS21MC047

ACKNOWLEDGEMENTS

We would like to express my gratitude to everyone who has been a part of this unforgettable adventure. When we look back from where our journey began, we recollect all the people who brought this project to completion.

First and foremost, we would like to thank our Principal**, Dr. N. V. R. Naidu** for providing us with the environment and facilities to carry out the project successfully.

We express our great appreciation to our HOD and project guide, **Dr. S Seema** for her guidance, support and valuable inputs along with necessary resources which helped in completion of the project.

We express our gratitude to our project coordinator **Nithya BN** for guiding our throughout the course of the project.

We also express our sincere thanks to all the teaching and non-teaching staff of the **Department of Master of Computer Applications** for their kind support and ever helping nature.

We would also like to thank our parents and other family members for their continuous support, without which, we would not have been able to achieve whatever we have. We would like to thank all our friends who have directly or indirectly helped me throughout the duration of the project.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Ashok Mahala
1MS21MC010

Sharat Krishna Gouda
1MS21MC047

ABSTRACT

Abnormal event detection is a growing demand to process a plethora of surveillance videos. Our project helps detect the abnormality in videos with high accuracy, thus saving time for organizations and individuals who would have to go through the entire footage instead.

The proposed method utilizes various feature extraction and selection techniques to identify and extract meaningful features from the input data. A machine learning algorithm is then trained on the extracted features to classify normal and abnormal events. The performance of the proposed method is evaluated using real-world datasets, and the results demonstrate its effectiveness in detecting abnormal events with high accuracy. The proposed method has applications in various domains such as surveillance, intrusion detection, and anomaly detection in industrial systems. We implement and evaluate several of these methods on standard benchmark datasets, including the UCSD Pedestrian dataset and the ShanghaiTech Campus dataset. Our results show that deep learning methods, particularly those based on convolutional neural networks (CNNs), are effective at detecting abnormal events in video streams. However, we also find that unsupervised methods such as autoencoders and generative adversarial networks (GANs) can perform well, particularly when training data is limited. Overall, our project provides a comprehensive survey of the state-of-the-art in abnormal event detection and presents practical recommendations for deploying these techniques in real-world scenarios.

Table of Contents

1. Introduction

1.1 Overview

The expansion of video data has led to a growing demand for object recognition and abnormal event detection. In particular, there is a need to identify uncommon or suspicious occurrences in a vast amount of ordinary data. This is crucial for various applications, from quality control to surveillance. Despite the importance of detecting anomalies in video sequences, like surveillance footage, manual detection is a challenging and time-consuming task, given the low frequency of meaningful events. Hence, there is a need for automated detection and segmentation of these sequences, but current technology is complex and requires substantial configuration efforts.

Surveillance cameras are very common across various industries throughout the world. The applications of these cameras can range from theft deterrence to weather monitoring and more. Parks, communities, and neighbourhoods — all public spaces — should be outfitted with video surveillance systems to help deter crime and enhance public safety. Law enforcement can also view video directly from their smartphones, enabling quicker response times. Video surveillance can help enormously with crowd control as well as prevent crime by providing security staff with real-time images from an event. Zoom in on suspicious behaviour before it becomes a problem with modern IP HD surveillance systems. Computerized abnormal event detection can be applied in these scenarios as such events are time sensitive and detecting them with very little delay and high accuracy is critical. Also computerized event detection reduces human prone error and with its high frame rates, has become indispensable.

Traditional image processing algorithms to detect abnormalities are heavy on computations and are thus slow and require very powerful systems to be run on. Sparsity based techniques convert these heavy computations into smaller costless least square optimizations which speeds up the process and provides faster detection rates.

1.2 Problem Definition

Abnormal event detection in videos involves identifying unusual or unexpected events in a video stream, such as a person breaking into a building, a vehicle accident, or a person behaving strangely in a public space. This task is challenging because abnormal events can take many different forms and are often rare, making them difficult to detect using traditional methods such as motion detection or background subtraction. Additionally, the background of the video can be dynamic and complex, making it difficult to establish a clear threshold for what constitutes an abnormal event. The goal of abnormal event detection in videos is to automatically detect these events and alert security personnel or other relevant parties in real-time.

2. Literature Survey

TITLE	YEAR	METHODOLOGY	OUTCOME
Abnormal Event Detection in Videos using Spatiotemporal Autoencoder, by Yong Shean Chong Yong Haur Lee Kong Chian Faculty of Engineering Science, Universiti Tunku Abdul Rahman, 43000 Kajang, Malaysia.	January 9, 2017	In this research, they have applied deep learning to the abnormal event detection problem. They formulate abnormal event detection as a spatiotemporal sequence outlier detection problem and applied a combination of spatial feature extractor and temporal sequencer ConvLSTM to tackle the problem. The ConvLSTM layer not only preserves the advantages of FC-LSTM but is also suitable for spatiotemporal data due to its inherent convolutional structure. By incorporating convolutional feature extractor in both spatial and temporal space into the structure, they build an end-to-end trainable model for abnormal event detection.	They Detected the Abnormal events in videos but the accuracy was not good. The processing is also very slow and requires high resources like memory and cpu.
A Background-Agnostic Framework With Adversarial Training for Abnormal Event Detection in Video, Mariana Iuliana Georgescu , Radu Tudor Ionescu , Member, IEEE, Fahad Shahbaz Khan , Member, IEEE, Marius Popescu	2019	In this paper, they have presented a set of significant design changes to our abnormal event detection approach. More specifically, they replaced the k-means clustering and the one-versus-rest SVM with a set of binary classifiers that learn from normal and pseudo-abnormal examples. Additionally, modified the convolutional autoencoders by adding adversarial and segmentation branches, as well as skip connections. Our design changes resulted in significant performance improvements in terms of both RBDC and TBDC.	They Detected the Abnormal events in videos and the accuracy was good. But this method cannot be applied on live streaming videos.

Global Abnormal Events Detection in Surveillance video – A Hierarchical approach Kumar Biswas Department of Electronics and Communication Engineering Indian Institute of Technology, Kharagpur West Bengal, India	2009	In this paper, they proposed a global Abnormal Event detection method using hierarchical approach. The histogram of the orientation of optical flow for each spatio-temporal block of different size is computed and used as feature descriptor. The one-class SVM is applied to model the normal events and detect abnormal cubes. Finally, the spatio-temporal post-processing is performed to reduce false alarm rate. The hierarchical approach helps to increase the speed and accuracy. In future, we wish to apply the proposed method on different global abnormal event dataset such as PETS 2009 dataset and evaluate the performance.	Accuracy is very good and can be used for live streaming also. Requires high resources.
Abnormal Action Detection In Video Surveillance Omnia Ayman Elsayed , Noura Ahmed Mohamed Marzouk , Esraa Atef and Mohammed A.-M. Salem Faculty of Media Engineering and Technology, German	2020	The project presented in this paper aims to create a system that can detect humans and recognize their actions labelling those actions as normal or abnormal. Using the MHI, cascade Object Detection training (based on extracted HOG features), and action recognition training methods. The Cascade classifier used for object detection achieved 71% accuracy. Next, Motion History Image was used, HOG and multiclass SVM classifier to implement the human action recognition and classification algorithm. The multiclass SVM classifier has been trained and tested using Weizmann and KTH action datasets. The results were satisfactory as the datasets do not include a huge number of samples to train the classifier.	In mentioned training process, Motion History Image method, cascade object detection and recognition training were utilized. Cascade classifier applied for object detection achieved accuracy of 71%. Accuracy is not so good.

3. Hardware and Software Requirements

3.1 Hardware Requirements

A desktop with 3.4GHz CPU, 8G memory, 32-bit or 64-bit processor, minimum 8-bit graphic adapter and running windows operation system. Basic hardware like monitor, mouse and keyboard are a must for input and output. A CD-ROM or a USB port to install the necessary software. An additional camera will be required for real-time video processing.

3.2 Software Requirements

Python with any version with the following modules installed, OpenCV, NumPy, OS, time, fmatch, pickle, and tensarflow. Windows operation system. Camera modules installed for importing video file from camera.

4. Software Requirements Specification

4.1 System Features

4.1.1 Video Input

We will attach the CCTV with the model which will provide the videos for processing.

4.1.2 Frame Extraction

It will extract the frames from the video and add the frame matrix values to a data file which will be used for further processing. The system should process images in the chosen image formats.

4.1.3 Output

The system shall detect abnormal frames and display the exact time from the start at which the abnormal event is detected. The length of the test video can vary and no particular limit is imposed. In the presence of no abnormality, user should be displayed with a message that the video is normal.

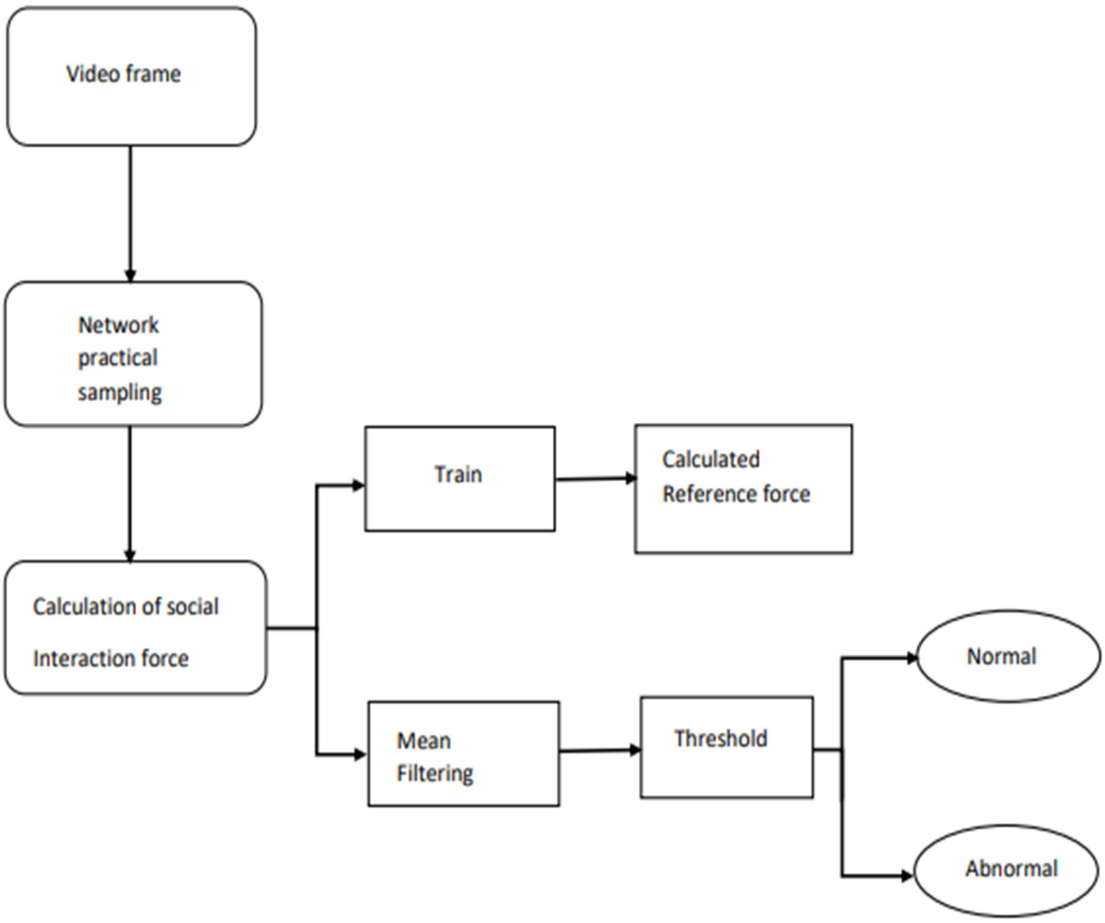
5. System Design Description (SDD)

5.1 System Overview

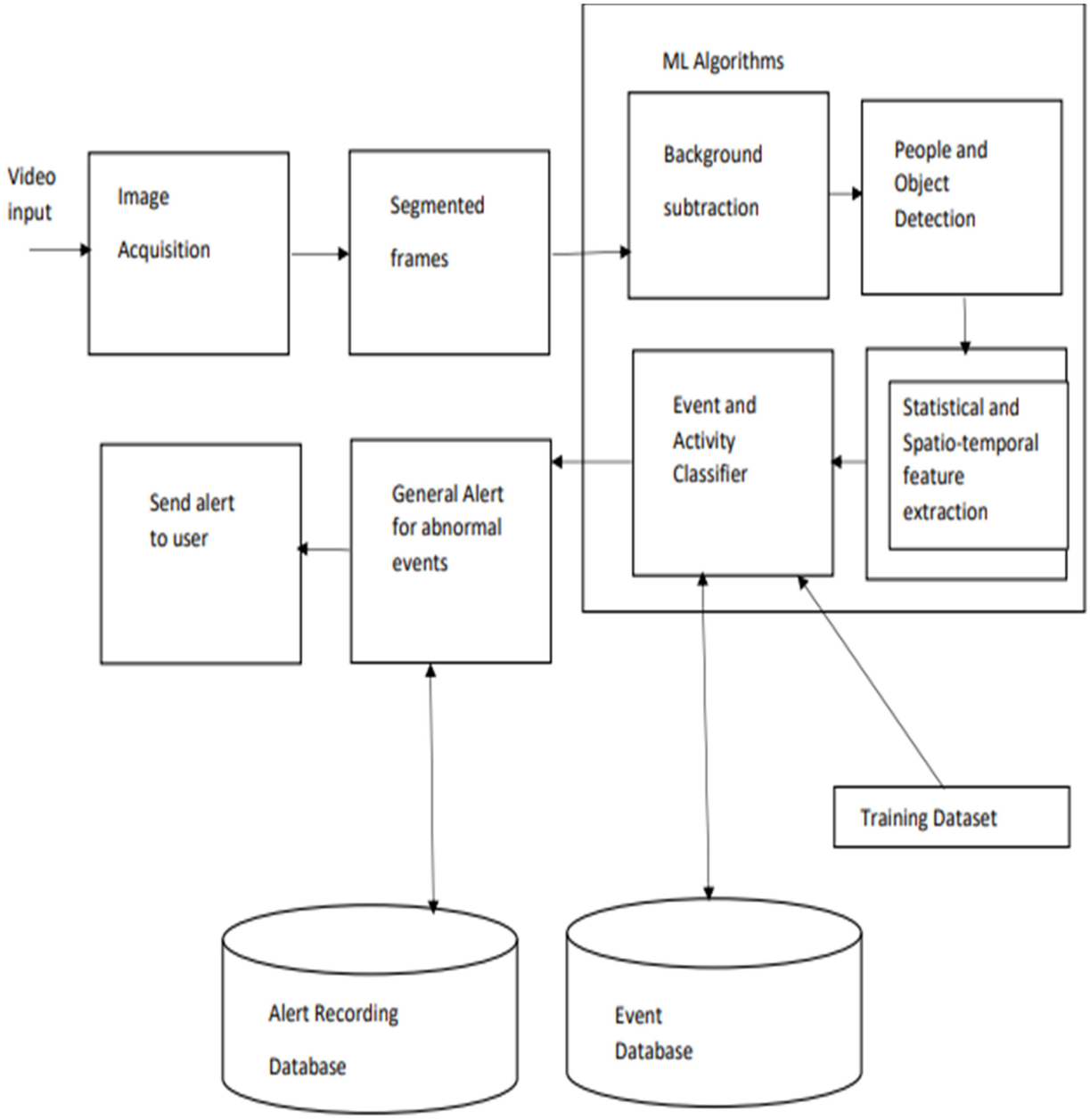
The pre-processing step includes importing the video frames and making it reading for training. It also involves feature extraction which is the input to the training algorithm. Initially the video is converted to frames. Each video will generate a set of frames which approximately denotes the number of seconds.

The training algorithm uses a method called Sparse Combination Learning which will be used to learn a set of combination using the given input features.

5.1.1 Design Diagram



5.1.2 System Architecture



5.2 Database Design/Data Set Description

5.2.1 Data Set Description

We train our model on five most commonly used benchmarking datasets: Avenue, UCSD Ped1 and Ped2, Subway entrance and exit datasets. All videos are taken from a fixed position for each dataset. All training videos contain only normal events. Testing videos have both normal and abnormal events. In Avenue dataset, there are total 16 training and 21 testing video. Each clips duration varies between less than a minute to two minutes long. The normal scenes consist of people walking between staircase and subway entrance, whereas the abnormal events are people running, walking in opposite direction, loitering and etc. The challenges of this dataset include camera shakes and a few outliers in the training data. Also, some normal pattern seldom appears in the training data.

5.3 Functional Design

5.3.1 Describe the functionalities of the system:

5.3.1.1 Video Input

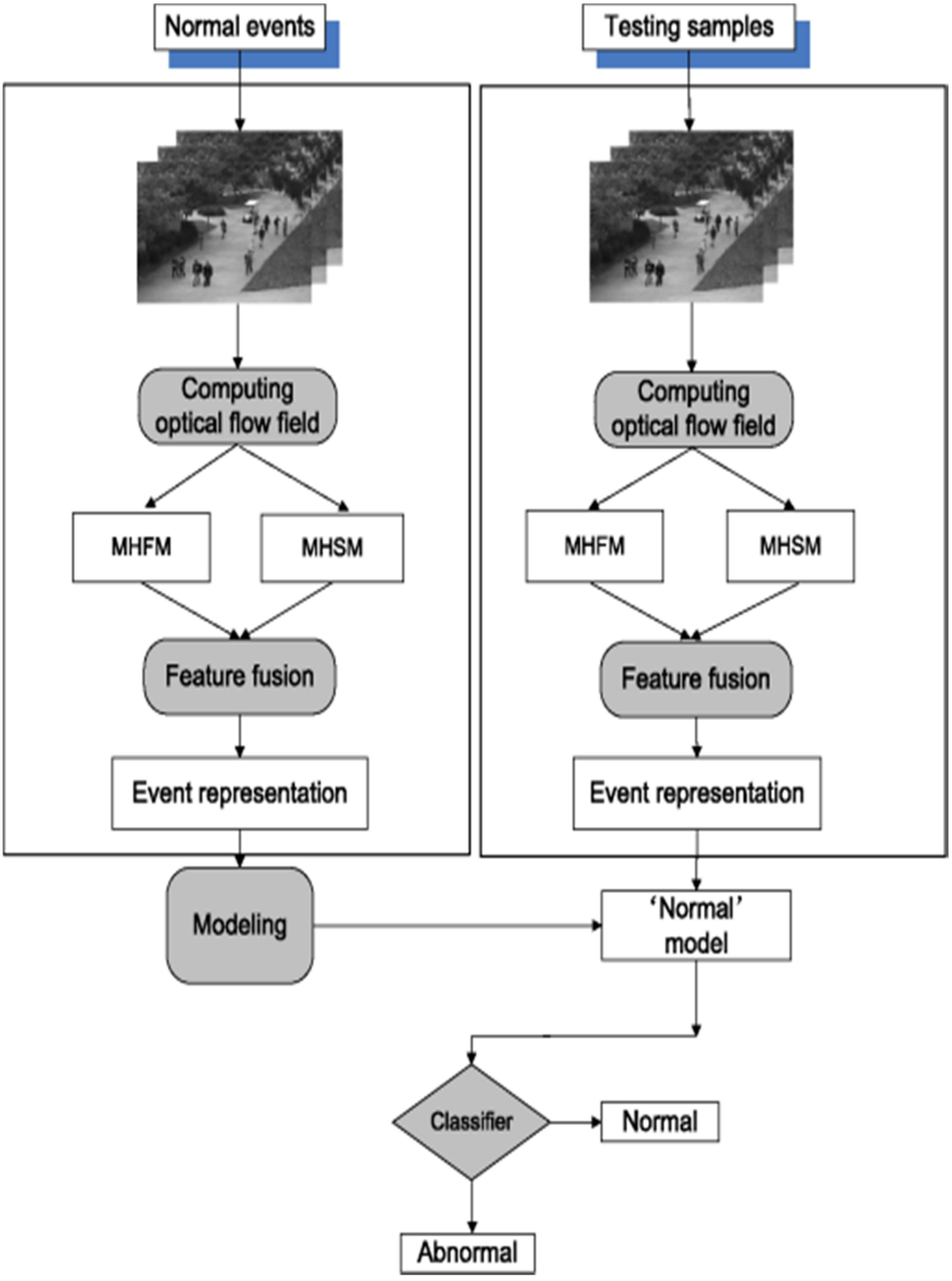
We will attach the CCTV with the model which will provide the videos for processing.

5.3.1.2 Frame Extraction

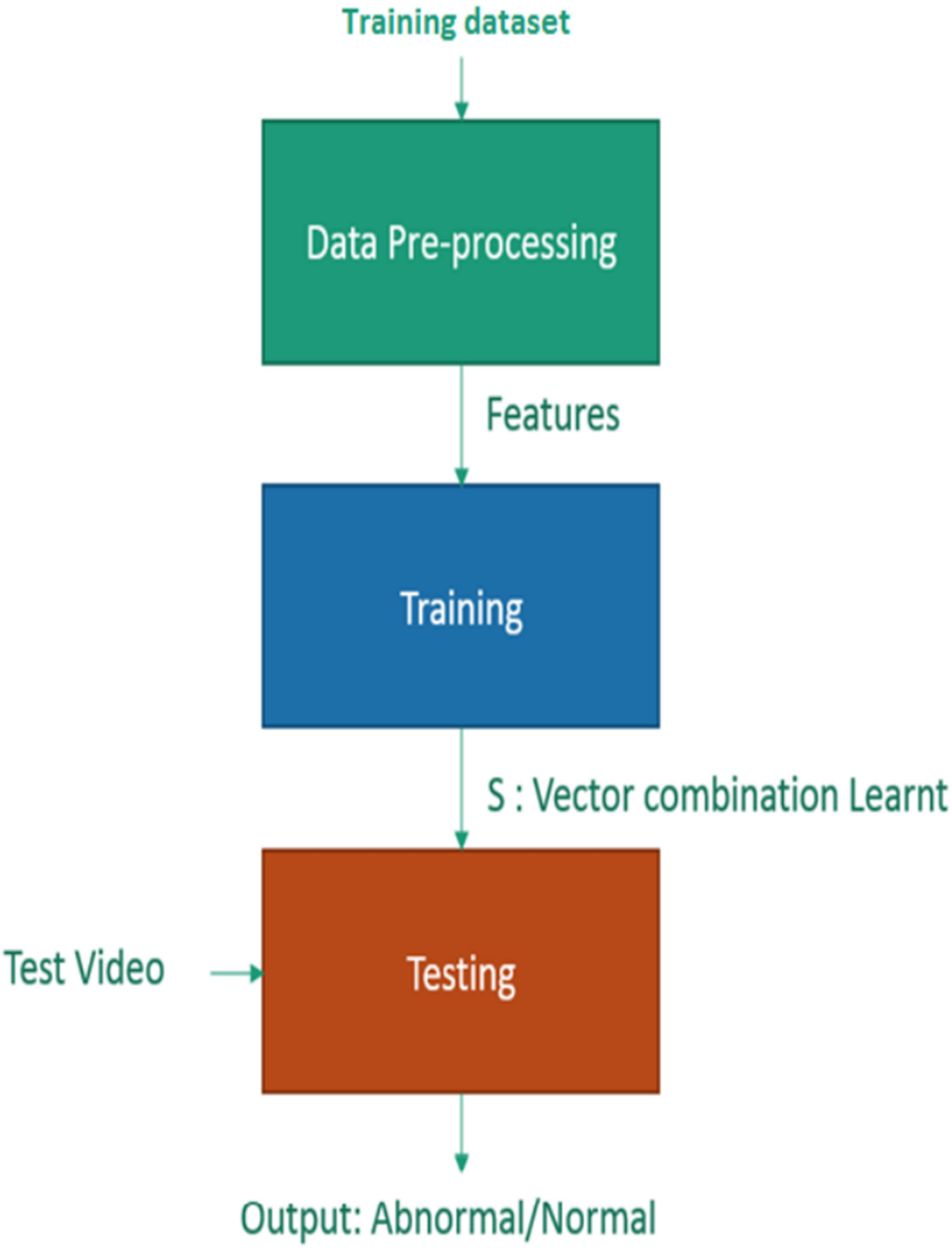
It will extract the frames from the video and add the frame matrix values to a data file which will be used for further processing. The system should process images in the chosen image formats.

5.3.1.3 Output

The system shall detect abnormal frames and display the exact time from the start at which the abnormal event is detected. The length of the test video can vary and no particular limit is imposed. In the presence of no abnormality, user should be displayed with a message that the video is normal.


Normal events
Testing samples
Computing optical flow field
Computing optical flow field
MHFM
MHSM
MHFM
MHSM
Feature fusion
Feature fusion
Event representation
Event representation
Modeling
'Normal' model
Classifier
Normal
Abnormal

5.4 Behavioral design:



6. Implementation

6.1 Preprocessing

Here frames are extracted from the videos and stored into local disk for further processing.

Code- processor.py

```
import tensorflow

import keras

from keras.utils import img_to_array,load_img

import numpy as np

import glob

import os

# from scipy.misc import imresize

from skimage.transform import resize

import argparse


imagestore=[]

parser=argparse.ArgumentParser(description='path of the training file')

parser.add_argument('source_vid_path',type=str)

parser.add_argument('fps',type=int)

args=parser.parse_args()


video_source_path= args.source_vid_path

fps=args.fps

def create_dir(path):

        if not os.path.exists(path):
```

```
            os.makedirs(path)
def remove_old_images(path):
    filelist = glob.glob(os.path.join(path, "*.png"))
    for f in filelist:
            os.remove(f)


def store(image_path):
    img=load_img(image_path)
    img=img_to_array(img)
    #Resize the Image to (227,227,3) for the network to be able to process it.
    img=resize(img,(227,227,3))
    #Convert the Image to Grayscale
    gray=0.2989*img[:,:,0]+0.5870*img[:,:,1]+0.1140*img[:,:,2]
    imagestore.append(gray)
#List of all Videos in the Source Directory.
videos=os.listdir(video_source_path)
print("Found ",len(videos)," training video")
#Make a temp dir to store all the frames
create_dir(video_source_path+'/frames')
#Remove old images
remove_old_images(video_source_path+'/frames')
framepath=video_source_path+'/frames'
for video in videos:
            os.system( 'ffmpeg -i {}/{} -r 1/{}
{}/frames/%03d.jpg'.format(video_source_path,video,fps,video_source_path))
```

```
        images=os.listdir(framepath)
        for image in images:
            image_path=framepath+ '/'+ image
            store(image_path)
imagestore=np.array(imagestore)
a,b,c=imagestore.shape
#Reshape to (227,227,batch_size)
imagestore.resize(b,c,a)
#Normalize
imagestore=(imagestore-imagestore.mean())/(imagestore.std())
#Clip negative Values
imagestore=np.clip(imagestore,0,1)
np.save('training.npy',imagestore)
#Remove Buffer Directory
#os.system('rm -r {}'.format(framepath))
```

6.2 Training

This is to Train the model and store it into model.h5

Code-

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
from model import load_model
import numpy as np
import argparse


parser=argparse.ArgumentParser()
parser.add_argument('n_epochs',type=int)
```

```
args=parser.parse_args()
X_train=np.load('training.npy')
frames=X_train.shape[2]
#Need to make number of frames divisible by 10
frames=frames-frames%10
X_train=X_train[:,:,:frames]
X_train=X_train.reshape(-1,227,227,10)
X_train=np.expand_dims(X_train,axis=4)
Y_train=X_train.copy()

epochs=args.n_epochs
batch_size=1
if __name__=="__main__":
  model=load_model()
  callback_save = ModelCheckpoint("model.h5",
                  monitor="mean_squared_error", save_best_only=True)
  callback_early_stopping = EarlyStopping(monitor='val_loss', patience=3)
  print('Model has been loaded')
  model.fit(X_train,Y_train,
      batch_size=batch_size,
      epochs=epochs,
      callbacks = [callback_save,callback_early_stopping]
      )
```

6.3 Testing on single video File

Through this we can test a single video file which is available into our local disk . Put it into Testing Folder and run test.py.

Test.py

Code-

```
from keras.models import load_model
import numpy as np
def mean_squared_loss(x1,x2):
  ''' Compute Euclidean Distance Loss  between
  input frame and the reconstructed frame'''

  diff=x1-x2
  #print("diff",diff)
  a,b,c,d,e=diff.shape

  n_samples=a*b*c*d*e
  #print("n_samples",n_samples)
  sq_diff=diff**2
  #print("sq_diff",sq_diff)
  Sum=sq_diff.sum()
  #print("Sum",sq_diff)
  dist=np.sqrt(Sum)
  #print("dist",dist)
  mean_dist=((dist/n_samples)*1000)
  print("mean_dist",mean_dist)
  #print("mean_dist",mean_dist)
  return mean_dist

'''Define threshold for Sensitivity
Lower the Threshhold,higher the chances that a bunch of frames will be flagged as Anomalous.'''

threshold=0.6
```

```
model=load_model('model.h5')

X_test=np.load('testing.npy')
frames=X_test.shape[2]
#Need to make number of frames divisible by 10
flag=0 #Overall video flagq
frames=frames-frames%10
X_test=X_test[:,:,:frames]
X_test=X_test.reshape(-1,227,227,10)
X_test=np.expand_dims(X_test,axis=4)

for number,bunch in enumerate(X_test):
  n_bunch=np.expand_dims(bunch,axis=0)
  #print(n_bunch)
  reconstructed_bunch=model.predict(n_bunch)
  #print(reconstructed_bunch)

  loss=mean_squared_loss(n_bunch,reconstructed_bunch)
  print(loss)

  if loss>threshold:
    print("Anomalous bunch of frames at bunch number {}".format(number))
    flag=1
  else:
    print('Bunch Normal')
if flag==1:
  print("Abnormal Events detected")
```

6.4 Processing of Live Video

Here we will process the live video using the CCTV

Start_live_feed.py

Code-

```
import cv2
from model import load_model
import numpy as np
#from scipy.misc import imresize
from skimage.transform import resize
from test import mean_squared_loss
from keras.models import load_model
import argparse
import winsound
duration = 1000  # milliseconds
freq = 440  # Hz
parser=argparse.ArgumentParser()
parser.add_argument('modelpath',type=str)
args=parser.parse_args()
modelpath=args.modelpath
vc=cv2.VideoCapture(0,cv2.CAP_DSHOW)
rval=True
print('Loading model')
model=load_model(modelpath)
print('Model loaded')

threshold=0.67
while True:
  imagedump=[]
  for i in range(10):
    rval,frame=vc.read()
    frame=resize(frame,(227,227,3))
    cv2.imshow('frame', frame)
```

```
    #Convert the Image to Grayscale

    gray=0.2989*frame[:,:,0]+0.5870*frame[:,:,1]+0.1140*frame[:,:,2]
    gray=(gray-gray.mean())/gray.std()
    gray=np.clip(gray,0,1)
    imagedump.append(gray)

  imagedump=np.array(imagedump)
  imagedump.resize(227,227,10)
  imagedump=np.expand_dims(imagedump,axis=0)
  imagedump=np.expand_dims(imagedump,axis=4)
  print('Processing data')
  output=model.predict(imagedump)
  loss=mean_squared_loss(imagedump,output)
  print("loss is:",loss)
  if loss>threshold:
    print('Abnormal Event Detected')
    winsound.Beep(freq, duration)
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break
vc.release()
cv2.destroyAllWindows()
```

7. Testing

7.1 Description of Testing

Unit testing: All the files are tested separately. And ensure that all the functions of each file are working properly.

Integration Testing: Integrate all the files and test it again check that all the functions the overall model is working properly. Check it on different systems and be sure that it will work on any independent system which fulfils the requirements.

7.2 Test Cases

Test case #	Test case Name	Test case Description	Inputs	Expected Output	Actual Output	Status
1.	Accuracy test	Here we train the model with videos which have only normal events.	12 Video Files	Event should be normal	Event normal	Successful in detection
2.	Accuracy test	Here we train the model with videos which have only normal events.	10 Video files	Event should be Abnormal	Event Abnormal	Successful in detection

8. Results and Discussion

8.1 Testing the video file which have abnormal event

Provide the path of the Video file which we want to test.

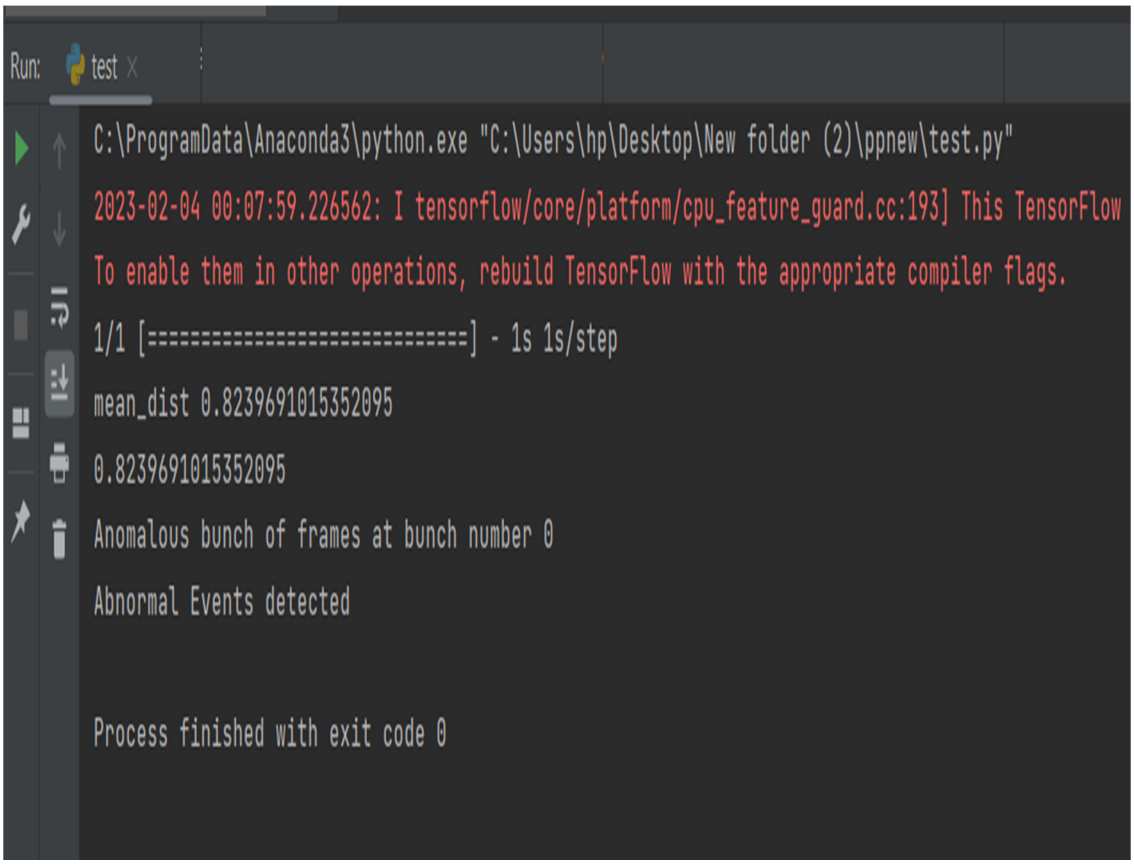
8.1.1 Everything is normal here

So the output is also Bunch Normal

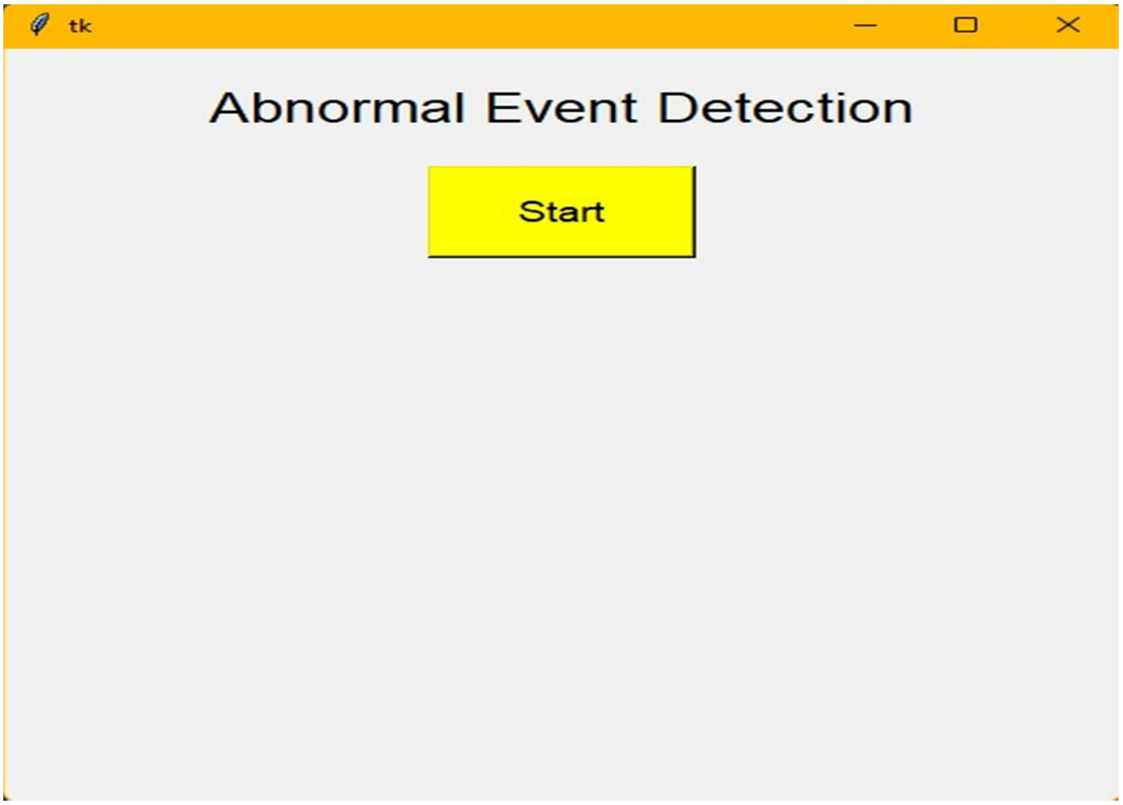
```
C:\ProgramData\Anaconda3\python.exe C:\Users\hp\Desktop\mini_project\project\test.py
2023-03-03 00:35:57.949401: I tensorflow/core/platform/cpu_feature_guard.cc:193] This
To enable them in other operations, rebuild TensorFlow with the appropriate compiler
1/1 [==============================] - 1s 1s/step
mean_dist 0.4290865117651438
0.4290865117651438
Bunch Normal
1/1 [==============================] - 0s 357ms/step
mean_dist 0.42624147561703724
0.42624147561703724
Bunch Normal

Process finished with exit code 0
```

8.1.2 In another Video the boy throws the bag so it's an abnormal event so the output is abnormal

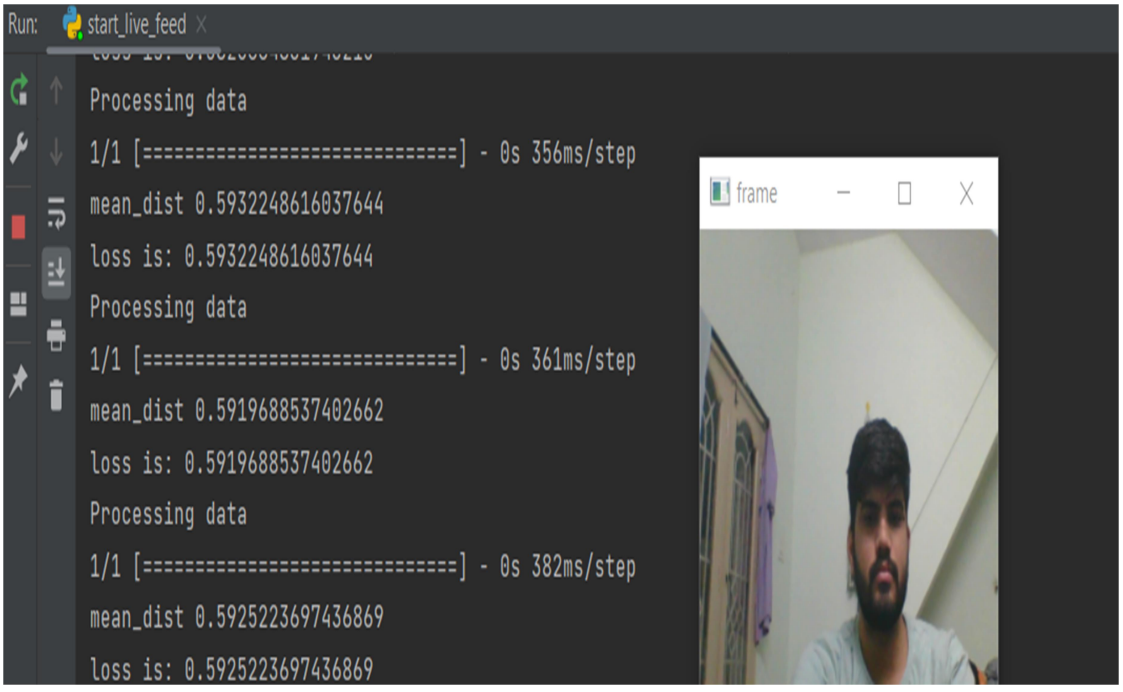
Output **Abnormal Event Detected**



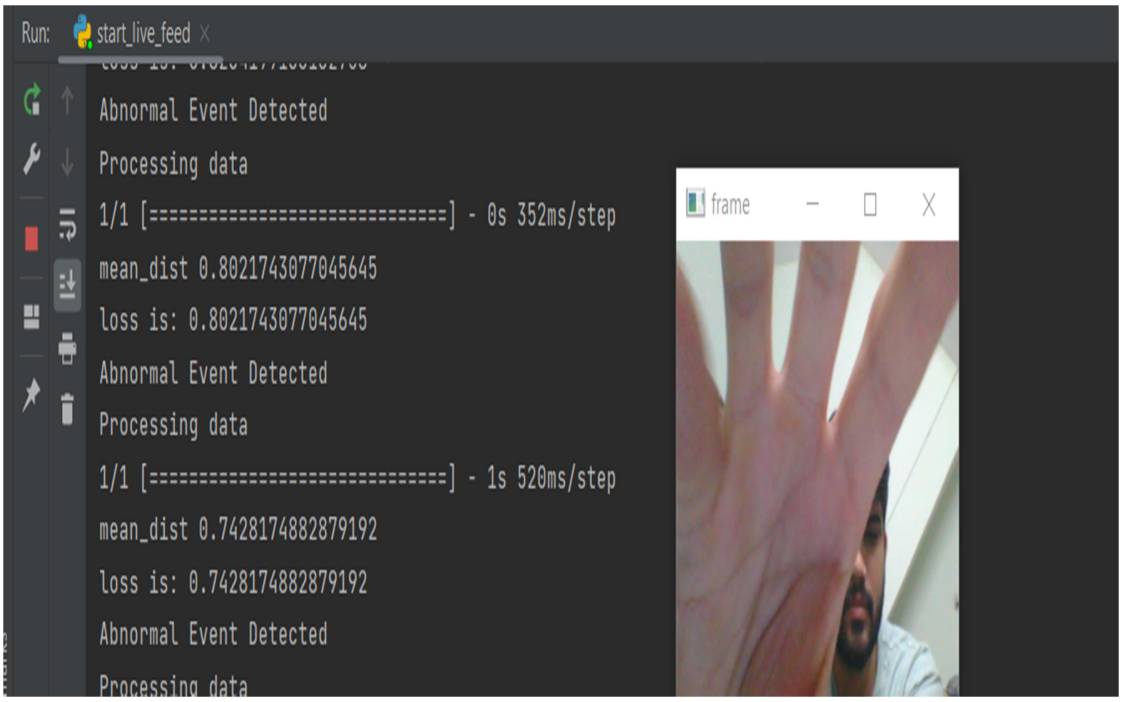
8.2 Testing live streaming



8.2.1 Everything is normal



8.2.2 Abnormal event: Trying to cover the camera



Abnormal Event is detected and beep alarm is triggered.

9. Conclusion

The conclusion for Abnormal Event Detection is that it is a crucial process in various applications such as surveillance, cyber security, and anomaly detection in data streams. It involves identifying events or patterns in data that deviate from the normal or expected behaviour, which can indicate potential issues or risks. Effective abnormal event detection requires robust algorithms, accurate data labelling, and the ability to handle high volumes of data in real-time. Despite the challenges, the development of this field is on-going, and there is a significant potential for future advancements in abnormal event detection techniques.

The advantage of this semi-supervised model is that it only requires a long video segment of normal events to be trained. However, the model's performance may still result in more false alarms than other methods, particularly in scenes with complex activity. Future work will focus on improving the results using active learning, where human feedback is used to update the model and reduce false alarms. One possibility is to add a supervised module to the current system, which only trains a discriminative model on the video segments filtered by the proposed method.

Since it achieves a frame rate of 100fpm, frames can be analysed at a decent rate and thus can be used in surveillance cameras to detect abnormalities automatically. Based on the signal of this system, alarms and other actions can be controlled.

10. Scope for Further Enhancement

Our future work will be to extend the sparse combination learning framework to other video applications. This can be extended to detect various other kind of abnormal events that are usually encountered. By doing this a system which can detect any abnormality will be build which can deployed in various environments just encouraging portability. The algorithm should be carefully analysed for areas where parallel processing is possible and suitably the algorithm should be tweaked which can help in achieving better accuracy.

10.1 Improved data labeling:

The utilization of computerized data labelling strategies aims to enhance the precision of data labelling while minimizing the chances of incorrect positive or negative identifications. This is achieved through the use of algorithms and machine learning techniques to label data automatically, reducing the need for manual labour and increasing efficiency. Techniques like active learning, transfer learning, and semi-supervised learning may be employed to enable the model to continually learn and adjust based on prior labelling results, but it is essential to train the model with diverse and representative data to prevent bias and increase accuracy.

10.2 Integrating with other technologies:

The integration of abnormal event detection systems with other technologies such as computer vision and natural language processing to improve performance.

10.3 Real-time monitoring:

Improving the speed and accuracy of real-time monitoring systems to detect abnormal events as soon as they occur.

11. Bibliography

Cha, S. & Kim, H. (2017). Anomaly detection in real-time streaming data with deep learning. Journal of Ambient Intelligence and Humanized Computing, 8(5), 497-514. https://link.springer.com/article/10.10.07/s126552-016-0398-6

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1-58. https://dl.acm.org/doi/10.1145/1541880.15.41882

Golan, R. & Karnin, Z. (2018). A deep learning framework for anomaly detection in multivariate time-series data. arXiv preprint arXiv:1802.03903. https://arxiv.org/abs/18.02.03903

Kim, H., & Lee, Y. (2018). A comprehensive survey on deep learning-based anomaly detection techniques. Journal of Information Security and Applications, 42, 1-17. https://www.sciencedirect.com/science/article/pii/S2214120.X18301092

Sabokrou, M., Minaee, S., & Anwer, M. (2018). Anomaly detection in videos using deep learning. 2018 22nd International Conference on Information Fusion (Fusion), 1-6. https://ieeexplore.org/abstract/document/84604167/

Zhang, X. & Liu, X. (2018). Time series anomaly detection a review. ACM Transactions on Knowledge Discovery from Data (TKDD), 12(3), 1-45. https://dl.acm.org/doi/10.1145/32198119.