# DRONE BASED PUBLIC SAFETY SYSTEM

Main Project Report

Submitted by

**Ayisha Nidha**

**Reg No : FIT20MCA-2038**

*Submitted in partial fulfillment of the requirements for the award of the degree of*

***Master of Computer Applications***
***Of***
***A P J Abdul Kalam Technological University***

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®**
**ANGAMALY-683577, ERNAKULAM(DIST)**
**JULY 2022**

# DECLARATION

I, **Ayisha Nidha** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Application is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

**Date :** **Signature :**

**Place: Angamaly** **Name :**

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®**
**ANGAMALY, ERNAKULAM-683577**

**DEPARTMENT OF COMPUTER APPLICATIONS**

Focus on Excellence

## CERTIFICATE

This is to certify that the project report titled **'Drone Based Public Saftey System'** submitted by **Ayisha Nidha [Reg No: FIT20MCA-2038]** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by her during the year 2022.

**Project Guide:**
**Dr.Sujesh P Lal**

**Head of the Department:**

**Dr.Deepa Mary Mathews**

**Submitted for Viva Voce held on: ………………………..**

**External Examiner: ………………………………………**

# ACKNOWLEDGEMENT

I am deeply grateful to **Dr. Manoj George** , Principal, FISAT, Angamaly for providing me with adequate facilities, way and means by which I was able to complete our main project work and I express my sincere thanks to
**Dr. C Sheela** ,Vice Principal FISAT, Angamaly.

My sincere thanks to **Dr. Deepa Mary Mathews**, Head of the department,MCA, FISAT, who had been a source of inspiration. I express heartiest thanks to **Dr. Sujesh P Lal**, Asst.Professor(Special Grade), FISAT my project guide for the encouragement and valuable suggestion . I express my heartiest gratitude to my scrum master **Ms. Manju Joy** , Asst.Professor(Senior Grade), FISAT and the faculty members in the department for their constant encouragement and never ending support throughout the project. I would also like to express my sincere gratitude to the lab faculty members for their guidance.

Finally I express my thanks to all my friends who gave me wealth of suggestions for successful completion of this project.

## ABSTRACT

Violent action recognition has significant importance in developing automated video surveillance systems. Over last few years, violence detection such as fight activity recognition is mostly achieved through hand-crafted features detectors. Some researchers also inquired learning based representation models. Deep representation based approaches have been successfully used in image recognition and human action detection tasks. This project gives a deep representation based model using concept of transfer learning for violent scenes detection to identify aggressive human behaviors

Recently, the number of violence-related cases in places such as remote roads, pathways, shopping malls, elevators, sports stadiums, and liquor shops, has increased drastically which are unfortunately discovered only after it’s too late. The aim is to create a complete system that can perform realtime video analysis which will help recognize the presence of any violent activities and notify the same to the concerned authority, such as the police department of the corresponding area. Using the deep learning networks CNN along with a well-defined system architecture, we have achieved an efficient solution that can be used for real-time analysis of video footage so that the concerned authority can monitor the situation through a mobile application that can notify about an occurrence of a violent event immediately.

# Contents

# Chapter 1

# INTRODUCTION

On the rapid growth of cities, there is an increased concern for law enforcement. Surveillance cameras are installed in every nook and corner of the city to monitor. The state-of-the-art approaches for violence detection in surveillance videos consider audio features extracting, spatiotemporal analysis with and MoSIFT action descriptors, or definition of optical flow motion vectors. Despite the promising results with near 85 percentage accuracy rates, current methods still lack precision, are memory demanding and bear the considerable computational cost which makes them inapplicable for real purposes, particularly surveillance, where high accuracy must be accompanied by the timeliness of the results.

Monitoring the surveillance time to time is cumbersome to the security personnel as they have to painstakingly go through the footage from one camera to another or view it in real-time to detect violent activities before or as they are occurring. The quality of surveillance videos is also a major constraint for this.

We live in a society that relies heavily on the usage of CCTV cameras for ensuring high-security levels. However, such an approach is highly controversial as we usually use CCTV footages only hours or even days after the incident have already happened. It provides valuable evidence in court but is rarely used to prevent crime or react to it in real-time. The reason for such inefficiency is that the task of monitoring huge quantities of CCTV footage is mainly performed by a limited number of security staff members. Fatigue, worker boredom, and discontinuity of observation make human supervision unreliable. In this project computer vision is used to automate and facilitate the processing of visual information in terms of security monitoring and detection of violent scenes.

Monitoring include the behavioral analysis of people, whether their activities are suspicious or normal. Surveillance cameras are already deployed in most of the public places and private institutes. The efficient violent detection technique can help the government or authorities to take a fast and formalized approach to identify the violence and to prevent the destruction caused to human life and public property. As we all are a part of society, we want safe streets, neighborhoods, and work around us. Deep learning is better than the machine learning technique as it does not require any explicit feature engineering.

Violence detection system using computer vision employs convolutional neural network(CNN), and analyzes the surveillance camera videos. In the proposed system, the violence detection is done basically in three steps.First step is to divide a whole video into segments and frames. Secondly, detect the objects from the video frames. Thirdly, extract the features of the video according to the applied method. Lastly, detects the anomalous activity from the frames.

## 1.1 SYSTEM STUDY

The cross platform source code editors used in this project are Visual Studio Code and Sublime.

**Visual Studio Code**

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 70 percentage of 82,000 respondents reporting that they use it.

**Sublime**

Sublime Text is a shareware cross-platform source code editor. It natively supports many programming languages and markup languages. Users can expand its functionality with plugins, typically community-built and maintained under free-software licenses. To facilitate plugins, Sublime Text features a Python API.

Tensorflow and keras are the software libraries used.

**Tensorflow**

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

### Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the Xception deep neural network model.

# Chapter 2

# PROOF OF CONCEPT

The initial work for violence classification mainly revolved around audio-video correlation [10], detecting the presence of blood, vigorous degrees of motion, and identifying sound features such as screams [11, 12]. Some earlier methods rely on detecting the presence of highly relevant objects (e.g., gun shooting, blaze, blood, blast) rather than directly recognizing the violent events [4]–[5]. In 2011, Nievas et al. firstly released two video datasets for violence detection, namely the Hockey Fight dataset [6] and the Movies Fight dataset. Later, Hassner et al. [7] proposed the Crowd Violence dataset in 2012. Since then, most works turn to develop methods that directly recognize violence in videos

Shakil et al. [8] used pre-training modules with different deep learning methods to detect large-scale violence such as riots in the streets. They combined ResNet50 net and long short-term memory (LSTM) net together for violence detection. They achieved an average accuracy of 97.06 percentage based on their collected database. Eknarin et al. [9] proposed a deep learning method based on time series images. By running a deep convolution neural network on the video database, the violence level of the video database can be classified. They obtained an average accuracy of 88.74 percentage based on the movie database.

Carneiro et al. [13] focused to implement a hand-drawn high-level description and multi-streambased learning model to solve the conflict detection problem in videos. One of the recent works in this field proposed [14] a system that works on the HOG features of video frames. The authors extracted HOG features from binary images and used the random forest classifier to identify the existence of violence in each frame. Finally, they employed the majority voting technique to classify the video clip into violence or non-violence. Although this system doesn't require a GPU for computations and establishes improved results compared to previous works, it suffers from low accuracy.

Recent research work on fight and violence detection shows the extensive implementation of deep learning architectures such as convolutional neural networks (CNNs) [15], long short-term memory (LSTMs), and two stream CNNs [16]. These automatic methods perform much better than the hand-crafted algorithms used for spatio-temporal feature extraction. Mumtaz et al. [17] used transfer learning with GoogLeNet (Inception) [18] which consisted of 22 layers, over the two popular datasets-Hockey and Movies. Both of the datasets have their own complexity. The annotated videos were converted into labelled image frames, and the 1000 class layer is replaced with two categories (violence and non-violence). The result shows the accuracy of 99.28 percentage and 99.97 percentage in the Hockey and Movies dataset respectively.

In [19], Perez et al. proposed a methodology of feature extraction with the two-stream based solution, consisting of two different 2D-CNN models. One is trained with the RGB frames of the video and the other one is trained with a stack of optical flows from the video frames. Violence detection using CNN and LSTM has been employed in [20] where the author combines the three benchmark datasets. The methodology involves the use of a pre-trained CNN architecture VGG-19 [21] followed by LSTM which works with an input of 30 frames at a time. The results from CNN of each frame are grouped and then fed to the LSTM as a sequence. The model performs well, with an accuracy of 94.765 percentage on their combined dataset. The methodology proposed in [22] involves the use of CNN along with ConvLSTM for characterizing the videos. They have made use of the AlexNet model pre-trained on the ImageNet database as the CNN model for feature extraction. Their results have been proved with the three benchmark datasets, achieving the maximum 100 percentage accuracy with the Movies dataset.

The methodology from [23] focuses on creating a new localized guided fight action detection framework for realistic surveillance videos. A new dataset with 1520 videos was proposed and state-of-the-art models were trained over the dataset to achieve high levels of accuracies. They have used the pre-trained SSD-VGG16 for human detection and the pretrained FlowNet 2.0 [24] model for estimating optical flow. A two-stream C3D network [25] is trained on the active regions extracted from the localization phase which are later combined for predictions. Interestingly, Serrano et al. [26] proposed a method where a video sequence is summarized into a representable image, which can be used to classify the scene as violent or non-violent. Additionally, the methodology leverages those zones that could be important for the classification.

# Chapter 3

# IMPLEMENTATION

The prime step of developing this model is to design a deep learning CNN model.The proposed model uses YOLO V3 for training purpose, CNN for image classification, and Open CV for loading the camera module.

## 3.1 SYSTEM ARCHITECTURE

The CNN deep model is originally data driven, it requires large labeled dataset for training. On the other hand, providing insufficient amount of data would not leverage CNN model to learn optimal deep features instead network suffer from significant overfitting issue. To solve the problem of overfitting for small dataset, utilizing modern deep learning network architectures, the approach of transfer learning comes into play. In which, existing network architecture with pre-trained learned features as source task network is employed to build new target task network architecture for limited dataset[1].

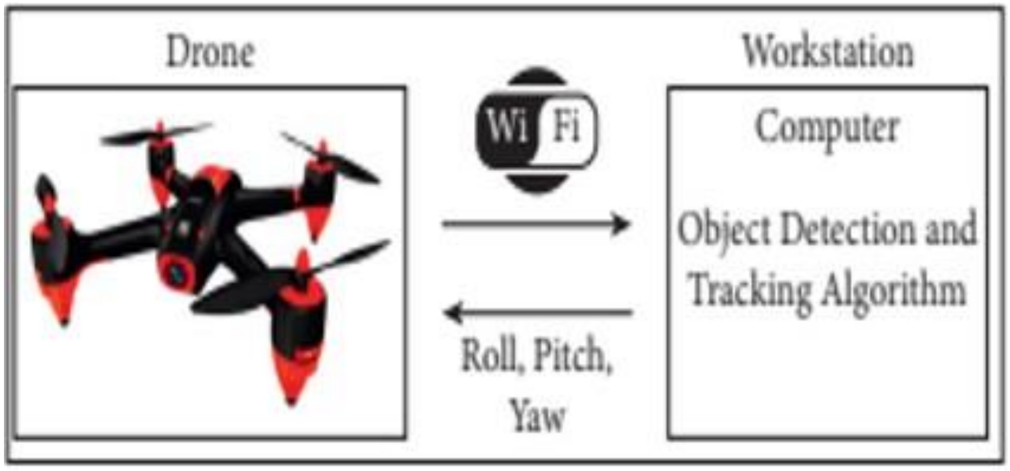


Figure 1: Basic architecture of drone surveillance system.

Figure 3.1: Basic architecture of drone surveillance system

System architecture consist of 5 modules namely Data preparation module, dataset, Deep Leaning model, video engine and database. This system is implemented in python and TensorFlow as a backend. User gives video file as an input and system gives output as video is violent or nonviolent. System supports .mp4 and .avi video formats. Modules: This system is divided into five parts according to functions performed by individuals.

Data Preparation: This module deals with raw video data. It consists of two submodules Data Augmentation and Data Annotation.

Data Augmentation: It is a method of augmenting the available data. Main purpose of augmentation is to increase the size of available dataset.

Data Annotation: System is based of supervised learning so annotation is an important module which labels the data.

Dataset: Dataset consist of data prepared by data preparation module. Dataset is further split into training and testing.

Deep Learning Model: This is the deep learning model trained using input dataset. This model will be invoked by video engine and model will classify input as violent or nonviolent.

Video Engine: This module is an interface between user and deep learning model. The video engine will take the input from user and will pass it through the DL model.

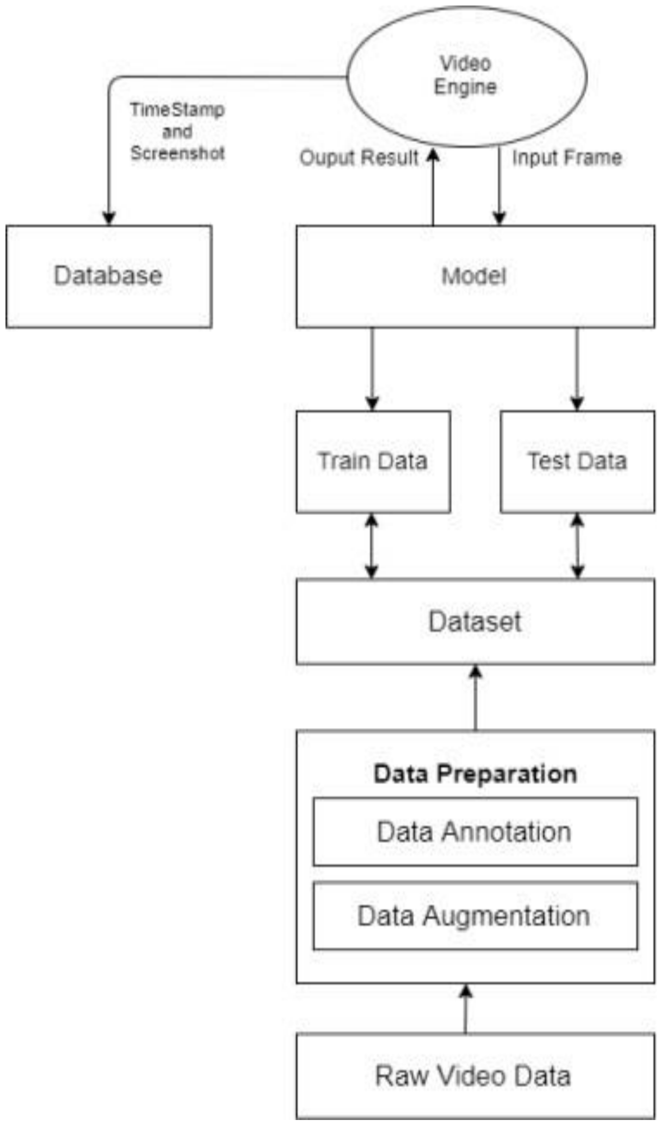


Figure 3.2: System Architecture

## 3.2 DATASET

Datasets containing the images of different types of violent actions, was taken from www.kaggle.com. The dataset obtained had 14 folders containing images of different types of violences as well as folders containing normal pictures.

The dataset obtained have 14 folders. The folders included contains the pictures of acts like Arrest,Bulgary,Fighting, Normal Pictures, Stealing, Robbery. Each folder containing about approximately 3500 images, and the dataset used for the training of the project consisting of approximately 49000 images.

## 3.3 MODULES

### 3.3.1 STRATEGY

In the first phase of this deep learning project was to collect the data; data acquisition. Later the data preprossessing followed by model training was done. As the user uploads a video image or a CCTV image this system reads the video frame by frame as in order to analyze each activity in the video and to detect whether the activity is normal or abnormal. Each of these phases can be split into several steps.

### 3.3.2 DATASET PREPARATION AND PREPROCESSING

Data is an inevitable part of any deep learning project. The second stage of the project involves the processing of the data. Each image stack is first resized to 128 by 171 pixels before they are square cropped to 112 by 112 px according to the input shape of the 3D CNN.

### Data collection

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.

### Data visualization

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

### Data selection

Data selection is defined as the process of determining the appropriate data type and source, as well as suitable instruments to collect data. Data selection precedes the actual practice of data collection.

### Data preprocessing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities.The proposed method deals with image data using Numpy and OpenCV. penCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

### 3.3.3 MODELING

During this stage, numerous data were taken to train the model.Max pooling operation for 2D spatial data was used in this projects. Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by poolsize ) for each channel of the input. The window is shifted by strides along each dimension for the input-ed video

### 3.3.4 MODEL DEPLOYMENT

A pretrained model was deployed to train this system to give appropriate predictions, using the keras interface. Keras is used for creating deep models which can be productized on smartphones. Keras is also used for distributed training of deep learning models.

**Layers in CNN**

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

1- Input Layer: This layer holds the raw input of the image with width 32, height 32, and depth 3.

2- Convolution Layer: This layer computes the output volume by computing the dot product between all filters and image patches. Suppose we use a total of 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.

3- Activation Function Layer: This layer will apply an element-wise activation function to the output of the convolution layer. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.

4- Pool Layer: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation

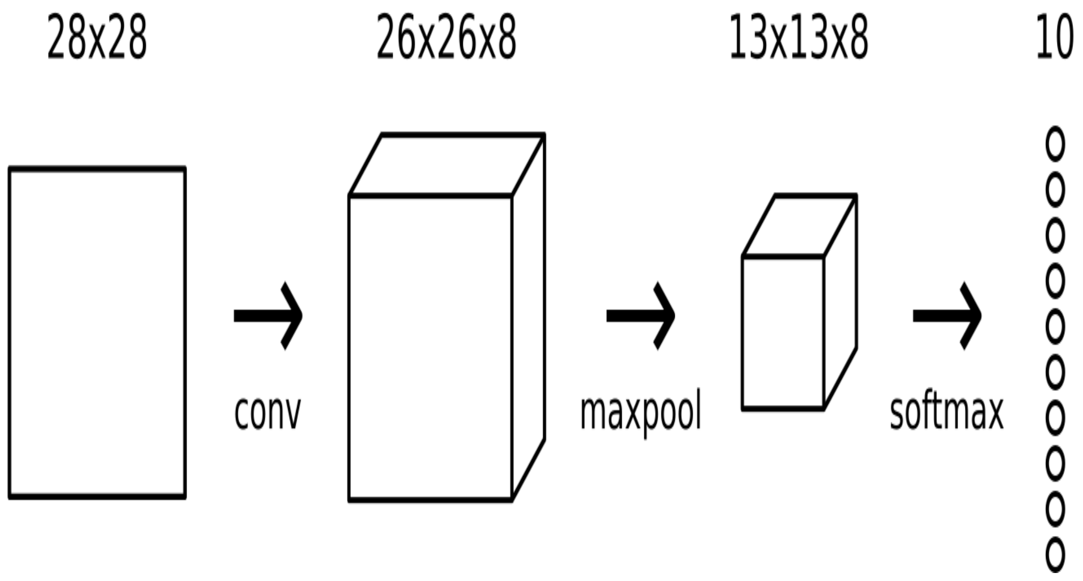


Figure 3.3: Layers in CNN

fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

## 3.4 ALGORITHM

### 3.4.1 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The ”full connectivity” of these networks make them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes[9] [10] [11] [12] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

### 3.4.2 INTRODUCTION

A convolutional neural network is used to extract frame level features from a video. The frame level features are then aggregated using a variant of the long short term memory that uses convolutional gates. The convolutional neural network along with the convolutional long short term memory is capable of capturing localized spatio-temporal features which enables the analysis of local motion taking place in the video. We also propose to use adjacent frame differences as the input to the model thereby forcing it to encode the changes occurring in the video. Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as images generally have red, green, and blue channels).

Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

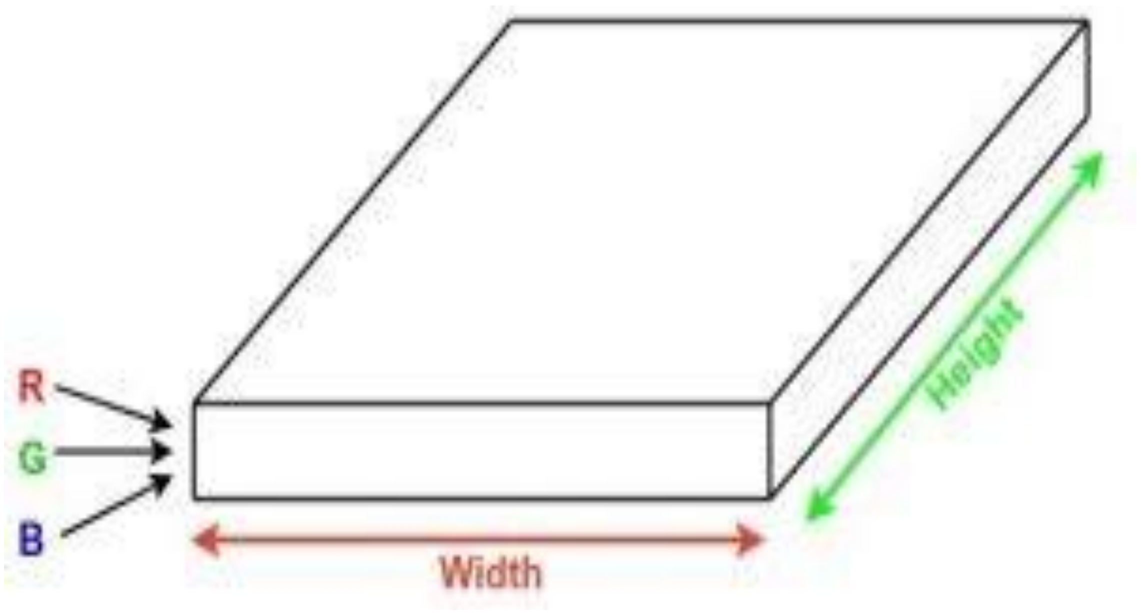


Figure 3.4: Deep Learning Udacity

The performance of this feature extraction pipeline is evaluated on three standard benchmark datasets in terms of recognition accuracy. Comparison of the results obtained with the state of the art techniques revealed the promising capability of the proposed method in recognizing violent videos.

### 3.4.3 ARCHITECTURE OF THE MODEL

— CNN and LSTM: - it uses the Convolutional neural network as a spatial features extractor, as suggested in CNN considered the best spatial features extractor that outperformed almost all the kind of handcraft methods for spatial features extraction, then the extracted features feed into LSTM Layer to learn the temporal relation than using any classification layer such as ANN or any other approach for learning and classification. This approach can benefit from transfer learning by using a pre-trained model in the CNN layer such as vgg19, resnet, and other pre-trained models to extract the general spatial features.[2] — Conv3D several studies show the excellent ability for Conv3d to learn spatiotemporal relation, and it was able to outperform the (CNN and LSTM) approach. Conv3D convolved on four dimensions the time(frame) and height and width and colors channel. It is simple, fast, and more straightforward to train then (CNN and LSTM), the study shows that Conv3D with enough data is the best architecture for action recognition. —Convlstm it extends the LSTM model to have a convolutional structure in both input-to-state and state- to-state transitions. ConvLSTM can capture spatiotemporal correlations consistently.[2]

A basic convolutional neural network can be viewed as a series of convolutional layers, followed by an activation function, followed by a pooling (downscaling) layer, repeated many times.

With the repeated combination of these operations, the first layer detects simple features such as edges in an image, and the second layer begins to detect higher-level features. By the tenth layer, a convolutional neural network is able to detect more complex shapes such as eyes. By the twentieth layer, it is often able to differentiate human faces from one another.

This power comes from the repeated layering of operations, each of which can detect slightly higher-order features than its predecessor.

### 3.4.4 YOLO

YOLO is a Deep Learning architecture proposed by Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi in the paper ‘You Only Look Once: Unified, Real-Time Object Detection’ [3] uses a totally different approach. It is a clever convolutional neural network (CNN) for object detection used in real-time.

Further, It is popular because it has a very high accuracy while also being able to run in real-time or used for real-time applications. The YOLO algorithm “only looks once” at the input image that is it needs only one forward propagation pass through the network to make the predictions.

YOLO is a Convolutional Neural Network (CNN) for performing object detection in real-time. CNNs are classifier-based systems that can process input images as structured arrays of data and identify patterns between them (view image below). YOLO has the advantage of being much faster than other networks and still maintains accuracy.

It allows the model to look at the whole image at test time, so its predictions are informed by the global context in the image. YOLO and other convolutional neural network algorithms “score” regions based on their similarities to predefined classes.

High-scoring regions are noted as positive detections of whatever class they most closely identify with. For example, in a live feed of traffic, YOLO can be used to detect different kinds of vehicles depending on which regions of the video score highly in comparison to predefined classes of vehicles.

It contains 53 convolutional layers which have been, each followed by batch normalization layer and Leaky ReLU activation. Convolution layer is used to convolve multiple filters on the images and produces multiple feature maps No form of pooling is used and a convolutional layer with stride 2 is used to downsample the feature maps. It helps in preventing loss of low-level features often attributed to pooling.

The YOLOv3 algorithm first separates an image into a grid. Each grid cell predicts some number of boundary boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes.

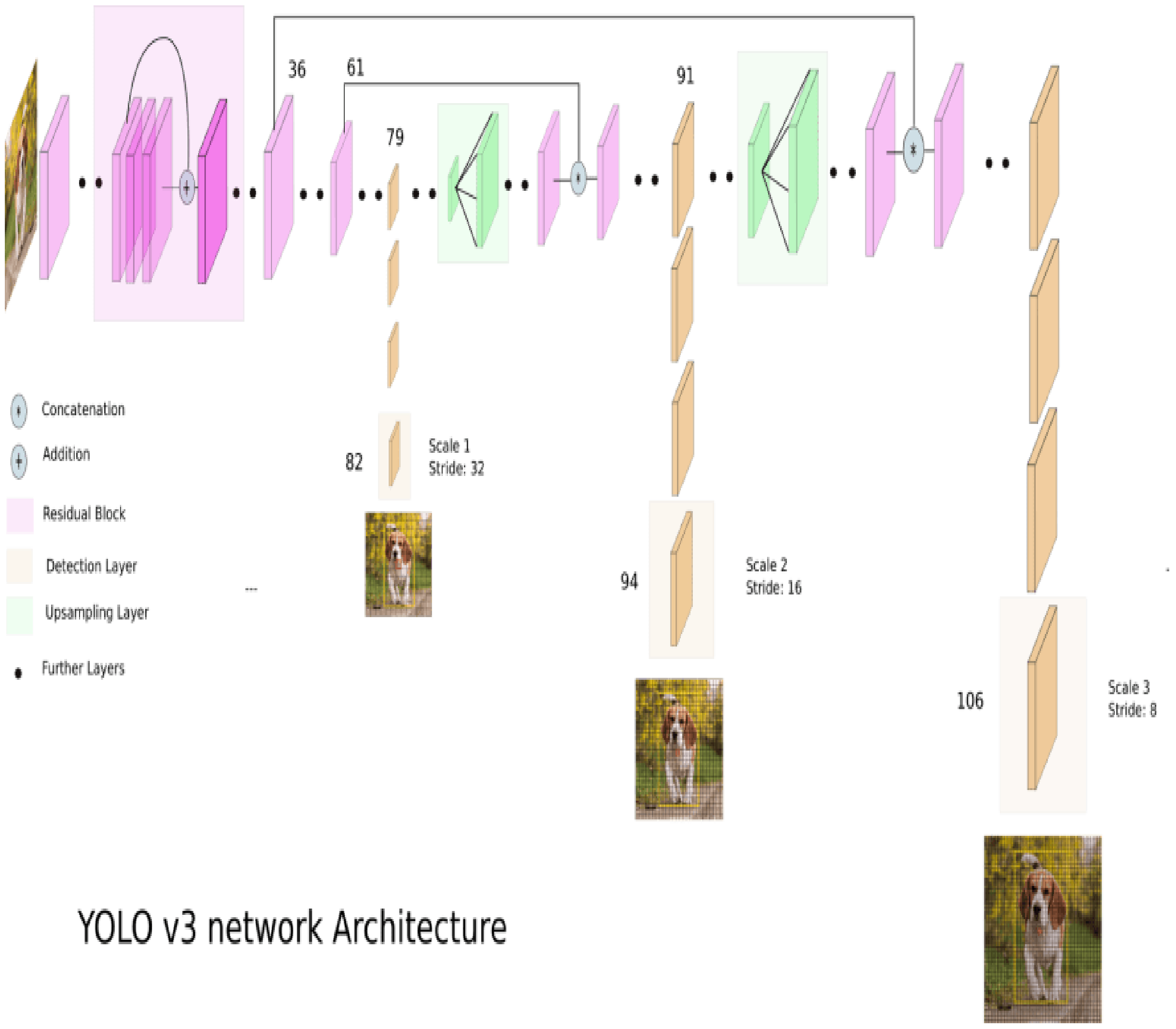


Figure 3.5: Architecture of YOLO

Each boundary box has a respective confidence score of how accurate it assumes that prediction should be and detects only one object per bounding box. The boundary boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes.

Other comparable algorithms that can carry out the same objective are R-CNN (Region-based Convolutional Neural Networks made in 2015) and Fast R-CNN (R-CNN improvement developed in 2017), and Mask R-CNN.

However, unlike systems like R-CNN and Fast R-CNN, YOLO is trained to do classification and bounding box regression at the same time.

**Training for Classification**

The classification training here is all pre-training on ImageNet, which mainly includes two steps. The data set is ImageNet, 160 Epochs are trained, the input image size is 224×224, and the initial learning rate is 0.1. Standard data increment methods are used during the training process, such as random cropping, rotation, and chroma and brightness adjustments.

**Fine-tune the network**

At this time, using a 448×448 input, all parameters remain unchanged except for the epoch and the learning rate.

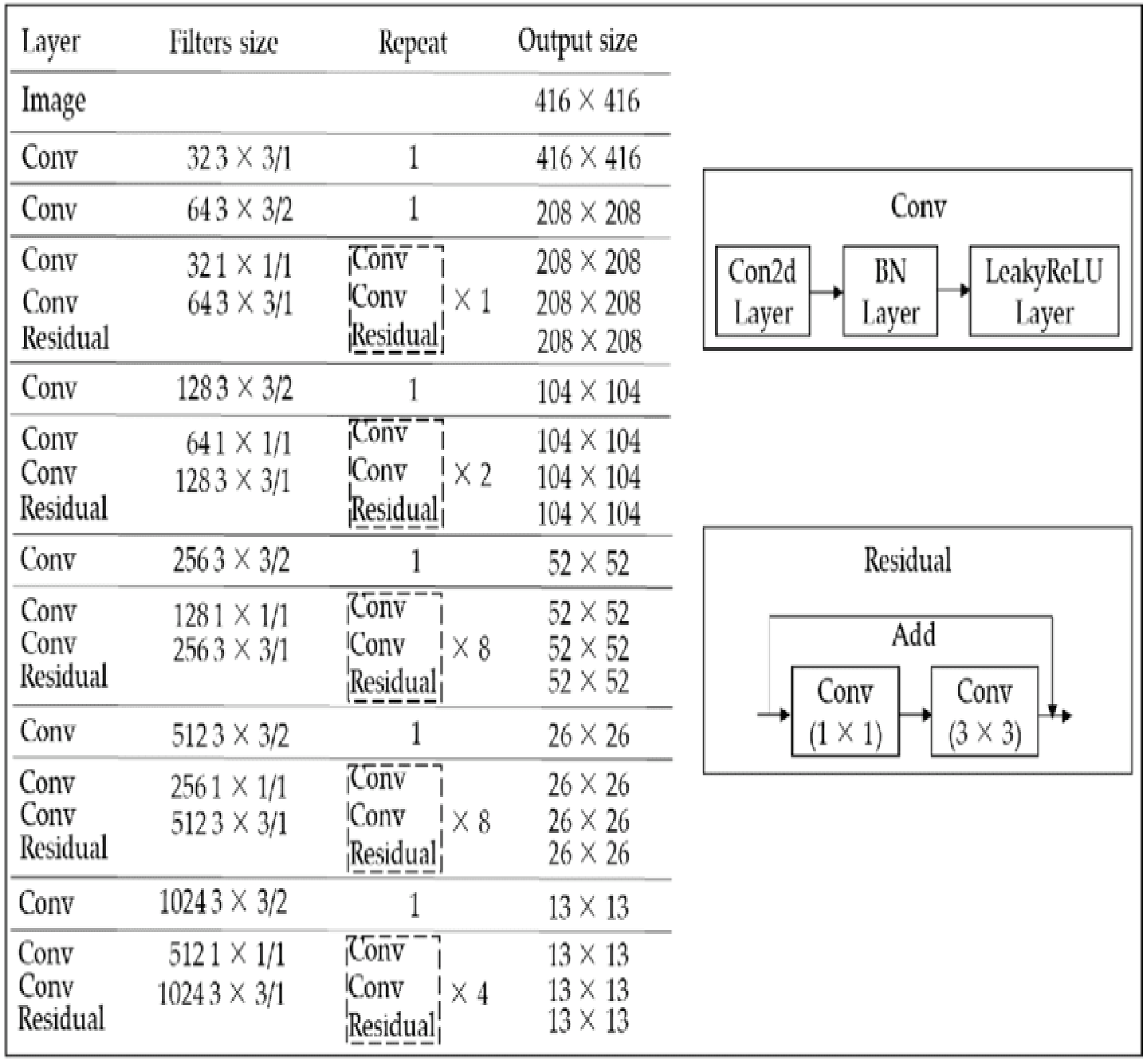
Here, the learning rate is changed to 0.001, and training is performed ten times.  The results show that the accuracy of top-1 and top-5 after fine-tuning are 76.5percentage and 93.3percentage, respectively. According to the original training method, the accuracy of Darknet-19's top-1 and top-5 is 72.9percentage and 91.2percentage.

**Training for Detection**

First, delete the last convolution layer, and then add three convolution layers 33. Each convolution layer has 1024 filters, and each convolution layer is connected to 11 convolution layers. The category probabilities of the two boxes corresponding to this cell are the same, but in YOLO V2, the category probabilities belong to this box, and each box corresponds to a category probability instead of a grid.

Compared with YOLO V2, YOLO V3 has two points: using multi-scale features for object detection and adjusting the basic network structure.

On the one hand, YOLO V3 adopts feature graphs of three scales (when the input is (416×416), (13×13), (26×26) and (52×52)). YOLO V3 uses three prior boxes for each position, so K-means is used to get nine prior boxes and divide them into three scale feature maps. Feature maps with larger-scale use smaller prior boxes. On the other hand, YOLO V3 feature extraction network used the residual model. Compared with Darknet- 19 used by YOLO V2, it contained 53 convolution layers, so it was called Darknet-53. YOLO V4 style has a significant change, more focus on comparing data, and has a substantial improvement.



| Layer | Filters size | Repeat | Output size |
|---|---|---|---|
| Image | | | 416 × 416 |
| Conv | 32 3 × 3/1 | 1 | 416 × 416 |
| Conv | 64 3 × 3/2 | 1 | 208 × 208 |
| Conv | 32 1 × 1/1 | Conv | 208 × 208 |
| Conv | 64 3 × 3/1 | Conv × 1 | 208 × 208 |
| Residual | | Residual | 208 × 208 |
| Conv | 128 3 × 3/2 | 1 | 104 × 104 |
| Conv | 64 1 × 1/1 | Conv | 104 × 104 |
| Conv | 128 3 × 3/1 | Conv × 2 | 104 × 104 |
| Residual | | Residual | 104 × 104 |
| Conv | 256 3 × 3/2 | 1 | 52 × 52 |
| Conv | 128 1 × 1/1 | Conv | 52 × 52 |
| Conv | 256 3 × 3/1 | Conv × 8 | 52 × 52 |
| Residual | | Residual | 52 × 52 |
| Conv | 512 3 × 3/2 | 1 | 26 × 26 |
| Conv | 256 1 × 1/1 | Conv | 26 × 26 |
| Conv | 512 3 × 3/1 | Conv × 8 | 26 × 26 |
| Residual | | Residual | 26 × 26 |
| Conv | 1024 3 × 3/2 | 1 | 13 × 13 |
| Conv | 512 1 × 1/1 | Conv | 13 × 13 |
| Conv | 1024 3 × 3/1 | Conv × 4 | 13 × 13 |
| Residual | | Residual | 13 × 13 |

Figure 3.6: Architecture Diagram of YOLO

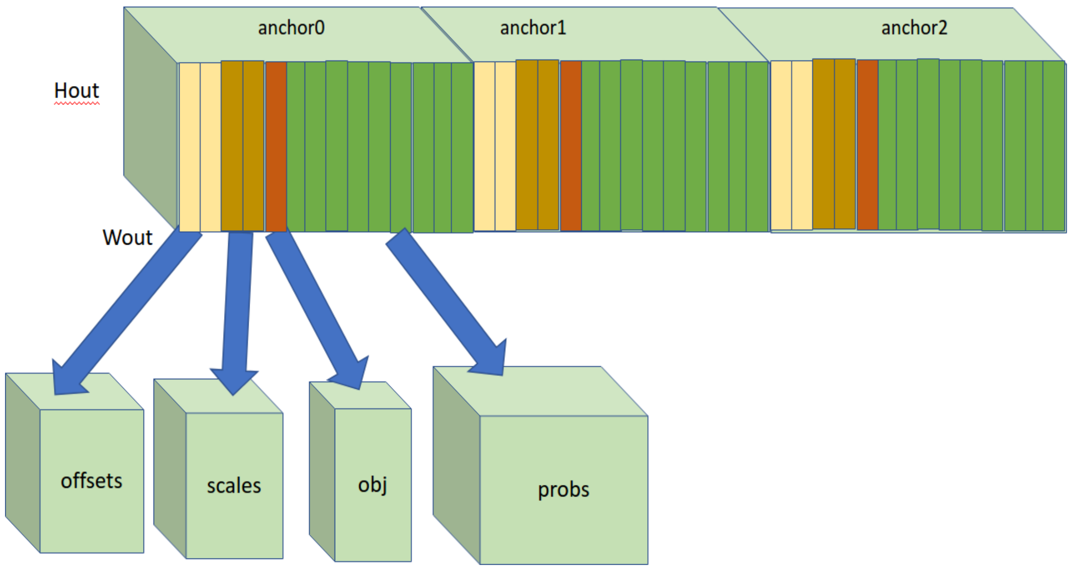


Figure 3.7: YOLO V2 and YOLO V3 output layer. Wout and Hout are spatial dimensions of the output feature map. For each anchor, the features are arranged in the described order.

# Chapter 4

# RESULT ANALYSIS

This project would cover all the security cameras that are installed in the authority's locality under control. During any occurrence of violence, the officials governing that locality are immediately notified. The alert received provides all the crucial information required to take any necessary action immediately. It also provides a few snapshots of the area that can help them make better decisions on how to bring the situation under control. Every functionality is tested in real-time and has been connected to work with the proposed deep learning model

Observed that the model trained on the benchmark datasets do not work accurately with the real time CCTV footage. It is mainly due to the fact that the videos are unrealistic and do not aptly depict the real world scenarios. These videos differ a lot from the actual CCTV ones in terms of the camera angle too. To overcome this and validate the architecture for real-time analysis, the UCF Crime dataset was taken which makes the model perform better in real-time. The modified UCF Crime dataset consists of an equal number of 160 trimmed violence and non-violence videos. Trained the model on this for 50 epochs, with a train/test split scheme with 80 percentage for training and 20 percentage for testing.

Results were achieved from the testing set of accuracy of 98.87 percentage, which is better than the results that were obtained using the benchmark data sets and the other existing systems. The loss depicted by both train and test data sets are calculated using categorical cross-entropy. The proposed model trained takes approximately 7.89 milliseconds per frame for the classification (without including the time for preprocessing). Overall, the proposed method takes approximately 0.28 seconds for processing of a 3-second video clip at 30 fps. The use of the UCF Crime dataset and fast processing time makes it suitable for violence detection in real-time video processing applications. The entire experiment was performed on a 12 GB NVIDIA Tesla K80 GPU.

# Chapter 5

# CONCLUSION AND FUTURE SCOPE

## 5.1 Conclusion

Nowadays, the rate of violence around us is increasing drastically, acting as a threat to humans, buildings, and systems. There has always been a need for a better system that can aid the police in monitoring the violence, which is usually hard to handle as it is a group activity and the process of elimination to find the culprit is time-consuming. The results from our experiment demonstrate the effective use of CNN architecture, for training the model a modified UCF Crime dataset. This work provides a fully integrated system that can help the police authorities monitor their area under control.

This model that has been trained over the benchmark datasets has shown very powerful results. The excellent results over the UCF Crime dataset with a good speed as compared to other approaches makes our model function accurate in real-life scenarios. Apart from producing a novel model for violence detection, we have constructed a fully functioning system that supports this model to work well in the real-world, making our approach unique, detailed, and advanced from the existing solutions. The system precisely alerts the police through the app during any occurrence of violence and allows the police to take action accordingly. The authority can use our system to manage the crimes around them in a much efficient manner.

This system can be improved to perform better in many aspects. The system can be improved by specifying how severe the detected violence is. This can help the authorities make better decisions. The proposed Deep Learning architecture can be altered by modifying the hyperparameters, to improve the performance.

The planned system solely detects suspicious human behavior and presence of guns. Detection of fire and different weapons will be enforced in future. Cloud readying can be done.

# Chapter 6

# SAMPLE CODE

## 6.1 main.py

```
from flask import Flask
from public import public


app=Flask(__name__)
app.secret_key="abc"

app.register_blueprint(public)

app.run(debug=True,port=5666)
```

## 6.2 newproject.py

```
# coding: utf-8

# In[ ]:
import os

import tensorflow as tf
# sess = tf.Session()
import keras
from keras.engine.saving import load_model
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Dense, Activation, Dropout, Flatten

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np

#------------------------------
# sess = tf.Session()
# keras.backend.set_session(sess)
#------------------------------
#variables
num_classes =14
batch_size = 40
epochs = 50
#------------------------------

import os, cv2, keras
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.engine.saving import load_model
# manipulate with numpy,load with panda
import numpy as np
# import pandas as pd
# data visualization
import cv2
import matplotlib
import matplotlib.pyplot as plt
# import seaborn as sns

# get_ipython().run_line_magic('matplotlib', 'inline')


def read_dataset1(path):
    data_list = []
    label_list = []

    file_path = os.path.join(path)
    img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    res = cv2.resize(img, (48, 48), interpolation=cv2.INTER_CUBIC)
    data_list.append(res)
    # label = dirPath.split('/')[-1]

    # label_list.remove("./training")
    return (np.asarray(data_list, dtype=np.float32))


model = Sequential()

# 1st convolution layer
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPooling2D(pool_size=(5, 5), strides=(2, 2)))

# 2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3, 3), strides=(2, 2)))

# 3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3, 3), strides=(2, 2)))

model.add(Flatten())

# fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax'))
# ------------------------------
# batch process

def predictcnn(fn):
    dataset=read_dataset1(fn)
    (mnist_row, mnist_col, mnist_color) = 48, 48, 1
    dataset /= 255
    dataset = dataset.reshape(dataset.shape[0], mnist_row, mnist_col, mnist_color)

    mo = load_model("model1.h5")

    # predict probabilities for test set

    yhat_classes = mo.predict_classes(dataset, verbose=0)
    return yhat_classes
    #
    # print(yhat_classes)
```

```
data_list = []
label_list = []

file_path = os.path.join(path)
img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
res = cv2.resize(img, (48, 48), interpolation=cv2.INTER_CUBIC)
data_list.append(res)
# label = dirPath.split('/')[-1]

# label_list.remove("./training")
return (np.asarray(data_list, dtype=np.float32))
```

## 6.3 public.py

```
from flask import *
from database import *
# from newcnn import *
from yolo_video import *
import uuid
public=Blueprint('public',__name__)
@public.route('/')
def home():
        return render_template('home.html')


@public.route('/about')
def about():
        return render_template('about.html')



@public.route('/fileupload',methods=['get','post'])
def fileupload():
        data={}

        if "upload" in request.form:
        i=request.files['i']
        path="static/uploads/"+str(uuid.uuid4())+i.filename
        i.save(path)
        q="insert into file values(null,'%s',curdate())"%(path)
        insert(q)
```

```
                    outs=valssss(path)
                    print("hhh",outs)
                    flash(outs," Detected")
                    return redirect(url_for('public.fileupload'))
                    q="SELECT * FROM file "
                    r=select(q)
                    data['users']=r
                    return render_template('fileupload.html',data=data)

@public.route('/detectviolence')
def detectviolence():
                    outs=valssss(0)
                    print("hhh",outs)
                    flash(outs," Detected")
                    return render_template('home.html')
```

## 6.4 database.py

```
import mysql.connector

password=""
database = "drone"
port=3306
def select(q):
	cnx = mysql.connector.connect(user="root", password=password, host="localhost", 
database=database,port=port)
	cur = cnx.cursor(dictionary=True)
	cur.execute(q)
	result = cur.fetchall()
	cur.close()
	cnx.close()
	return result
def update(q):
	cnx = mysql.connector.connect(user="root", password=password, host="localhost", 
database=database,port=port)
	cur = cnx.cursor(dictionary=True)
	cur.execute(q)
	cnx.commit()
	result = cur.rowcount
	cur.close()
	cnx.close()
	return result
     def delete(q):



  cnx = mysql.connector.connect(user="root", password=password,        host="localhost", 
database=database,port=port)
	cur = cnx.cursor(dictionary=True)
	cur.execute(q)
	cnx.commit()
	result = cur.rowcount
	cur.close()
	cnx.close()
```

```
    def insert(q):
    cnx = mysql.connector.connect(user="root", password=password, host="localhost",
database=database,port=port)
    cur = cnx.cursor(dictionary=True)
    cur.execute(q)
    cnx.commit()
    result = cur.lastrowid
    cur.close()
    cnx.close()
    return result
   def insert(q):
    cnx = mysql.connector.connect(user="root", password=password, host="localhost",
database=database,port=port)
    cur = cnx.cursor(dictionary=True)
    cur.execute(q)
    cnx.commit()
    result = cur.lastrowid
    cur.close()
    cnx.close()
    return result
```

## 6.5 yolovideo.py

```
# from dbconnection import *

from new_project import predictcnn
# USAGE
# python yolo_video.py --input videos/airport.mp4 --output output/airport_output.avi --yolo yolo-coco

import uuid
import argparse

import time
import cv2
import os
loopstatus="f"
loopcount=0
def valssss(path):
# while True:
        loopstatus="f"
        loopcount=0
        out="No Violence"

        # qry="SELECT * FROM `videos` WHERE `video_id` NOT IN(SELECT `video id` FROM
`process`)"
        # res=selectall(qry)
        lisclass=["Abuse","Arrest","Arson","Assault","Burglary","Explosion","Fighting",
"NormalVideos","RoadAccidents","Robbery","Shooting","Shoplifting","Stealing","Vandalism"]
        # for i in res:
        # print(i)
        # print(r"C:\Users\ASUS\Downloads\Violenece\c1\src\static\file/"+i[2])
        # vs = cv2.VideoCapture(0)
        vs = cv2.VideoCapture(path)
        writer = None
        (W, H) = (None, None)
        # try to determine the total number of frames in the video file
```

```
   counti=0
   while True:
            if loopstatus=="f":
                     loopstatus=="n"
                     counti=counti+1
                     print("cccii",counti)
                     # read the next frame from the file

                     (grabbed, frame) = vs.read()

                     print("==> ",counti)
                     # if the frame was not grabbed, then we have reached the end
                     # of the stream
                     if not grabbed:
                              break
                     cv2.imwrite("static\\file\\sample.jpg",frame)


ress=predictcnn("static\\file\\sample.jpg")
print("bbb",lisclass[ress[0]])

if lisclass[ress[0]]=="Arson" or lisclass[ress[0]]=="Fighting" or lisclass[ress[0]]==
"Shooting":
                              out="Violence"
                              break


# qry="INSERT INTO `notification` VALUES(NULL,CURDATE(),%s,%s)"
# val=(i[1],lisclass[ress[0]])
# result_id=iud(qry,val)
# cv2.imwrite(r"C:\Users\ASUS\Downloads\Violenece\c1\src\static\result/"+
str(uuid.uuid4())+".jpg", frame)
            else:
                     print("lcccc:",loopcount)
                     loopcount=loopcount+1
                     if loopcount==50:
                              loopcount=0
                              loopstatus="f"
            cv2.imshow('frame', frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                     break
```

```
  # After the loop release the cap object
        vs.release()
        # Destroy all the windows
        cv2.destroyAllWindows()
        return out

# outs=valssss("static/file/Rocky_takes_revenge_on
_Adheera____K.G.F._2_Fight_since___Rocky_vs_Adheera.mp4")
# print("hhh",outs)
        # time.sleep(10)



# C:\Users\ASUS\Downloads\yolo_video.py
```

# Chapter 7

# SCREEN SHOTS

Here I add some sample screenshots of system which includes,

- Home Screen
- Home Screen About
- Detection Screen for Pictures and Video footage
- Live Detection Screen

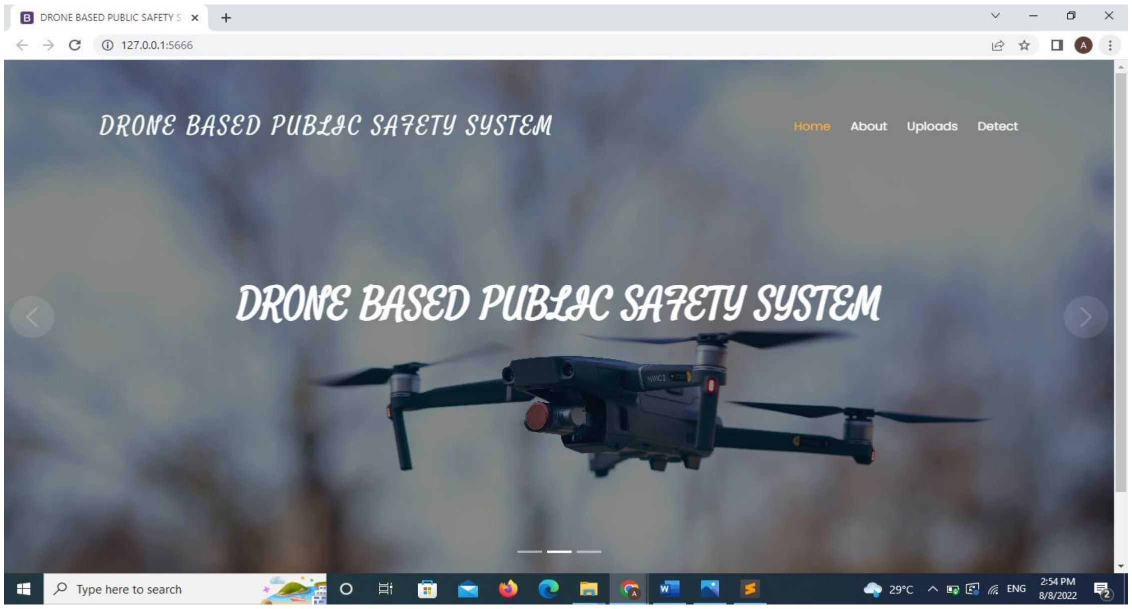
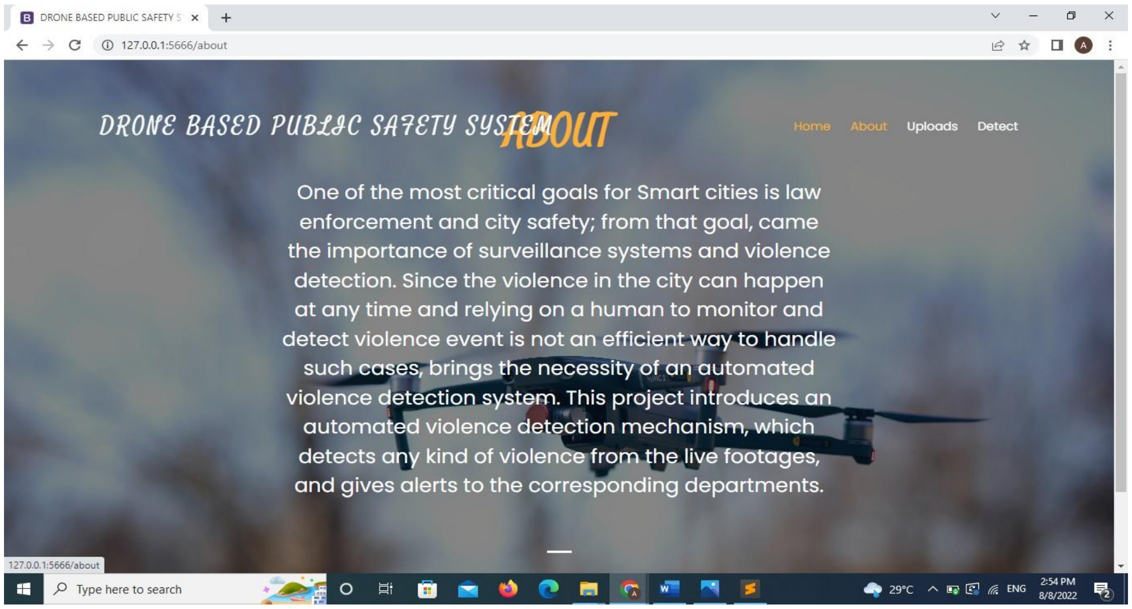
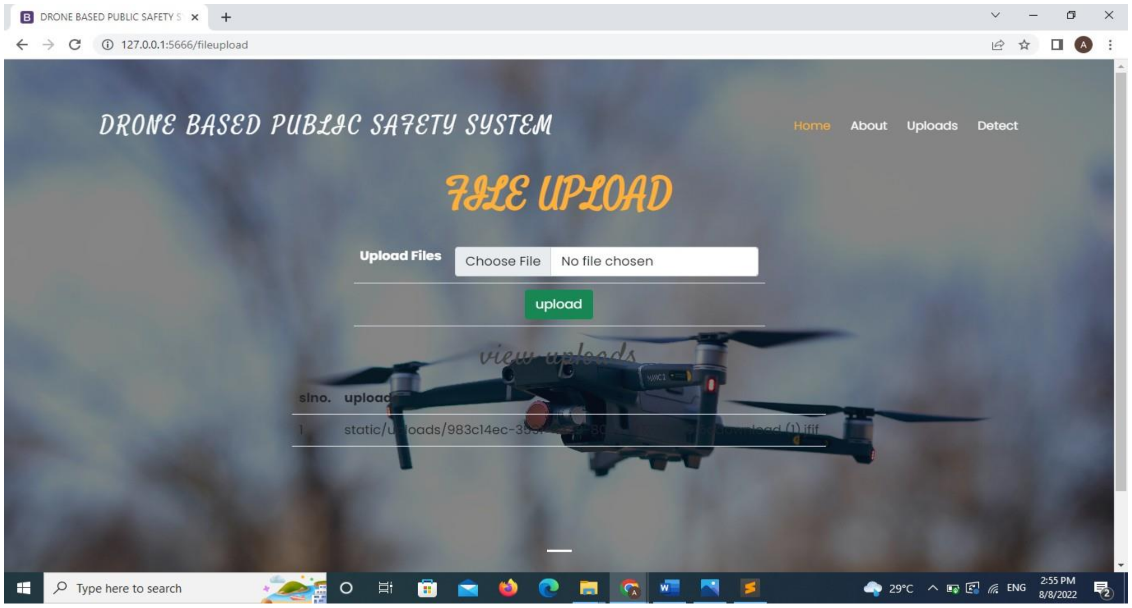

DRONE BASED PUBLIC SAFETY S
127.0.0.1:5666
DRONE BASED PUBLIC SAFETY SYSTEM
Home
About
Uploads
Detect
DRONE BASED PUBLIC SAFETY SYSTEM
Type here to search
29°C
ENG
2:54 PM
8/8/2022

Figure 7.1: Home Screen



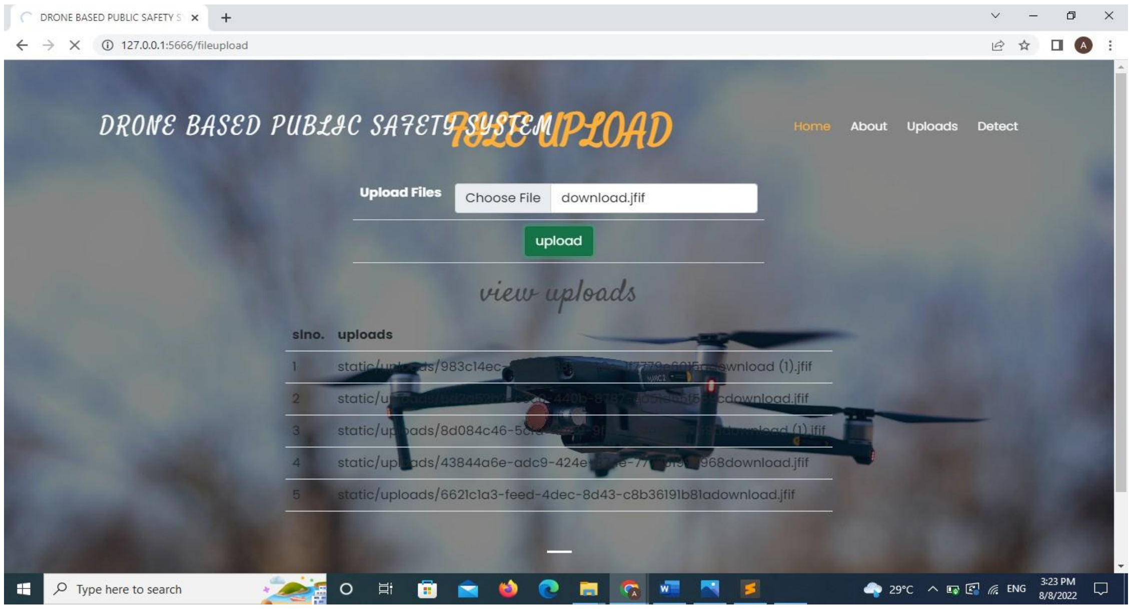
One of the most critical goals for Smart cities is law enforcement and city safety; from that goal, came the importance of surveillance systems and violence detection. Since the violence in the city can happen at any time and relying on a human to monitor and detect violence event is not an efficient way to handle such cases, brings the necessity of an automated violence detection system. This project introduces an automated violence detection mechanism, which detects any kind of violence from the live footages, and gives alerts to the corresponding departments.

Figure 7.2: Home Screen About



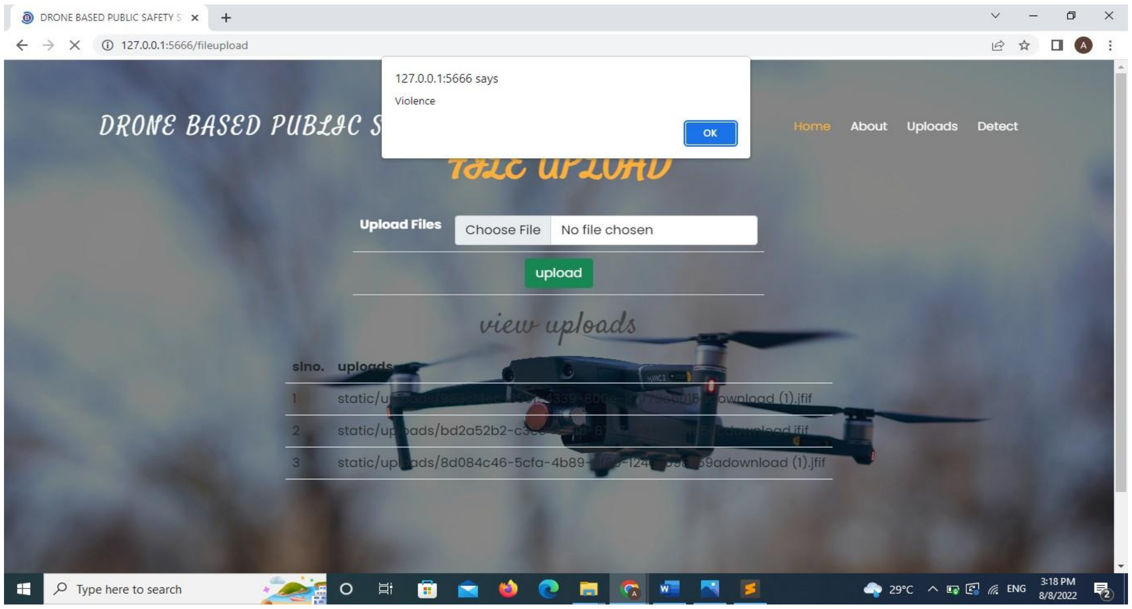
| slno. | uploads |
|---|---|
| 1 | static/uploads/983c14ec-35[illegible]download (1).jfif |

Figure 7.3: Screen to upload images and footage in order to detect violence or non violence



| slno. | uploads |
|---|---|
| 1 | static/uploads/983c14ec-[illegible]f7779a6015a[illegible]ownload (1).jfif |
| 2 | static/u[illegible]-440b-8787-[illegible]cdownload.jfif |
| 3 | static/uploads/8d084c46-5c[illegible]download (1).jfif |
| 4 | static/uploads/43844a6e-adc9-424e[illegible]e-77[illegible]968download.jfif |
| 5 | static/uploads/6621c1a3-feed-4dec-8d43-c8b36191b81adownload.jfif |

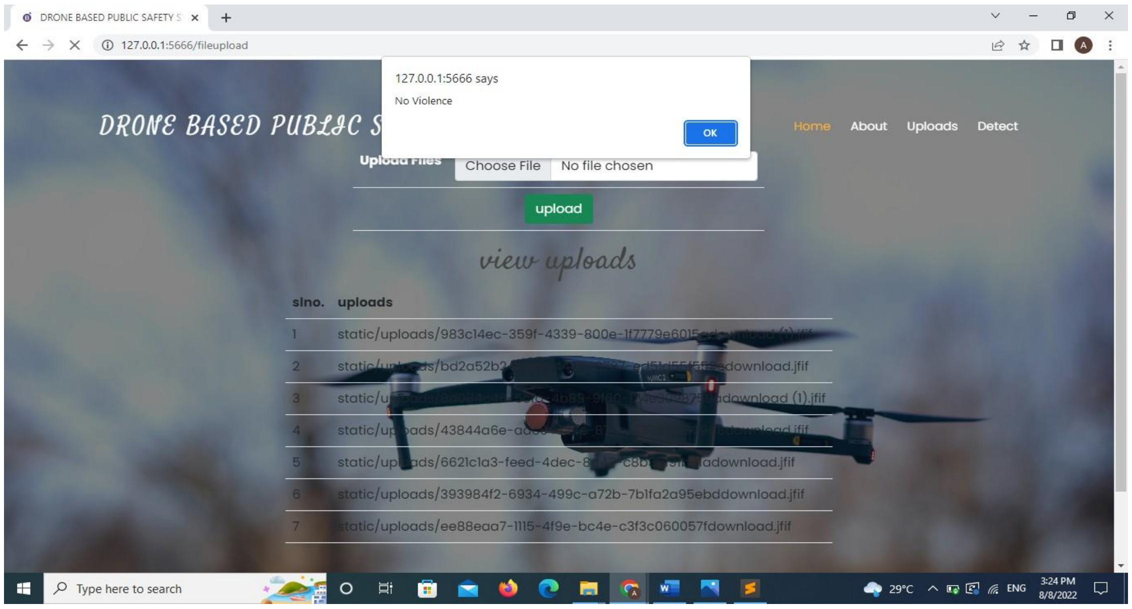
Figure 7.4: Screen showing file chosen


127.0.0.1:5666 says

Violence

| slno. | uploads |
|---|---|
| 1 | static/u[illegible]4339-800e-[illegible]ownload (1).jfif |
| 2 | static/uploads/bd2a52b2-c3[illegible]download.jfif |
| 3 | static/uploads/8d084c46-5cfa-4b89-[illegible]-124[illegible]59adownload (1).jfif |

Figure 7.5: Screen predicting violence


127.0.0.1:5666 says

No Violence

| slno. | uploads |
|---|---|
| 1 | static/uploads/983c14ec-359f-4339-800e-1f7779e6015[illegible] |
| 2 | static/uploads/bd2a52b2[illegible]-ed51d55f55[illegible]download.jfif |
| 3 | static/u[illegible]4b89-9f80-[illegible]download (1).jfif |
| 4 | static/uploads/43844a6e-ad[illegible]download.jfif |
| 5 | static/uploads/6621c1a3-feed-4dec-8[illegible]c8b[illegible]adownload.jfif |
| 6 | static/uploads/393984f2-6934-499c-a72b-7b1fa2a95ebddownload.jfif |
| 7 | static/uploads/ee88eaa7-1115-4f9e-bc4e-c3f3c060057fdownload.jfif |

Figure 7.6: Screen Predicting non violence

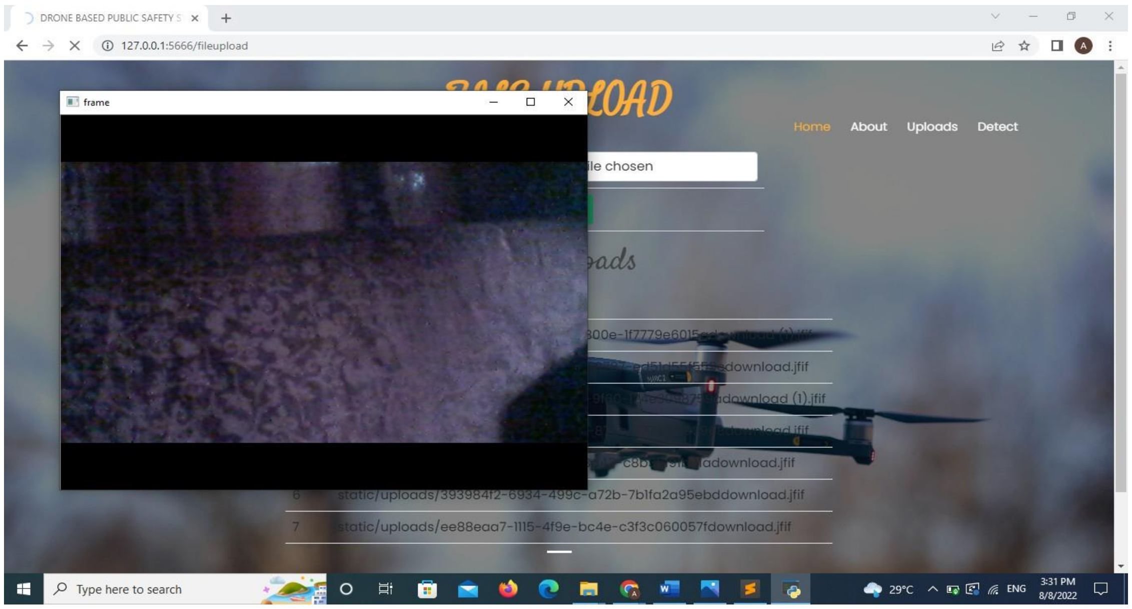

DRONE BASED PUBLIC SAFETY S
127.0.0.1:5666/fileupload
frame
Home
About
Uploads
Detect
ile chosen
6 static/uploads/393984f2-6934-499c-a72b-7b1fa2a95ebddownload.jfif
7 static/uploads/ee88eaa7-1115-4f9e-bc4e-c3f3c060057fdownload.jfif
Type here to search
29°C
ENG
3:31 PM
8/8/2022

Figure 7.7: Screen showing live detction

# Chapter 8

# REFERENCES

[1]A. Mumtaz, A. B. Sargano and Z. Habib, ”Violence Detection in Surveillance Videos with Deep Network Using Transfer Learning,” 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS), 2018, pp. 558-563, doi: 10.1109/EECS.2018.00109.

[2] 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 (AVSS), Lecce, pp. 1-6,2017.

[3]Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi: You Only Look Once: Unified, Real-Time Object Detection, May 2016.

[4] C. Clarin, J. Dionisio, M. Echavez, and P. Naval, “Dove: Detection of movie violence using motion intensity analysis on skin and blood,” PCSC, vol. 6, pp. 150–156, 2005.

[5] E. B. Nievas, O. D. Suarez, G. B. Garcia, and R. Sukthankar, “Hockey fight detection dataset,” in Computer Analysis of Images and Patterns. Springer, 2011, pp. 332–339.

[6] T. Hassner, Y. Itcher, and O. Kliper-Gross, “Violent flows: Real-time detection of violent crowd behavior,” in 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE, 2012, pp. 1–6

[7]D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 4489–4497.

[8] Sumon, S.A.; Goni, R.; Hashem, N.B.; Shahria, T.; Rahman, R.M. Violence detection by pretrained modules with different deep learning approaches. Vietnam J. Comput. Sci. 2020, 7, 22–23.

[9] Eknarin, D.; Luepol, P.; Suwatchai, K. Video Representation Learning for CCTV-Based Violence Detection. In Proceedings of the 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 12–14 December 2018.

[10] Y. Gong, W. Wang, S. Jiang, Q. Huang, and W. Gao, “Detecting violent scenes in movies by auditory and visual cues,” Pacific-Rim Conference on Multimedia, 2008, pp. 317–326.

[11] T. Giannakopoulos, D. Kosmopoulos, A. Aristidou, and S. Theodoridis, “Violence content classification using audio features,” Hellenic Conference on Artificial Intelligence, 2006, pp. 502–507.

[12] J. Nam, M. Alghoniemy, and A. H. Tewfik, “Audio-visual content-based violent scene characterization,” Proceedings 1998 International Conference on Image Processing (ICIP98), 1998, pp. 353-357.

[13] S. A. Carneiro, G. P. da Silva, S. J. F. Guimara es, and H. Pedrini, "Fight Detection in Video Sequences Based on Multi-Stream Convolutional Neural Networks," 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2019, pp. 8-15.

[14] S. Das, A. Sarker, and T. Mahmud, "Violence Detection from Videos using HOG Features," 2019 4th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 2019, pp. 1–5.

[15] G. Sakthivinayagam, R. Easawarakumar, A. Arunachalam, and M. Pandi, "Violence Detection System using Convolution Neural Network," SSRG International Journal of Electronics and Communication Engineering (IJECE), vol. 6, no. 2, 2019, Art. no. 102.

[16] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," Advances in Neural Information Processing Systems (NIPS), 2014, pp. 568–576.

[17] A. Mumtaz, A. Bux and Z. Habib, "Violence Detection in Surveillance Videos with Deep Network using Transfer Learning," Electrical Engineering and Computer Science (EECS), pp. 558–563, 2018.

[18] C. Szegedy et al., "Going Deeper with Convolutions," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2001, pp. 1–9.

[19] M. Perez and A. C. Kot, "Detection of real-world fights in surveillance videos," International Conference on Acoustics, Speech, and Signal Processing, 2019, pp. 2662-2666.

[20] Al-Maamoon R. A. and R. F. Al-Tuma, "Robust Real-Time Violence De-

tection in Video Using CNN And LSTM," Student Conference on Conservation Science, 2019, pp. 104-108.

[21] Basu S, Mitra S, Saha N. Deep learning for screening covid-19 using chest x-ray images. In2020 IEEE Symposium Series on Computational Intelligence (SSCI) 2020.