A Project Report

On

Human Violence Recognition for Video Surveillance using neural network

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelors of Technology in Computer Science and Engineering

Under The Supervision of

Mr. Tarun Kumar

Assistant Professor

Department of Computer Science and Engineering

Submitted By:

Ritika Mishra

19SCSE1050013

Harsh Lohia

19SCSE1010

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "**Human Violence Recognition for Video Surveillance using neural network**" in partial fulfillment of the requirements for the award of the **Bachelor Of Technology In Computer Science And Engineering** degree submitted in the **School of Computing Science and Engineering** of **Galgotias University**,Greater Noida, is an original work carried out during the period of July – 2021 to December 2021, under the supervision of **Mr. Tarun Kumar**, **Assistant Professor, Department of Computer Science and Engineering**, Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ritika Mishra, 19SCSE1050013
Harsh Lohiya, 19SCSE1180117

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

……,
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **19SCSE1050013 – Ritika Mishra, 19SCSE1180117 – Harsh Lohiya** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

With the growing expansion of surveillance cameras to monitor human activity, a system that automatically recognises violence and suspicious activities is required. In computer vision and image processing, the detection of deviant and aggressive activities has become a hot study issue, drawing new researchers. Different strategies for detecting such behaviours from video have been proposed in recent years, according to the relevant literature. This study examines a variety of cutting-edge violence detection algorithms. The methods of detection in this research are classified into three groups depending on the classification techniques used: violence detection using classical machine learning, support vector machine (SVM), and deep learning. Each method's feature extraction methodologies and object identification approaches are also discussed. Furthermore, the datasets and video characteristics employed in the approaches are explored, as well as their importance in the recognition process. We investigate how difficult it is to recognise human aggressiveness in film, including fistfights, kicking, and striking with objects, among other things. To identify hostility, we employ both motion trajectory and limb orientation information. The temporal derivative of an Acceleration Measure Vector (AMV) composed of motion direction and amplitude is defined as jerk. The results of several data sequences involving different forms of violent acts are given.

Key Words: CNN, SVM,, Data-Set, Deep Learning, Silhouette-Bounding

report

Table of Contents

1. Introduction

Outstanding infrastructure growths have been seen in security-related challenges across the world during the last few decades. As a result of the increased need for security, video-based surveillance has become a critical arena for investigation. By using some equipment, an intelligent video surveillance system primarily censored the performance, events, or ever-changing information in terms of human people, vehicles, or other things from a distance (usually digital camera). Intelligent video processing abilities are capable of scopes including prevention, detection, and intervention that have led to the creation of actual and consistent video surveillance systems. Surveillance systems in the past were more reliant on human operators. Automated methods are favoured these days due to higher efficiency and dependability, and in terms of security, they are highly useful in detecting violence. CNN and 3D CNN were considered by the researchers.

They presented a CNN approach for detecting people in videos, which reduces processing time. Following that, photos of people were put into a 3D CNN model that had been trained on spatiotemporal properties, and final predictions were formed. They also want to provide a cutting-edge solution. We can prevent additional deadly accidents by identifying them with a sophisticated surveillance system. This method may be applied on a broad scale in a variety of areas such as streets, parks, and medical institutions to warn authorities of violent behaviours.

Videos, of course, are sequences of images. While most state-of-the-art image classification systems use convolutional layers in one form or another, sequential data is frequently processed by Long Short-Term Memory (LSTM) Networks. Consequently, a combination of these two building blocks is expected to perform well on a video classification task.One such combination has the self-descriptive name of ConvLSTM[6]. Standard LSTM uses simple matrix multiplication to weigh the input and previous state inside the different gates. In ConvLSTM, these operations are replaced by convolutions.

2. Literature Survey

The input video is segmented into frames in the first data preparation stage, and the backdrop is removed in the second pre-processing step. The feature extraction stage, which may be done manually or automatically, creates the behavioral structure of the data to be modeled and obtains the feature representation. The items are then recognised using CNN, and a two-class based classifier makes the final judgment. Background subtraction and optical flow are used in the detection strategy, whereas Neural Networks are used in the classification approach. For an intelligent surveillance system, he has proposed two bespoke architectures. VGGNet and AlexNet inspired custom architecture .

Fully linked layers are lowered from 4096 * 4096 * 1000 to 1024 * 1024 *7. A convolutional neural network is used for frame level feature extraction, and a convolutional long short-term memory is used for feature aggregation in the temporal domain in the proposed model.

The model makes use of three separate datasets, resulting in better outcomes.Intelligence for recognising violence in the Internet of Things (IoT) leveraging smart devices for timely replies. A Surveillance Video Anomaly Detection Technique Based on Deep Learning.

To extract features from the video frames of the input sequence, a Convolutional Neural Network (CNN) Stack is used. Then, to forecast future movements, a Convolutional Long Short-Term Memory (convLSTM) stack is applied.

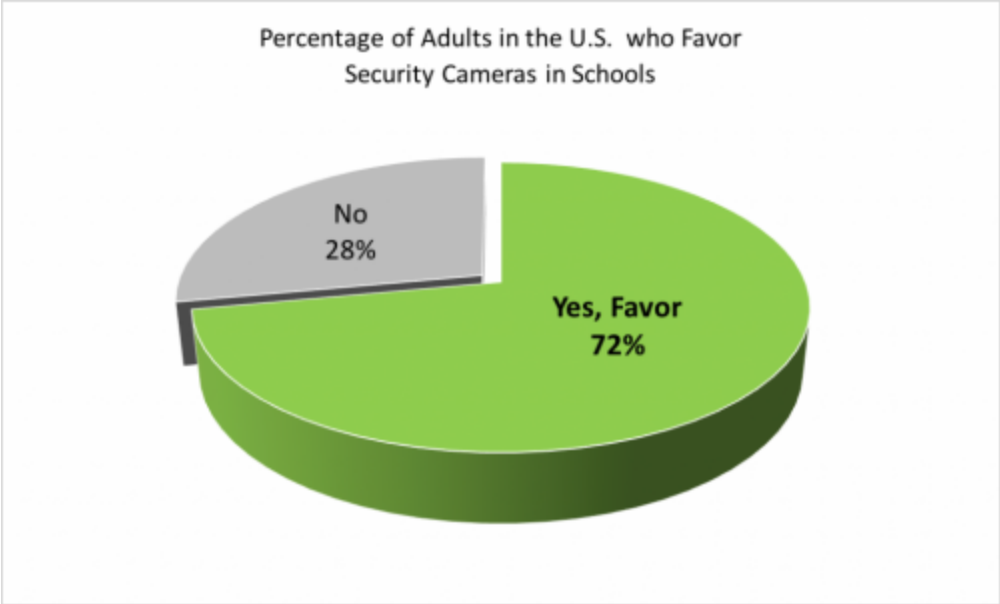
Our suggested system's goal is to create an intelligent surveillance system that can detect violence in a video frame. The intelligent surveillance system initially learns features, after which it trains on them. It identifies violence in a video and, if it finds violence in frames, sends an alarm to the appropriate authorities and stores the observed frame in a local database.

The system's input will be a video frame, and the output will be either a violent or non-violent frame in binary form. The device is designed to assist the user by determining whether or not the crime happens in a brief video clip.

This system can assist government agencies in responding more quickly. In our system, we're utilising TensorFlow GPU libraries to combine GPU and CPU to create a system that can take advantage of parallelization for quick processing.

One distinctive feature of the proposed approach is that we entirely rely on motion features, by completely disregarding appearance information. The decision is based on results from previous work on human activity recognition [46], which demonstrated that motion is far more informative than appearance information. Furthermore, to make effective use of appearance information in classification, given the often-subtle differences between the appearance of a person running or

kicking in a fight, one needs huge amounts of data and complex representation models. An additional advantage of relying exclusively on motion information is a high invariance to illumination changes. This includes safe operation in night vision scenarios, implying infrared monochromatic sensing.



Survey Methodology

studies were executed throughout the time span from January 2016 through March 2016, focusing on grown-ups all through all districts of the United States. An external study administration was utilized to run the study, with an autonomous source list.

Respondent Demographics

A delegate inspecting of the different areas of the United States: Northeast, Midwest, South and West. Respondent ages were 18 and more seasoned.

Local area Favorability toward School Security Cameras

Favorability toward Video Surveillance on School Campuses

Almost 3 of every 4 grown-ups (72%) favor surveillance camera frameworks in schools, including preschool/childcare, K-12, and school levels. This number has ascended lately, with expanded.

Year-over-Year Change in Favorability

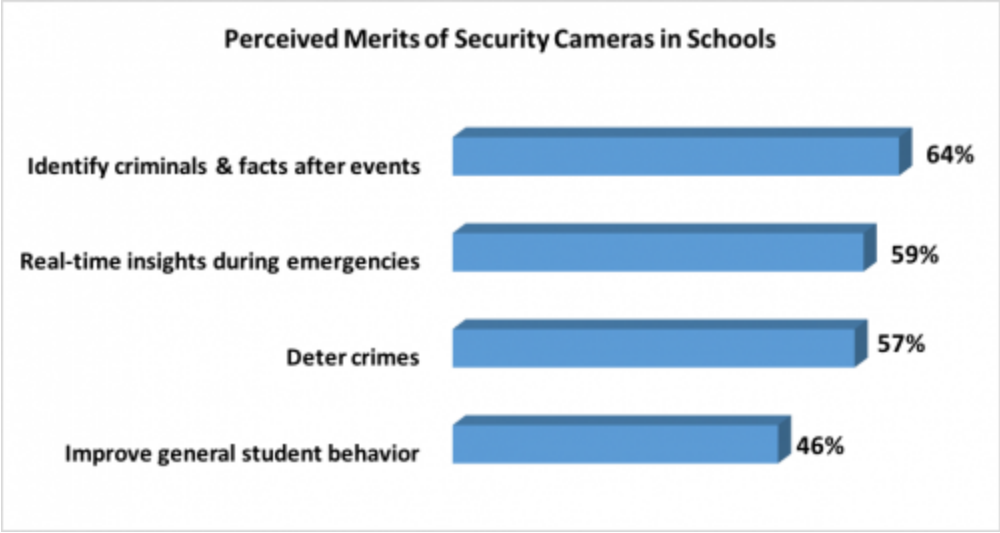
The idealness toward utilization of video observation in schools is up 7 rate focuses since the earlier year, ascending to 72 percent in mid 2016 from 65 percent in mid 2015.

All the 2016 overview information had 1500 respondents. Interestingly, the 2015 information depended on an overview run in January 2016, from similar interest group of US grown-ups, with 1000 respondents.

Perceived Merits of School Video Surveillance

The review respondents who saw a positive incentive for surveillance cameras, were approached to refer to the positive benefits of the video observation in schools. Their top positive legitimacy is to "Recognize hoodlums and realities later occasions" (64%), trailed by "continuous bits of knowledge during crises (59%) and stopping violations (57%).

Somewhat not exactly 50% of respondents considered further developing general understudy conduct to be a critical positive reason.



Expected Impact on Bullying

When asked expressly whether they anticipated that visible security cameras should diminish tormenting in schools, a greater part of the respondents (56%) accepted they would.

3. Working of Project

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. It has feature of alerting govt authorities if any suspicious activity is detected.

Database:

This database contains timestamp and screenshot of suspicious activities identified by system.

Model code :

```
import numpy as np
from skimage.transform import resize


def video_mamonreader(cv2,filename):
    frames = np.zeros((30, 160, 160, 3), dtype=np.float)
    i=0
    print(frames.shape)
    vc = cv2.VideoCapture(filename)
    if vc.isOpened():
        rval , frame = vc.read()
    else:
        rval = False
    frm = resize(frame,(160,160,3))
    frm = np.expand_dims(frm,axis=0)
    if(np.max(frm)>1):
        frm = frm/255.0
    frames[i][:] = frm
    i +=1
    print("reading video")
    while i < 30:
        rval, frame = vc.read()
        frm = resize(frame,(160,160,3))
        frm = np.expand_dims(frm,axis=0)
        if(np.max(frm)>1):
            frm = frm/255.0
        frames[i][:] = frm
        i +=1
    return frames
```

```
def mamon_videoFightModel(tf,wight='mamon-videofight100.hdf5'):
    layers = tf.contrib.keras.layers
    models = tf.contrib.keras.models
    losses = tf.contrib.keras.losses
    optimizers = tf.contrib.keras.optimizers
    metrics = tf.contrib.keras.metrics
    num_classes = 2
    input_shapes = (160,160,3)
    vg19 = tf.contrib.keras.applications.vgg19.VGG19
    base_model = vg19(include_top=False,weights=None,input_shape=(100,100,3))
    for layer in base_model.layers:
        layer.trainable = False
    model = models.Sequential()
    num_classes = 2
    cnn = models.Sequential()
    cnn.add(base_model)
    cnn.add(layers.Flatten())
    model = models.Sequential()
    model.add(layers.TimeDistributed(cnn,  input_shape=(40, 100, 100, 3)))
    model.add(layers.LSTM(40))
    model.add(layers.Dense(13, activation='relu'))
    model.add(layers.Dropout(0.1))
    model.add(layers.Dense(num_classes, activation="sigmoid"))
    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
    model.load_weights(wight)
                    model.compile(loss='binary_crossentropy',      optimizer=      adam,
metrics=["accuracy"])
    return model




def mamon_videoFightModel2(tf,wight='mamonbest947oscombo.hdfs'):
    layers = tf.contrib.keras.layers
    models = tf.contrib.keras.models
    losses = tf.contrib.keras.losses
    optimizers = tf.contrib.keras.optimizers
    metrics = tf.contrib.keras.metrics
```

```
    num_classes = 2
    cnn = models.Sequential()
    #cnn.add(base_model)

    input_shapes=(160,160,3)
    np.random.seed(1234)
    vg19 = tf.keras.applications.vgg19.VGG19
       base_model =  vg19(include_top=False,weights='imagenet',input_shape=(160,
160,3))
    # Freeze the layers except the last 4 layers
    #for layer in base_model.layers:
    #    layer.trainable = False

    cnn = models.Sequential()
    cnn.add(base_model)
    cnn.add(layers.Flatten())
    model = models.Sequential()

    model.add(layers.TimeDistributed(cnn,  input_shape=(30, 160, 160, 3)))
    model.add(layers.LSTM(30 , return_sequences= True))

    model.add(layers.TimeDistributed(layers.Dense(90)))
    model.add(layers.Dropout(0.1))

    model.add(layers.GlobalAveragePooling1D())

    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dropout(0.3))

    model.add(layers.Dense(num_classes, activation="sigmoid"))

    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
    model.load_weights(wight)
    rms = optimizers.RMSprop()

                        model.compile(loss='binary_crossentropy',       optimizer=adam,
metrics=["accuracy"])

    return model
```

```
def pred_fight(model,video,acuracy=0.9):
    pred_test = model.predict(video)
    if pred_test[0][1] >=acuracy:
        return True , pred_test[0][1]
    else:
        return False , pred_test[0][1]
```

Main file Source code:

```
from __future__ import absolute_import
from __future__  import division
from __future__ import print_function
import tensorflow as tf
import numpy as np
from skimage.io import imread
from skimage.transform import resize
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import os
from mamonfight22 import *
model = mamon_videoFightModel2(tf,wight='mamonedeffrbktoldmmon.hdf5')
cap = cv2.VideoCapture('hospital.mp4')
i = 0
frames = np.zeros((30, 160, 160, 3), dtype=np.float)
old = []
j = 0
while(True):
    ret, frame = cap.read()

    # describe the type of font
    # to be used.
    font = cv2.FONT_HERSHEY_SIMPLEX
    if i > 29:
        ysdatav2 = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
```

```
        ysdatav2[0][:][:] = frames
        predaction = pred_fight(model,ysdatav2,acuracy=0.96)
        if predaction[0] == True:
            cv2.putText(frame,
                'Violance Deacted  ... Violence .. violence',
                (50, 50),
                font, 3,
                (0, 255, 255),
                2,
                cv2.LINE_4)
            cv2.imshow('video', frame)
            print('Violance detacted here ...')
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
                    vio = cv2.VideoWriter("./videos/output-"+str(j)+".avi", fourcc, 10.0,
(fwidth,fheight))
                                #vio = cv2.VideoWriter("./videos/output-"+str(j)+".mp4",
cv2.VideoWriter_fourcc(*'mp4v'), 10, (300, 400))
            for frameinss in old:
                vio.write(frameinss)
            vio.release()
        i = 0
        j += 1
        frames = np.zeros((30, 160, 160, 3), dtype=np.float)
        old = []
    else:
        frm = resize(frame,(160,160,3))
        old.append(frame)
        fshape = frame.shape
        fheight = fshape[0]
        fwidth = fshape[1]
        frm = np.expand_dims(frm,axis=0)
        if(np.max(frm)>1):
            frm = frm/255.0
        frames[i][:] = frm

        i+=1

    cv2.imshow('video', frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()

cv2.destroyAllWindows()

import time
millis = int(round(time.time() * 1000))
print("started at " , millis)
vid = video_mamonreader(cv2,'dvpalfight.dvd')
millis2 = int(round(time.time() * 1000))
print("time processing " , millis2 - millis)

datav = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
datav[0][:][:] = vid
millis = int(round(time.time() * 1000))
print("started at " , millis)
print(pred_fight(model,datav,acuracy=0.6))
millis2 = int(round(time.time() * 1000))
print("time processing " , millis2 - millis)
def video_mamonreader(cv2,filename):
    frames = np.zeros((30, 160, 160, 3), dtype=np.float)
    i=0
    print(frames.shape)
    vc = cv2.VideoCapture(filename)
    if vc.isOpened():
        rval , frame = vc.read()
    else:
        rval = False
    frm = resize(frame,(160,160,3))
    frm = np.expand_dims(frm,axis=0)
    if(np.max(frm)>1):
        frm = frm/255.0
    frames[i][:] = frm
    i +=1
    print("reading video")
    while i < 30:
```

```
        rval, frame = vc.read()
        frm = resize(frame,(160,160,3))
        frm = np.expand_dims(frm,axis=0)
        if(np.max(frm)>1):
            frm = frm/255.0
        frames[i][:] = frm
        i +=1
    return frames
novid = video_mamonreader(cv2,'nofight.mp4')
nodatav = np.zeros((1, 40, 170, 170, 3), dtype=np.float)
nodatav[0][:][:] = novid
pred_fight(model,nodatav,acuracy=0.6)
ysvid2 = video_mamonreader(cv2,'hdfight.mp4')
ysdatav2 = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
ysdatav2[0][:][:] = ysvid2
import time

millis = int(round(time.time() * 1000))
print("started at " , millis)
predaction = pred_fight(model,ysdatav2,acuracy=0.9)
print(predaction)
millis2 = int(round(time.time() * 1000))
print("time processing " , millis2 - millis)

if predaction[0] == True:
    print('violence')
novid3 = video_mamonreader(cv2,'golsss.mp4')
```

```
nodatav3 = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
nodatav3[0][:][:] = novid3

millis = int(round(time.time() * 1000))
print("started at " , millis)
print(pred_fight(model,nodatav3,acuracy=0.8))
millis2 = int(round(time.time() * 1000))
print("time processing " , millis2 - millis)


novid4 = video_mamonreader(cv2,'dvpalfight.dvd')
nodatav4 = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
nodatav4[0][:][:] = novid4
millis = int(round(time.time() * 1000))
print("started at " , millis)
print(pred_fight(model,nodatav4,acuracy=0.9))
millis2 = int(round(time.time() * 1000))
print("time processing " , millis2 - millis)
```

WEB-FIGHT.py file :

```
from __future__ import absolute_import
from __future__  import division
from __future__ import print_function
import tensorflow as tf
import numpy as np
from skimage.io import imread
from skimage.transform import resize
import cv2
import numpy as np
import os
from mamonfight22 import *
from flask import Flask , request , jsonify
from PIL import Image
from io import BytesIO
import time


np.random.seed(1234)
```

```
model22 = mamon_videoFightModel2(tf)
graph = tf.get_default_graph()
model22._make_predict_function()

app = Flask("main-webapi")

@app.route('/api/fight/',methods= ['GET','POST'])
def main_fight(accuracyfight=0.91):
    res_mamon = {}
    if os.path.exists('./tmp.mp4'):
        os.remove('./tmp.mp4')
    filev = request.files['file']
    file = open("tmp.mp4", "wb")
    file.write(filev.read())
    file.close()
    vid = video_mamonreader(cv2,"tmp.mp4")
    datav = np.zeros((1, 30, 160, 160, 3), dtype=np.float)
    datav[0][:][:] = vid
    millis = int(round(time.time() * 1000))
    with graph.as_default():
        f , precent = pred_fight(model22,datav,acuracy=0.65)
    res_mamon = {'fight':f , 'precentegeoffight':str(precent)}
    millis2 = int(round(time.time() * 1000))
    res_mamon['processing_time'] =  str(millis2-millis)
    resnd = jsonify(res_mamon)
    resnd.status_code = 200
    return resnd

app.run(host='0.0.0.0',port=3091)
```

General training model :

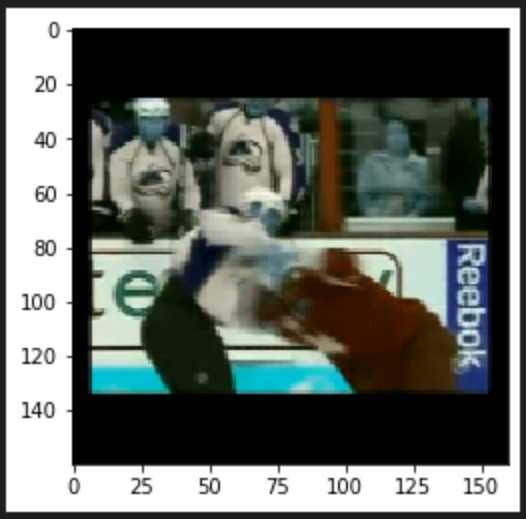
```
from __future__ import absolute_import
from __future__  import division
from __future__ import print_function
import tensorflow as tf
import numpy as np
from skimage.io import imread
from skimage.transform import resize
import cv2
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import os
fights_train = np.zeros((700, 40, 160, 160, 3), dtype=np.float)
labels_train = []
def capture(filename):
    frames = np.zeros((40, 160, 160, 3), dtype=np.float)
    i=0
    vc = cv2.VideoCapture(filename)
    if vc.isOpened():
        rval , frame = vc.read()
    else:
        rval = False
    #frm = cv2.resize(frame,(200,200))
    frm = resize(frame,(160, 160, 3))
    frm = np.expand_dims(frm,axis=0)
    if(np.max(frm)>1):
        frm = frm/255.0
    frames[i][:] = frm
    i +=1
    while i < 40:
        rval, frame = vc.read()
        #print(i)
        #plt.imshow(frame)
        #plt.show()
        #frm = cv2.resize(frame,(200,200))
        frm = resize(frame,(160, 160, 3))
        frm = np.expand_dims(frm,axis=0)
        if(np.max(frm)>1):
            frm = frm/255.0
        frames[i][:] = frm
        i +=1
        #print(frame)
    return frames

def cut_save(main_dir,mod):
    i = 0
```

```
    #fights = np.zeros((399, 40, 200, 200, 3), dtype=np.float)
    #noFights = np.zeros((599, 42, 200, 200, 3), dtype=np.float)
    for x in os.listdir(main_dir):
        if 1 == 1:
            td = main_dir+x+'/'
            #for y in os.listdir(main_dir+x+'/'):
                #print(y)
            for file in os.listdir(td):
                fl = os.path.join(td, file)
                videos = capture(fl)
                if mod == 'train':
                    fights_train[i][:][:] = videos
                    i +=1
                    if x =='fights':
                        labels_train.append(1)
                    else:
                        labels_train.append(0)
                elif mod =='test':
                    fights_test[i][:][:] = videos
                    i +=1
                    if x =='fights':
                        labels_test.append(1)
                    else:
                        labels_test.append(0)
                elif mod =='val':
                    fights_val[i][:][:] = videos
                    i +=1
                    if x =='fights':
                        labels_val.append(1)
                    else:
                        labels_val.append(0)
cut_save('./trainm/',"train")
fights_train.shape
from sklearn.model_selection import train_test_split
X_train, fights_test, y_train, labels_test = train_test_split(fights_train,labels_train,
test_size=0.33, random_state=42)
fights_train = []
fights_test= np.zeros((300, 40, 160, 160, 3), dtype=np.float)
labels_test = []
```

cut_save('./testm/',"test")
plt.imshow(fights_test[19][5])
plt.show()
OUTPUT:

```
plt.imshow(fights_test[19][5])
plt.show()
```



layers = tf.keras.layers
models = tf.keras.models
losses = tf.keras.losses
optimizers = tf.keras.optimizers
metrics = tf.keras.metrics
utils = tf.keras.utils
callbacks = tf.keras.callbacks
layers = tf.keras.layers
models = tf.keras.models
ImageDataGenerator = tf.keras.preprocessing.image.ImageDataGenerator
losses = tf.keras.losses
optimizers = tf.keras.optimizers
metrics = tf.keras.metrics
utils = tf.keras.utils
callbacks = tf.keras.callbacks

plot_model = tf.keras.utils.plot_model
np.random.seed(1234)

```
num_classes = 2
np.random.seed(1234)
num_classes = 2
vg19 = tf.keras.applications.vgg19.VGG19
base_model = vg19(include_top=False,weights='imagenet',input_shape=(160,
160,3))
# Freeze the layers except the last 4 layers
for layer in base_model.layers:
    layer.trainable = False
# Check the trainable status of the individual layers
base_model.summary()
num_classes = 2

cnn = models.Sequential()
cnn.add(base_model)
cnn.add(layers.Flatten())
#cnn.add(layers.Dense(1024, activation='relu'))
#cnn.add(layers.Dropout(0.3))
#cnn.add(layers.Dense(512, activation='relu'))
#cnn.add(layers.Dropout(0.3))
#cnn.add(layers.LSTM(40))

# define LSTM model
model = models.Sequential()

model.add(layers.TimeDistributed(cnn,  input_shape=(40, 160, 160, 3)))
model.add(layers.LSTM(40 , return_sequences=True))

#model.add(layers.Dense(num_classes, activation="sigmoid"))
#model.add(layers.Dropout(0.3))

model.add(layers.TimeDistributed(layers.Dense(160, activation='relu')))

model.add(layers.GlobalAveragePooling1D(name="globale"))

'''
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(512, activation='relu'))
```

```
model.add(layers.Dropout(0.3))
'''
model.add(layers.Dense(num_classes, activation="sigmoid" , name="last"))

adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
model.load_weights('mamon98777.hdf5')
rms = optimizers.RMSprop()
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=["accuracy"])
model.summary()
```

OUTPUT:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, 40, 12800)         20024384
_________________________________________________________________
lstm (LSTM)                  (None, 40, 40)            2054560
_________________________________________________________________
time_distributed_1 (TimeDist (None, 40, 160)           6560
_________________________________________________________________
globale (GlobalAveragePoolin (None, 160)               0
_________________________________________________________________
last (Dense)                 (None, 2)                 322
=================================================================
Total params: 22,085,826
Trainable params: 2,061,442
Non-trainable params: 20,024,384
_________________________________________________________________
```

```
class AccuracyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.acc = []
        self.val_acc = []
        self.loss = []
        self.val_loss = []

    def on_epoch_end(self, batch, logs={}):
        self.acc.append(logs.get('acc'))
        self.val_acc.append(logs.get('val_acc'))
```

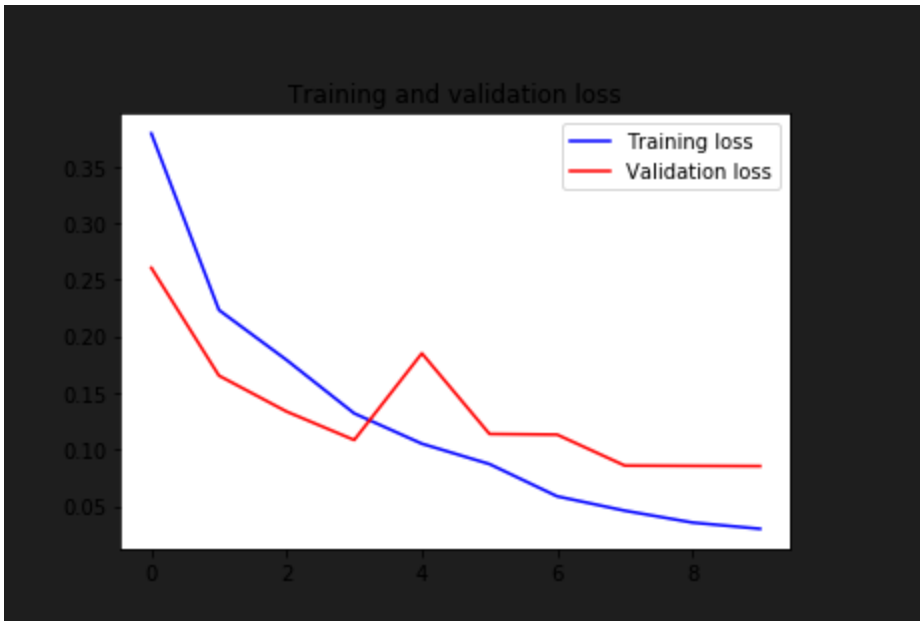
```
        self.loss.append(logs.get('loss'))
        self.val_loss.append(logs.get('val_loss'))

history = AccuracyHistory()
earlyStopping = callbacks.EarlyStopping(monitor='val_loss',
patience=8,min_delta=1e-5, verbose=0, mode='min')
mcp_save = callbacks.ModelCheckpoint('mamon98777.hdf5',
save_best_only=True, monitor='val_loss', mode='min')
reduce_lr_loss = callbacks.ReduceLROnPlateau(monitor='val_loss',patience=1,
verbose=2,factor=0.5,min_lr=0.0000001)

batch_size =3
epochs = 10
y_train = utils.to_categorical(labels_train)
print(y_train)
```

OUTPUT :

```
[[ 0.  1.]
 [ 0.  1.]
 [ 0.  1.]
 ...,
 [ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]]
```

```
y_test = utils.to_categorical(labels_test)
print(y_test)
import time
millis = int(round(time.time() * 1000))
print("started at " , millis)

model.fit(fights_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
             validation_data=(fights_test, y_test),callbacks=[earlyStopping, mcp_save,
reduce_lr_loss,history])

#0.8995 4
fights_test = []
```

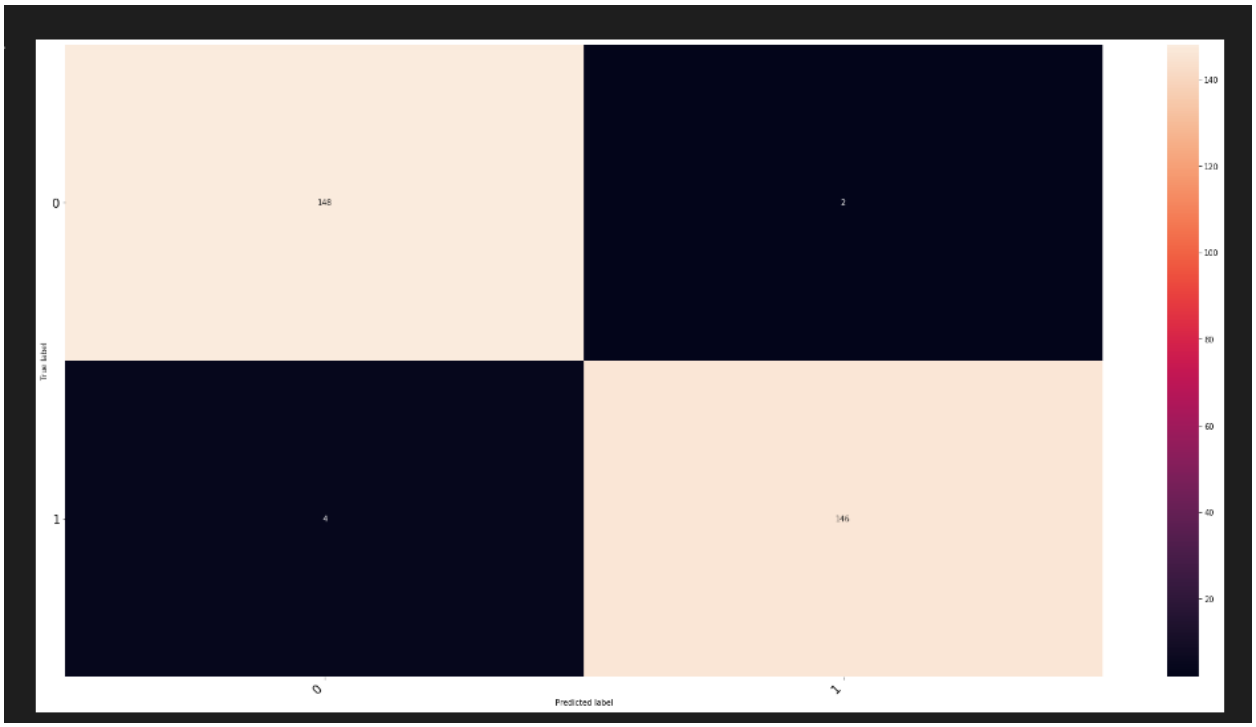
```
acc = history.acc
val_acc = history.val_acc
loss = history.loss
val_loss = history.val_loss
epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

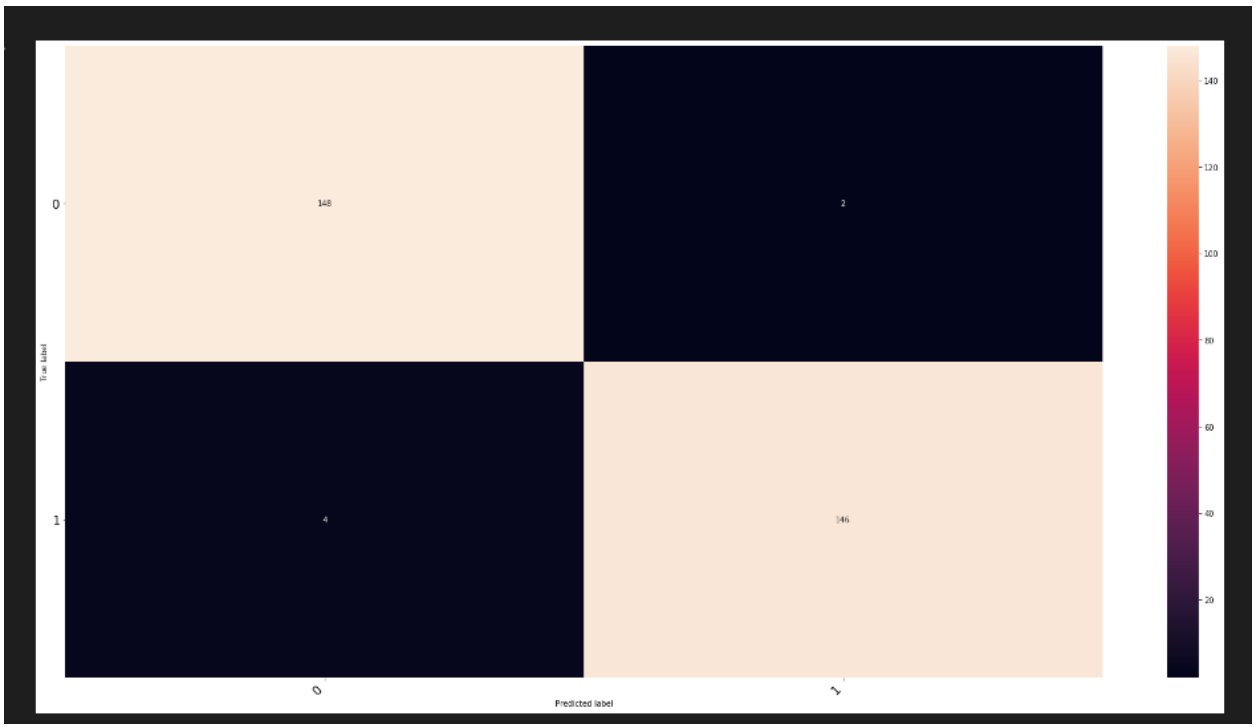



```
fights_train = []
score = model.evaluate(fights_test, y_test, batch_size=3)
score
from sklearn.metrics import classification_report, confusion_matrix

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
def print_confusion_matrix(confusion_matrix, class_names, figsize = (10,7),
fontsize=14):
```

```
    df_cm = pd.DataFrame(
        confusion_matrix, index=class_names, columns=class_names,
    )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")
            heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(),    rotation=0,
ha='right', fontsize=fontsize)
            heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(),    rotation=45,
ha='right', fontsize=fontsize)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    return fig
Y_pred = model.predict(fights_test , batch_size=1)
yprd = Y_pred > 0.5
yprd
ypredicted = []
for zero,one in yprd:
    if zero == True:
        ypredicted.append(0)
    else:
        ypredicted.append(1)
ypredicted
y_test
y = []

for zero,one in y_test:
    if zero == True:
        y.append(0)
    else:
        y.append(1)
y = []

for zero,one in y_test:
    if zero == True:
        y.append(0)
    else:
```

```
        y.append(1)
confusion = confusion_matrix(y,ypredicted)
confusion.shape
print_confusion_matrix(confusion, [0,1], figsize = (30,15), fontsize=16)
```






```
print('Classification Report')
```

print(classification_report(y, ypredicted, target_names=['no-violance','violance']))

```
    print('Classification Report')
    print(classification_report(y, ypredicted, target_names=['no-violance','violance']))
```

```
...  Classification Report
                precision    recall  f1-score   support

    no-violance       0.97      0.99      0.98       150
       violance       0.99      0.97      0.98       150

       accuracy                           0.98       300
      macro avg       0.98      0.98      0.98       300
   weighted avg       0.98      0.98      0.98       300
```

model.save("mamonbest980hocky.hdfs")

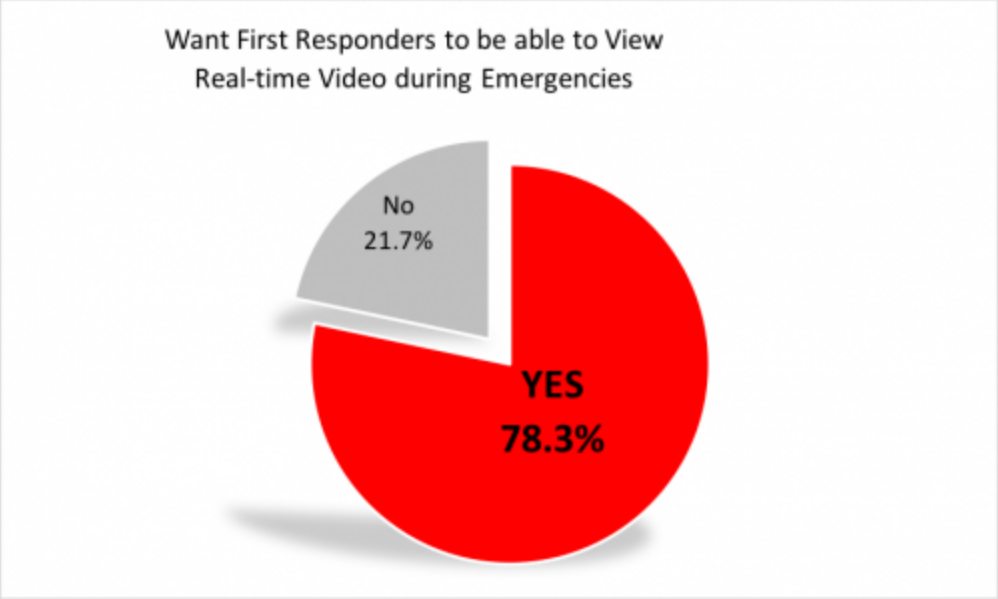
Note : the trained wights can be downloaded from this url https://drive.google.com/file/d/11IN2npH3i8PhzECNMcxfIQNFWPROr5gt/view?usp=sharing

Detection of a violence event in surveillance systems is playing a significant role in law enforcement and city safety. The effectiveness of violence event detectors measures by the speed of response and the accuracy and the generality over different kind of video sources with a different format. Several studies worked on the violence detection with focus either on speed or accuracy or both but not taking into account the generality over different kind of video sources. In this paper, we proposed a real-time violence detector based on deep-learning methods. The proposed model consists of CNN as a spatial feature extractor and LSTM as temporal relation learning method with a focus on the three-factor (overall generality - accuracy - fast response time). The suggested model achieved 98% accuracy with speed of 131 frames/sec. Comparison of the accuracy and the speed of the proposed model with previous works illustrated that the proposed model provides the highest accuracy and the fastest speed among all the previous works in the field of violence detection.

please use Tensorflow version 2.0.0 , the other dependencies is numpy skimage.io opencv PIL , BytesIO , time

First Responder Real-time Video Access

Almost 8 out of 10 grown-ups (78.3%) refer to first responders really must can get to the school video during a crisis. This outcome flags the high worth the local area places on guaranteeing quick situational mindfulness during an emergency nearby.



Location Priorities for Security Cameras

For US grown-ups who leaned toward having cameras in schools, their top school surveillance camera area needs were at passageways and ways out (76%), and corridors (62%), trailed by lounges, jungle gyms and exercise centers (53%). A lot more modest number longing having them in study halls (36%) and storage spaces and restrooms, just 18%.

Parental Access to Video

72% accepted guardians ought to have the option to see video of their kids in grades K-12, while 28 percent were against it. Most of those respondents who inclined toward parental survey of video for grades K-12 said it was proper just after an occurrence, rather than whenever.

77% trusted guardians of youngsters in preschool/childcare ought to have the option to see video of their kids. Particularly essential was the 20 rate point hop in

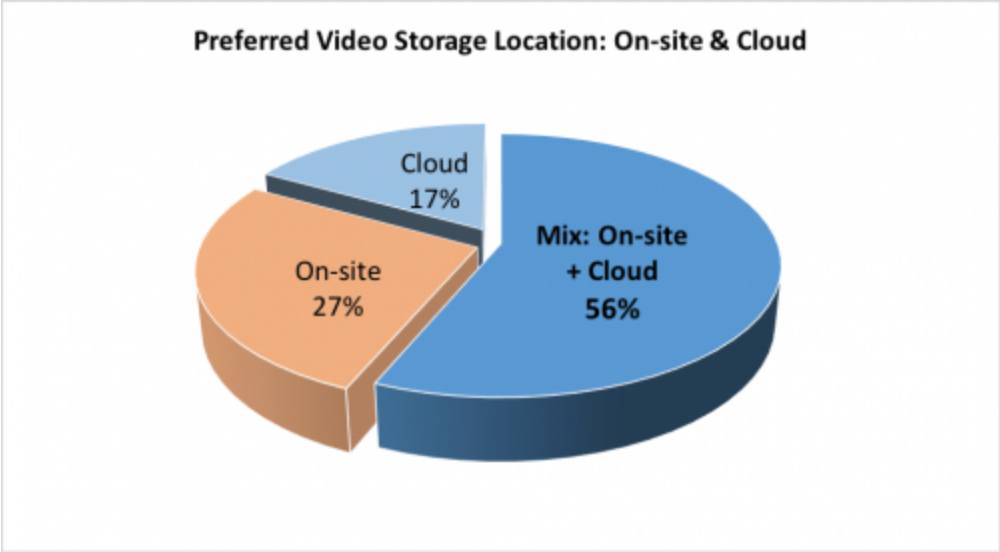
respondents who said it was satisfactory for guardians to unreservedly see video of their preschool/childcare youngsters whenever, rather than just after an occurrence. Sees here and there shifted dependent on sex. Ladies by and large accepted more in the parent's on the whole correct to see the video for grade school matured kids. 79% of ladies said guardians ought to have the option to see video of grade younger students, contrasted and just 68% of men, a 11 point distinction.

Video Storage
Desired Length of Video Storage

16% of the respondents need just live video – with no video recording. For those grown-ups who need video recorded, 80% blessing something like one month of video recording; with a normal wanted recording length of a half year.

Video Storage Preferences (on hand, Cloud, Mixed)

A larger part (56%) favor that schools keep a blend of cloud and on-premise video stockpiling.



Perceived Advantages of Mixed (Cloud + in the vicinity) Storage

For the people who see benefits for blended On-reason and Cloud video stockpiling, refer to their top reasons as: Dual capacity decreases video altering (47%); Redundant capacity stays away from loss of video (43%); and Cloud adds additional video stockpiling limit (43%). 36% say Cloud has better remote/portable video access.

School Security Camera Adoption Hurdles

The respondents refer to the top obstacle for school surveillance camera frameworks as financing impediments (32%). Security concerns were the subsequent explanation, at 23%. Other potential obstructions just got ostensible notice.

Preferred Payment Model (Up-front, Subscription)

A slight greater part favor an installment model utilizing a month to month membership, with a crossing out choice, at 54%, over forthright installment of 46%.

Real-time processing

Going back to the anomaly detector, this feature vector is used as the input to a 3-layer fully-connected neural network with Dropout. The last layer in this architecture has just one unit and computes the anomaly score through the application of the sigmoid activation function to the weighted input.
The paper reports an area under curve (AUC) value of the receiver operating characteristic (ROC) of 0.754 and a false alarm rate of 1.9 for a classification threshold of 50%. For both metrics, the presented system achieves the highest scores when compared to three other methods.
To be viable in real-world applications, anomaly detectors need to be computationally efficient and sound the alarm in time. In extreme cases, the difference between slow and fast performance can be a matter of life and death. Using an Nvidia Geforce GTX 1080 GPU as reference, the authors report that the model is capable of real-time processing with a run-time speed of 367 frames per second.

4. Result and Discussion

This section describes the experimental setup and results of the proposed framework for violence detection from video sequences. The implementation of the framework was accomplished in Python. The deep learning architecture sequential CNN and inception v4 used the Keras open-source library and tensor flow as the backend. In the experiment, the keyframe extraction technique was implemented on four video datasets. The frame rate was 5 fps, and the adjusted threshold for keyframing was 300000 for all datasets. Tab. 5 presents the results of the keyframe extraction technique; the last column presents the number of eliminated frames for each dataset, which is approximately 25% of all frames in the dataset. These results indicate that many frames are not necessary for the training of the classification model; these frames are generally not required for training, but their inclusion in the training increases the processing time. These eliminated frames save computational time, which reduces the complexity of the classification technique.

The success that deep learning models have achieved with regard to image analysis is being replicated for video content. Violence detection is one important application, but there are many others. Neural networks have been used to detect falls, a major risk for elderly people that could be addressed with smart home healthcare systems. In primatology, automated processing of videos recorded through camera traps can help scientists comprehensively study the behavior of our evolutionary relatives.

While it may be argued that technology of this kind has an inherent bias towards one side or another, the potential is there to do both good and evil. Smart surveillance can be used by those who work to ensure public safety, protect loved ones and deter criminals. Or it can be abused by those seeking to establish or expand a police state.

Automated content classification on video platforms can seen as an efficient way to uphold community standards. Or it can be regarded as a means to a censorial system with the potential to stifle challenging and unconventional expressions.

To successfully democratize artificial intelligence, we need to explore and, potentially, promote defensive uses of double-edged technologies.

The proposed method has benefited from the CNNs for feature extraction from frames. Two-way learning of bidirectional LSTMs and the attention layers that can also determine the amount of given attention to each part of the sequence are found to improve the accuracy. As a result, proposed method has surpassed the state-of-the-art performance. Additionally, a new model is tested by using Fight-CNN, a modified version of Xception model.

Bi-LSTMs show better performance than regular LSTMs in action recognition, as also stated in related studies in. Also the studies in, show that the attention layer improves the performance of sequence learning. This study validates this finding and shows that using Bi-LSTM together with attention is a promising solution to classify fight scenes.

The experimental results also indicate that the more diversity a dataset contains, the more challenging it gets to classify fight scenes. Since the collected surveillance fight dataset contains different types of fight events, from different locations, under different conditions, it poses a significant challenge for the state-of-the-art action recognition systems.

5. Future Scope

Only suspicious human behaviour and the presence of weaponry are detected by the intended system. In the future, fire detection and the detection of various weapons will be enforced.

It is possible to prepare the cloud.

6. CONCLUSIONS

This paper highlights that a convolutional neural network which leverages transfer learning with long short-term memory networks outperforms all the other variance

of convolutional neural networks. By combining CNN with LSTM, the accuracy increases to a certain margin as compared to pure transfer learning models.

The system provides a simple graphical user interface to interact with deep learning model.The main objective of this study is detecting fight scenes from surveillance cameras in a fast and accurate way. The proposed method which employs attention layer along with Bi-LSTM networks has improved the detection accuracy and provided promising results. Moreover, using a pre-trained Fight-CNN for feature extraction proves its effectiveness on surveillance camera dataset experiments.

Another important contribution of the study is the collected surveillance camera fight dataset, which presents further challenges for automatic fight detection. This surveillance camera dataset can be extended by adding new samples from security camera footages on streets or underground stations.

7. References

Barla, A., Odone, F., Verri, A.: Histogram intersection kernel for image classification. In: Proceedings of ICIP. pp. 513–516 (2003)

2. Bregler, C.: Learning and recognizing human dynamics in video sequences. In: Proceedings of Computer Vision and Pattern Recognition (1997)

3. Chen, D., Wactlar, H., Chen, M., Gao, C., Bharucha, A., Hauptmann, A.: Recognition of aggressive human behavior using binary local motion descriptors. In:
Engineering in Medicine and Biology Society. pp. 5238–5241 (20-25 2008)

4. Chen, M., Hauptmann, A.: MoSIFT: Recognizing human actions in surveillance videos. Tech. rep., Carnegie Mellon University, Pittsburgh, USA (2009)

5. Cheng, W.H., Chu, W.T., Wu, J.L.: Semantic context detection based on hierarchical audio models. In: Proceedings of the ACM SIGMM workshop on Multimedia
information retrieval. pp. 109–115 (2003)

6. Clarin, C., Dionisio, J., Echavez, M., Naval, P.C.: DOVE: Detection of movie violence using motion intensity analysis on skin and blood. Tech. rep., University of
the Philippines (2005)
7. Csurka, G., Dance, C., Fan, L.X., Willamowski, J., Bray, C.: Visual categorization
with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision (2004)
8. Efros, A.A., Berg, A.C., Mori, G., Malik, J.: Recognizing action at a distance. In:
IEEE International Conference on Computer Vision. pp. 726–733 (2003)
9. Giannakopoulos, T., Makris, A., Kosmopoulos, D., Perantonis, S., Theodoridis, S.:
Audio-visual fusion for detecting violent scenes in videos. In: 6th Hellenic Conference on AI, SETN 2010, Athens, Greece, May 4-7, 2010. Proceedings. pp. 91–100.
Springer-Verlag, London, UK (2010)
10. Giannakopoulos, T., Kosmopoulos, D., Aristidou, A., Theodoridis, S.: Violence content classification using audio features. In: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 3955, pp. 502–507 (2006)