Data Science Cluster

School of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN - 248007, Uttarakhand

Final Report

For

REAL TIME VOILENCE ALTERT SYSTEM

Mentor: Dr. Surbhi Saraswat

Year of Graduation: 2025

Prepared by

NAME	SAPID	SPECIALIZATION
Harshita Chawdhary	500093995	BIG DATA
Jaya Agarwal	500092981	BIG DATA
Ritika Baluni	500096390	BLOCKCHAIN
Rupsa Ojha	500095081	BIG DATA

Table of Content

S.No.	Title	Page
1.	Abstract	3
2.	Introduction	4
3.	Literature Review	5-6
4.	Methodology	7-12
5.	Results and discussion	13-19
6.	Conclusion	20
7.	Reference	21

1. ABSTRACT:

Hearing about the violent activities that occur daily around the world is quite overwhelming. Personal safety and social stability are seriously threatened by the violent activities. A variety of methods have been tried to curb the violent activities which includes installing of surveillance systems. It will be of great significance if the surveillance systems can automatically detect violent activities and give warning or alert signals.

The whole system can be implemented with a sequence of procedures. Firstly, the system must identify the presence of human beings in a video frame. Then, the frames which are predicted to contain violent activities must be extracted. The irrelevant frames are to be dropped at this stage. Finally, the trained model detects violent behaviour, and these frames are separately saved as images. These images are enhanced to detect faces of people involved in the activity, if possible. The enhanced images along with other necessary details such as time and location are sent as an alert to the concerned authority.

The proposed method is a deep learning based automatic detection approach that uses Convolutional Neural Network to detect violence present in a video. But, the disadvantage of using just CNN is that, it requires a lot of time for computation and is less accurate. Hence, a pre-trained model, MobileNet, which provides higher accuracy and acts as a starting point for the building of the entire model. An alert message is given to the concerned authorities using telegram application.

2. INTRODUCTION:

Violent behaviour in public places is an issue that has to be addressed. Communities are also eroded by violence, which reduces productivity, lowers property values, and disrupts social services. Across the world, violence is a severe public health issue. It affects people at various phases of life, from infants to the elderly.

Recognizing violence is challenging since it must be done on real-time videos captured by many surveillance cameras at any time and in any location. It should be able to make reliable real-time detection and alert corresponding authorities as soon as violent activities occur.

Public video surveillance systems are widespread around the world and can provide accurate and complete information in many security applications. However, having to watch videos for hours reduces your ability to make quick decisions. Video surveil lance is essential to prevent crime and violence. In this regard, several studies have been published on the automatic detection of scenes of violence in video. This is so that authorities do not have to watch videos for hours to identify events that only last a few seconds. Recent studies have highlighted the accuracy of deep learning approaches to violence detection.

Indeed, deep learning methods have proven effective for extracting spatiotemporal features from videos. A function that represents the motion information contained in a series of frames in addition to the spatial information contained in a single frame. In this work we will be discussing about the implementation of a Real-Time violence alert system using MobileNetv2. The frames obtained as output from the model is enhanced. Then these frames along with time and location of the recorded incident are send to the nearby police station as an alert via the alert module of the proposed system.

The remaining report is categorized as follows, Section II presents related studies and their detailed comparisons. Section III discusses the selected approach in a de tailed manner explaining the proposed architecture and the dataset, whereas section IV provides details of experimentation and discusses the evaluation of the approach. Finally, section V concludes the report and discusses future innovations that could be brought into our project.

3. LITERATURE REVIEW:

Ahmed et al. (2016) published "Real-time human detection and tracking for video surveillance": In this study, a system for real-time human detection and tracking in video surveillance is presented. It combines background removal with features from the histogram of oriented gradients (HOG). The system demonstrated good detection and tracking accuracy when put to the test on several video sequences.

Cong et al. (2016) published "Real-time abnormal event detection in crowded scenes": This study suggests a system that combines deep learning and trajectory analysis to detect aberrant events in real time in crowded environments. The system demonstrated great accuracy in identifying anomalous events during testing on a variety of video sequences.

Liu et al.'s (2019) "A machine learning based framework for pedestrian detection in surveillance videos": Using a combination of feature extraction and classification, this research proposes a machine learning-based system for pedestrian detection in surveillance films. The method demonstrated high accuracy in pedestrian recognition across multiple datasets.

Liu et al. (2018) published a review of "Deep learning-based object detection in video surveillance": In-depth analysis of deep learning-based object detection techniques for video surveillance is provided in this research. It discusses numerous methods and assesses how well they work using several datasets, including Faster R-CNN, YOLO, and SSD.

Deep learning for real-time multi-camera vehicle tracking, Saikia et al. (2019): The method for real-time vehicle tracking suggested in this study makes use of deep learning and several cameras. The system demonstrated high accuracy in vehicle tracking throughout testing using several datasets.

4. METHODOLOGY

4.1 Problem Statement:

The aim is to develop a real time surveillance system that can recognize violence and give alert to notify the concerned authorities.

4.2 Use Case Diagram:

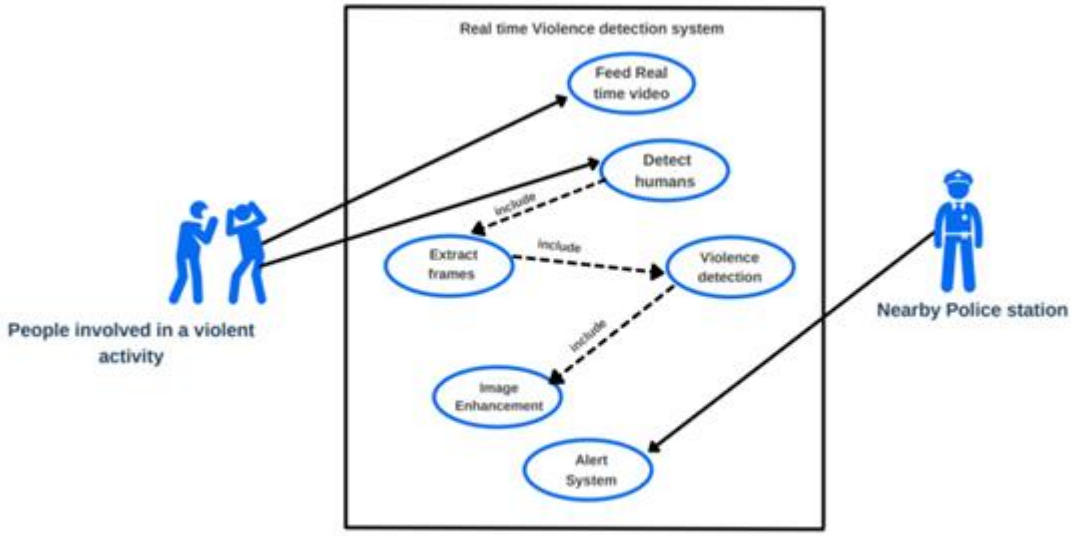


Figure 3.1: Use Case diagram

The rectangle helps to define the scope of the proposed architecture. Anything happens within the rectangle happens within the system. There are 2 actors in this scenario, People involved in a violent activity and a nearby police station. People involved in a violent activity are primary actors as their actions will be detected by a real-time system. Use cases are represented by oval shape and they represent an action that accomplishes some sort of task within the system. Whenever a real-time video is given as input, the system will try to detect humans. Every time humans are detected the given included use cases are executed as well. They are

Frames Extraction, Violence Detection and Image Enhancement. After these three steps alert system will send an alert to a nearby police station.

4.3 Architecture:

Footage from the surveillance camera is broken down into frames. The frames are given as input to MobileNet v2 classifier for detecting violent activities in the given sequence of input frames. If no violent activity is recognized the respective frames are discarded. The violence detected frame is obtained and it is enhanced for better clarity. That frame, along with the location are sent to the nearest authorities using Telegram bot.

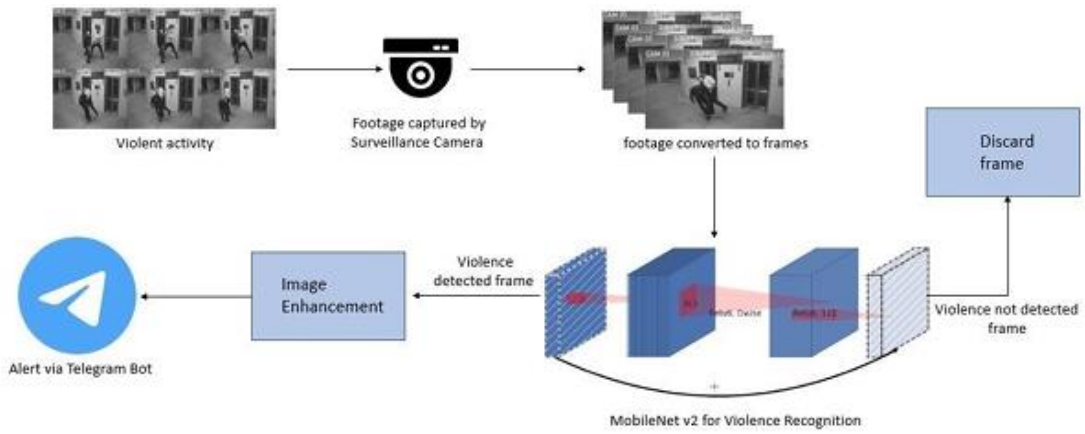


Figure 3.2: High level architecture diagram

Dataset:

The dataset contains 1000 video clips which belongs to two classes, violence and non-violence respectively. The average duration of the video clips is 5 seconds and majority of those videos are from CCTV footages. For training, 350 videos each from the violent and non-violent classes are taken at each epoch.

Figure 3.3: Video clips from the violence dataset

MobileNet V2:

The MobileNet architecture is primarily based on depth wise separable convolution, in which factors a traditional convolution into a depth wise convolution followed by a pointwise convolution. The module presents a residual cell (has a residual/identity connection) with stride of 1, and a resizing cell with a stride of 2. From Figure 3.3, "conv" is a normal con volution, "dwese" is a depth wise separable convolution, "Relu6" is a ReLu activation function with a magnitude limitation, and "Linear" is the use of the linear function. The main strategies introduced in MobileNetV2 were linear bottleneck and inverted residual blocks. In the linear bottleneck layer, the channel dimension of input is ex panded to reduce the risk of information loss by nonlinear functions such as ReLU. It stems from the fact that information lost in some channels might be preserved in other channels. The inverted residual block has a ("narrow"-"wide"-""narrow") structure in the channel dimension whereas a conventional residual block has a ("wide"- "narrow" "wide") one. Since skip connections are between narrow layers instead of wider ones, the memory footprint can be reduced.

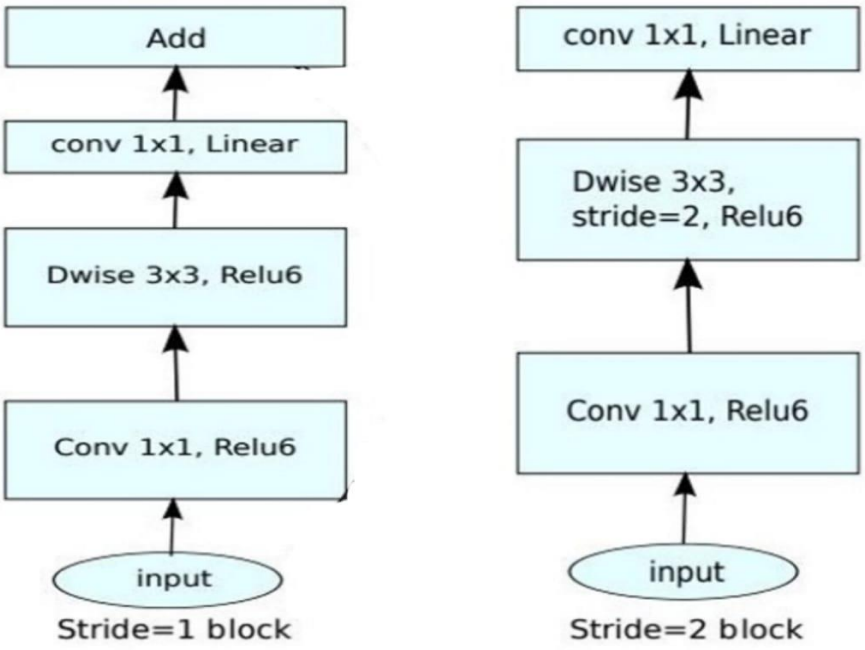


Figure 3.4: MobileNet v2 Architecture

Alert Module:

The alert module sends alert message to the specified authority. Figure 3.4 describes the architecture of the implemented alert system. When a frame is detected true for violence, the system initialises a counter variable to one. Then it checks the subsequent 30 frames, whether if they too have violence detected true. The counter is incremented at each consecutive frame that is true for violence. If a frame is false for violence, the counter variable is set to 0 and starts checking the consecutive frame respectively checking whether violence is recognized. On the other hand, if the violence is detected true for the 30 consecutive frames, the current time is obtained using an inbuilt python function and an alert is sent to a Telegram group that consists officials of higher authorities. The Alert message comprises of an image of the detected violent activity, current timestamp and the location where the camera is placed.

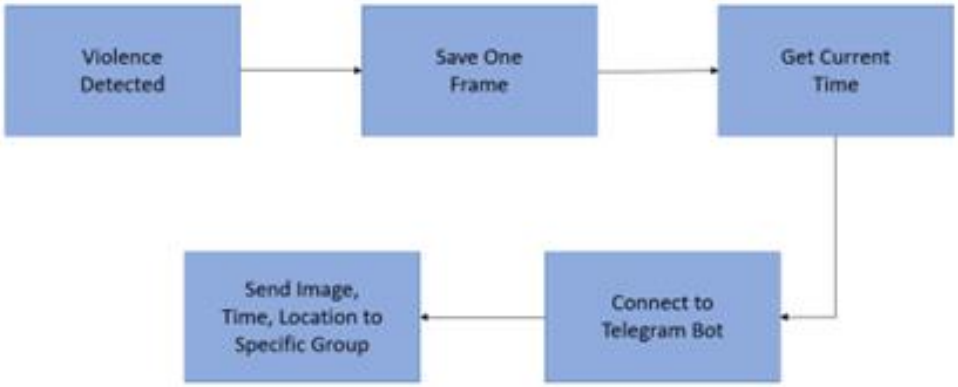
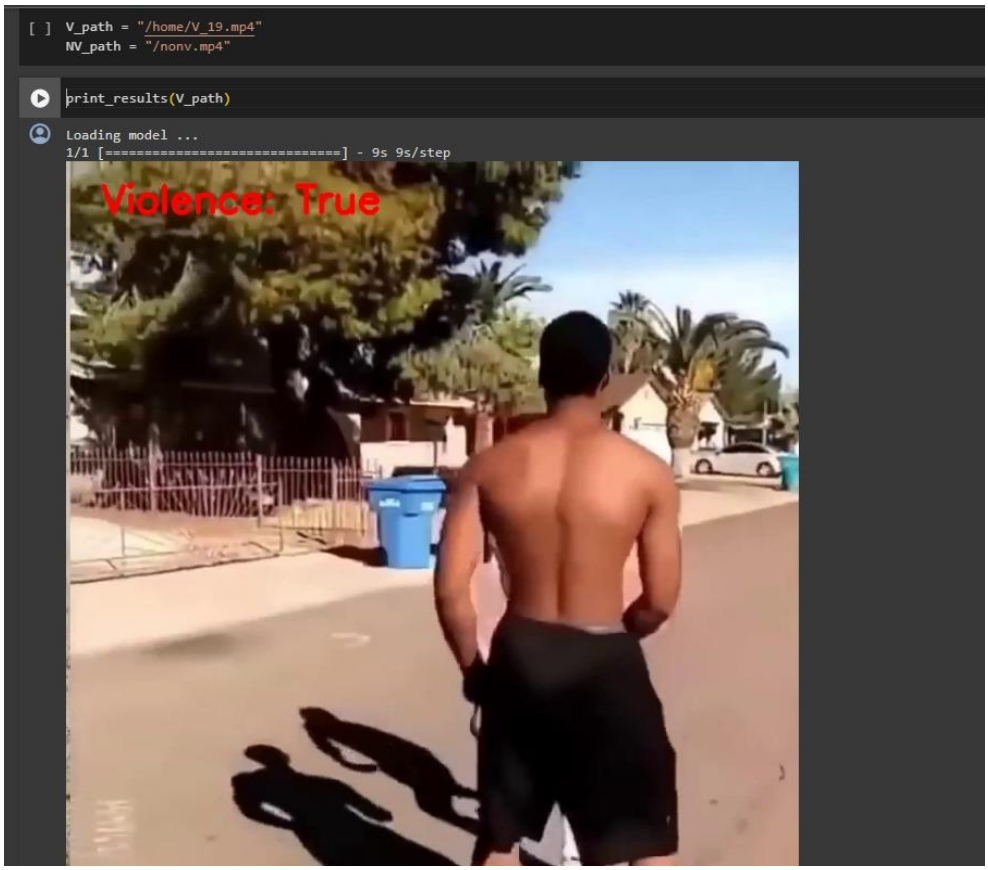


Figure 3.5: Architecture diagram of the Alert System

Figure 3.5 shows the alert message that is sent to the telegram group by the telegram bot. The concerned authorities can view the alert and take necessary actions.



4.3.1 Operating Environment Python:

The language used here is a popular programming language called Python which due to its huge number of pre-built libraries, is quite suitable for doing Machine Learning and Deep Learning projects. Python has libraries like Tensor flow and Open-CV which we have used in our project. Due to its easy to learn syntax, it is used world-wide for different purposes like web development etc. 12 Google Collaboratory- It is an environment, or a web application developed by Google for helping people do python related projects like Machine Learning and Deep Learning based. To help us with the tasks that require extreme amount of computational power, Google allows us to use their Graphical Processing Unit or their Tensor Processing Unit for free if we ever need them. This project does need a decent amount of computational power, hence is used here.

5. RESULTS AND DISCUSSION

In this section testing and training accuracy are displayed in the below given graphical representation. Fig. 4.1 displays the training and testing accuracy and loss for the MobileNet v2 model when a dataset containing 1000 videos of average duration 7 seconds is given as input. For each epoch 350 videos from the violence class and 350 videos from the non-violence are trained. 96% accuracy was obtained on training and a respective accuracy of 95% was obtained when a CCTV footage that was not included in the dataset was given for testing. The obtained output video frames are shown in Figure 4.3 In the graph in Figure 4.1 the accuracy and loss come to a constant level of increment and decrement after approximately 5 epochs. The obtained confusion matrix and other evaluation parameters are shown in Fig. 4.1 A video with violence is given as input to the system. Figure 4.3 shows one frame in the video that was labelled to have violent activity. Another video clip without violent activity was given as input. Figure 4.4 shows one frame of that video which is rightly labelled as false or violence.

5.1 Preprocessing:

```
[ ]: import os
import platform
from IPython.display import clear_output
print(platform.platform())

def resolve_dir(Dir):
    if not os.path.exists(Dir):
        os.mkdir(Dir)

def reset_path(Dir):
    if not os.path.exists(Dir):
        os.mkdir(Dir)
    else:
        os.system('rm -f {}/*'.format( Dir))
```

```
[5]: import tensorflow as tf
tf.random.set_seed(73)
TPU_INIT = False

if TPU_INIT:
    try:
        tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect()
        tpu_strategy = tf.distribute.experimental.TPUStrategy(tpu)

    except ValueError:
        raise BaseException('ERROR: Not connected to a TPU runtime!')
else:
    !nvidia-smi
;
print("Tensorflow version " + tf.__version__)
```

```
WARNING:tensorflow:From C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softma
x_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

Thu Apr 25 17:08:01 2024       Tensorflow version 2.15.0

+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 546.26                 Driver Version: 546.26       CUDA Version: 12.3     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM  | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce GTX 1650      WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   48C    P8              3W /  30W |    112MiB /  4096MiB |     20%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
```

Preprocessing

- **Getting frames form video**
- **some image argumentations**

[22]:
```
import cv2
import os
import imageio
import imgaug.augmenters as iaa
import imgaug as ia

IMG_SIZE = 128
ColorChannels = 3

def video_to_frames(video):
    vidcap = cv2.VideoCapture(video)

    import math
    rate = math.floor(vidcap.get(3))
    count = 0

    ImageFrames = []
    while vidcap.isOpened():
        ID = vidcap.get(1)
        success, image = vidcap.read()

        if success:
            # skipping frames to avoid duplications
            if (ID % 7 == 0):
                flip = iaa.Fliplr(1.0)
                zoom = iaa.Affine(scale=1.3)
                random_brightness = iaa.Multiply((1, 1.3))
                rotate = iaa.Affine(rotate=(-25, 25))

                image_aug = flip(image = image)
                image_aug = random_brightness(image = image_aug)
                image_aug = zoom(image = image_aug)
                image_aug = rotate(image = image_aug)

                rgb_img = cv2.cvtColor(image_aug, cv2.COLOR_BGR2RGB)
                resized = cv2.resize(rgb_img, (IMG_SIZE, IMG_SIZE))
                ImageFrames.append(resized)

            count += 1
        else:
            break
```

[8]:
```
%%time
from tqdm import tqdm

VideoDataDir = PROJECT_DIR
print('we have \n{} Violence videos \n{} NonViolence videos'.format(
              len(os.listdir(VideoDataDir + '/Violence')),
              len(os.listdir(VideoDataDir + '/NonViolence'))))

X_original = []
y_original = []

print('i choose 700 videos out of 2000, cuz of memory issue')
CLASSES = ["NonViolence", "Violence"]
#700 <- 350 + 350

for category in os.listdir(VideoDataDir):
    path = os.path.join(VideoDataDir, category)
    class_num = CLASSES.index(category)
    for i, video in enumerate(tqdm(os.listdir(path)[0:350])):
        frames = video_to_frames(path + '/' + video)
        for j, frame in enumerate(frames):
            X_original.append(frame)
            y_original.append(class_num)
```

```
we have
1000 Violence videos
1000 NonViolence videos
i choose 700 videos out of 2000, cuz of memory issue
100%|██████████| 350/350 [01:14<00:00,  4.71it/s]
100%|██████████| 350/350 [02:07<00:00,  2.74it/s]
CPU times: total: 11min 11s
Wall time: 3min 22s
```

[40]:
```
import numpy as np
X_original = np.array(X_original).reshape(-1 , IMG_SIZE * IMG_SIZE * 3)
y_original = np.array(y_original)
len(X_original)
```

[40]: 13979

5.2 Model Training:

```
from keras.layers import Input, Dropout, Flatten, Dense
from keras.models import Model  # Import Model class

import matplotlib.pyplot as plt
```

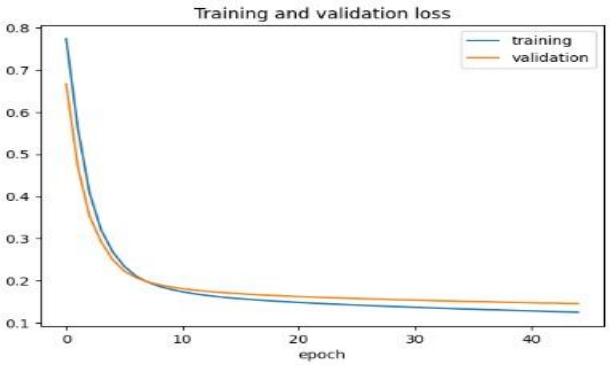
[45]:
```
epochs = 50

from keras import regularizers
kernel_regularizer = regularizers.l2(0.0001)

from keras.applications.mobilenet_v2 import MobileNetV2

def load_layers():
  """
  This function defines the model architecture.
  """
  # Define the input layer
  input_tensor = Input(shape=(IMG_SIZE, IMG_SIZE, ColorChannels))

  # Load the MobileNetV2 base model, excluding the top classification layer
  baseModel = MobileNetV2(pooling='avg', include_top=False, input_tensor=input_tensor)

  # Extract the output of the base model for further processing
  headModel = baseModel.output

  # Add a Dense layer with 1 unit and sigmoid activation for binary classification
  headModel = Dense(1, activation="sigmoid")(headModel)

  # Create the final model by specifying inputs and outputs
  model = Model(inputs=baseModel.input, outputs=headModel)

  # Freeze the base model layers (optional, can be adjusted for fine-tuning)
  for layer in baseModel.layers:
    layer.trainable = False

  # Now, assign the created model to the 'model' variable before returning
  return model  # Return the created model

if TPU_INIT:
  with tpu_strategy.scope():
    model = load_layers()
    model.summary()
else:
  model = load_layers()
  model.summary()
```

```
hNormalization)

block_4_depthwise_relu (Re  (None, 16, 16, 192)         0         ['block_4_depthwise_BN[0][0]']
LU)

block_4_project (Conv2D)    (None, 16, 16, 32)          6144      ['block_4_depthwise_relu[0][0]
                                                                  ']

block_4_project_BN (BatchN  (None, 16, 16, 32)          128       ['block_4_project[0][0]']
ormalization)

block_4_add (Add)           (None, 16, 16, 32)          0         ['block_3_project_BN[0][0]',
                                                                   'block_4_project_BN[0][0]']

block_5_expand (Conv2D)     (None, 16, 16, 192)         6144      ['block_4_add[0][0]']

block_5_expand_BN (BatchNo  (None, 16, 16, 192)         768       ['block_5_expand[0][0]']
rmalization)
```

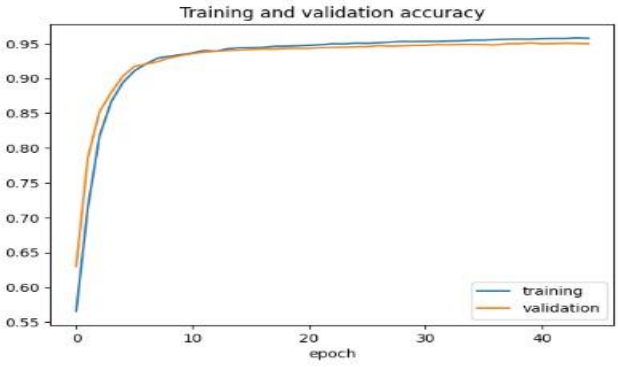
5.3 Evaluation:

```
Epoch 22/50
[33]: %matplotlib inline
def print_graph(item, index, history):
    plt.figure()
    train_values = history.history[item][0:index]
    plt.plot(train_values)
    test_values = history.history['val_' + item][0:index]
    plt.plot(test_values)
    plt.title('Training and validation ' + item)
    plt.legend(['training','validation'])
    plt.show()
    plot = '{}.png'.format(item)
    plt.savefig(plot)

def get_best_epoch(test_loss, history):
    for key, item in enumerate(history.history.items()):
        (name, arr) = item
        if name == 'val_loss':
            for i in range(len(arr)):
                if round(test_loss, 2) == round(arr[i], 2):
                    return i

def model_summary(model, history):
    print('---'*30)
    test_loss, test_accuracy = model.evaluate(X_test_nn, y_test, verbose=0)
    if history:
        index = get_best_epoch(test_loss, history)
        print('Best Epochs: ', index)
        train_accuracy = history.history['accuracy'][index]
        train_loss = history.history['loss'][index]
        print('Accuracy on train:',train_accuracy,'\tLoss on train:',train_loss)
        print_graph('loss', index, history)
        print_graph('accuracy', index, history)
        print('---'*30)

[81]: model_summary(model, history)
```

```
[61]: model_summary(model, history)
------------------------------------------------------------------------------------------
Best Epochs:  45
Accuracy on train: 0.9588145017623901   Loss on train: 0.12433040142059326
Accuracy on test: 0.9513590931892395    Loss on test: 0.14294649660587311
```




```
<Figure size 640x480 with 0 Axes>
```




```
------------------------------------------------------------------------------------------
<Figure size 640x480 with 0 Axes>
```

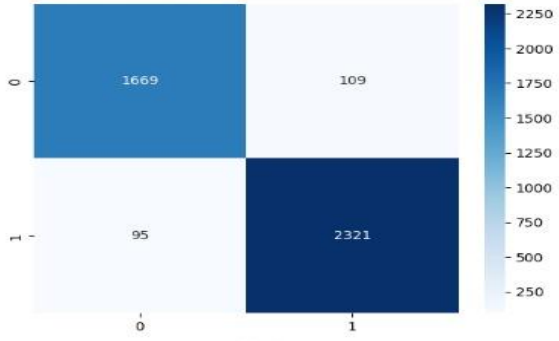
Evaluation on test set

[63]:
```
# evaluate the network
print("Evaluating network...")
predictions = model.predict(X_test_nn)
preds = predictions > 0.5
```

```
Evaluating network...
132/132 [==============================] - 12s 91ms/step
```

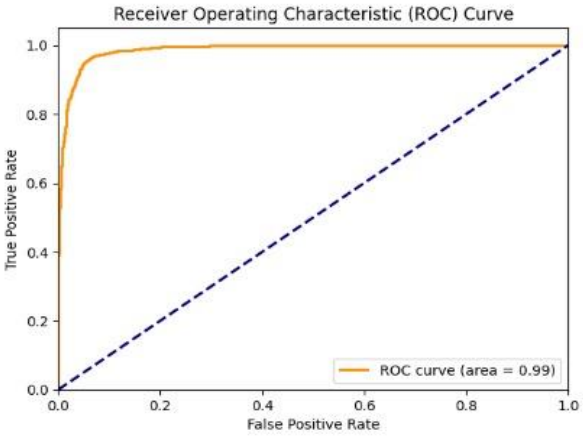
[69]:
```
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

corr_pred = confusion_matrix(y_test, preds)

n_correct = int(corr_pred[0][0] + corr_pred[1][1])  # Use int directly
print('> Correct Predictions:', n_correct)
n_wrongs = int(corr_pred[0][1] + corr_pred[1][0])  # Use int directly
print('> Wrong Predictions:', n_wrongs)

sns.heatmap(corr_pred, annot=True, fmt="d", cmap="Blues")
plt.show()

print(metrics.classification_report(y_test, preds,
                                    target_names=["NonViolence", "Violence"]))
```

```
> Correct Predictions: 3990
> Wrong Predictions: 204
```

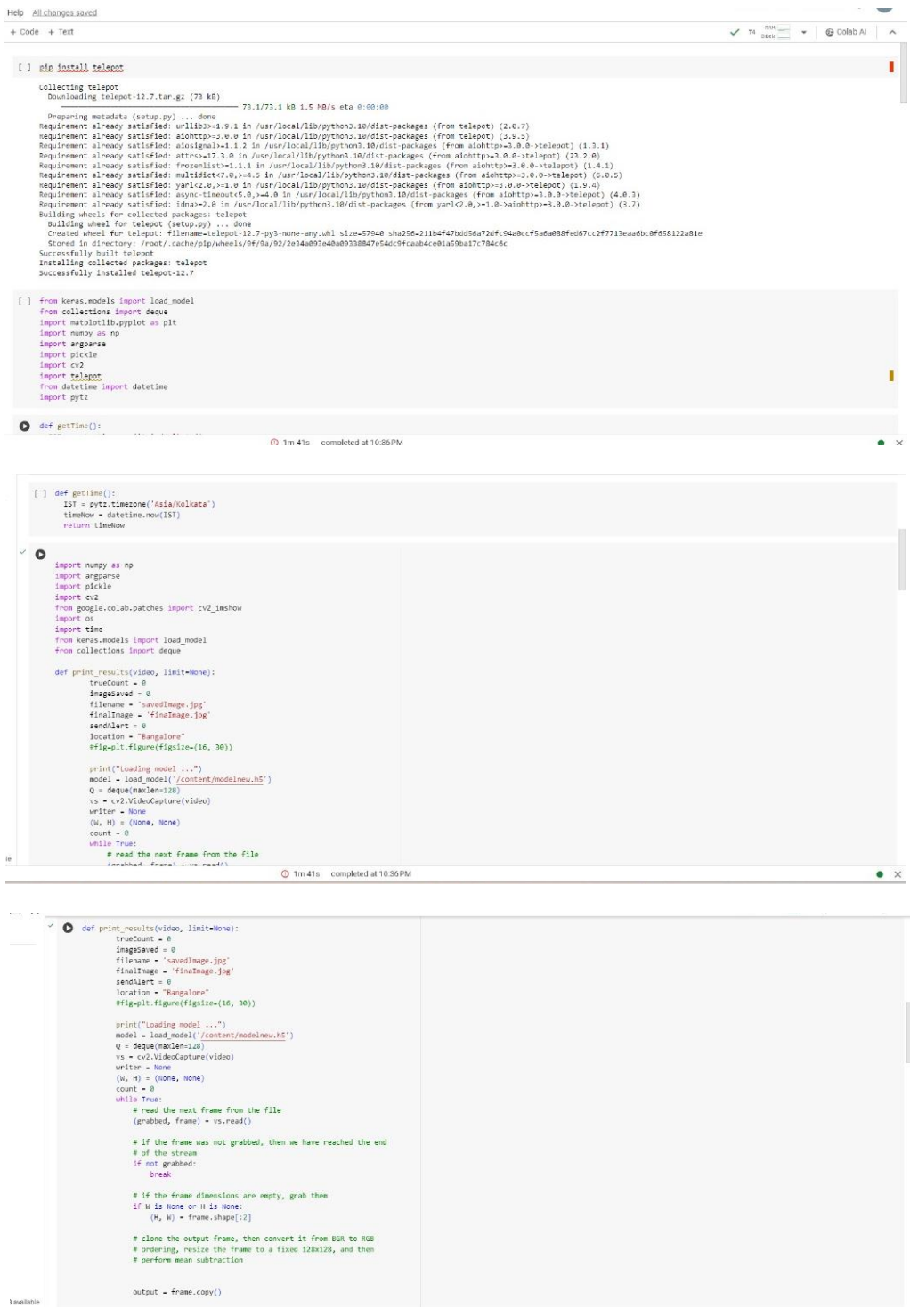


	precision	recall	f1-score	support
NonViolence	0.95	0.94	0.94	1778
Violence	0.96	0.96	0.96	2416
accuracy			0.95	4194
macro avg	0.95	0.95	0.95	4194
weighted avg	0.95	0.95	0.95	4194

[68]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
# Compute ROC curve and ROC area for each class
fpr, tpr, _ = roc_curve(y_test, predictions)
roc_auc = roc_auc_score(y_test, predictions)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



5.4 Violence and Non-Violence Detection:



```
pip install telepot

Collecting telepot
  Downloading telepot-12.7.tar.gz (73 kB)
     ---------------------------------------- 73.1/73.1 kB 1.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: urllib3>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from telepot) (2.0.7)
Requirement already satisfied: aiohttp>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from telepot) (3.9.5)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp>=3.0.0->telepot) (4.0.3)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.10/dist-packages (from yarl<2.0,>=1.0->aiohttp>=3.0.0->telepot) (3.7)
Building wheels for collected packages: telepot
  Building wheel for telepot (setup.py) ... done
  Created wheel for telepot: filename=telepot-12.7-py3-none-any.whl size=57040 sha256=211b4f47bdd56a72dfc94a0ccf5a6a088fed67cc2f7713eaa6bc0f658122a81e
  Stored in directory: /root/.cache/pip/wheels/0f/9a/92/2e34a093e40a09338847e54dc9fcaab4ce01a59ba17c784c6c
Successfully built telepot
Installing collected packages: telepot
Successfully installed telepot-12.7

from keras.models import load_model
from collections import deque
import matplotlib.pyplot as plt
import numpy as np
import argparse
import pickle
import cv2
import telepot
from datetime import datetime
import pytz

def getTime():
```

1m 41s completed at 10:36 PM

```
def getTime():
    IST = pytz.timezone('Asia/Kolkata')
    timeNow = datetime.now(IST)
    return timeNow

import numpy as np
import argparse
import pickle
import cv2
from google.colab.patches import cv2_imshow
import os
import time
from keras.models import load_model
from collections import deque

def print_results(video, limit=None):
        trueCount = 0
        imageSaved = 0
        filename = 'savedImage.jpg'
        finalImage = 'finaImage.jpg'
        sendAlert = 0
        location = "Bangalore"
        #fig=plt.figure(figsize=(16, 30))

        print("Loading model ...")
        model = load_model('/content/modelnew.h5')
        Q = deque(maxlen=128)
        vs = cv2.VideoCapture(video)
        writer = None
        (W, H) = (None, None)
        count = 0
        while True:
            # read the next frame from the file
```

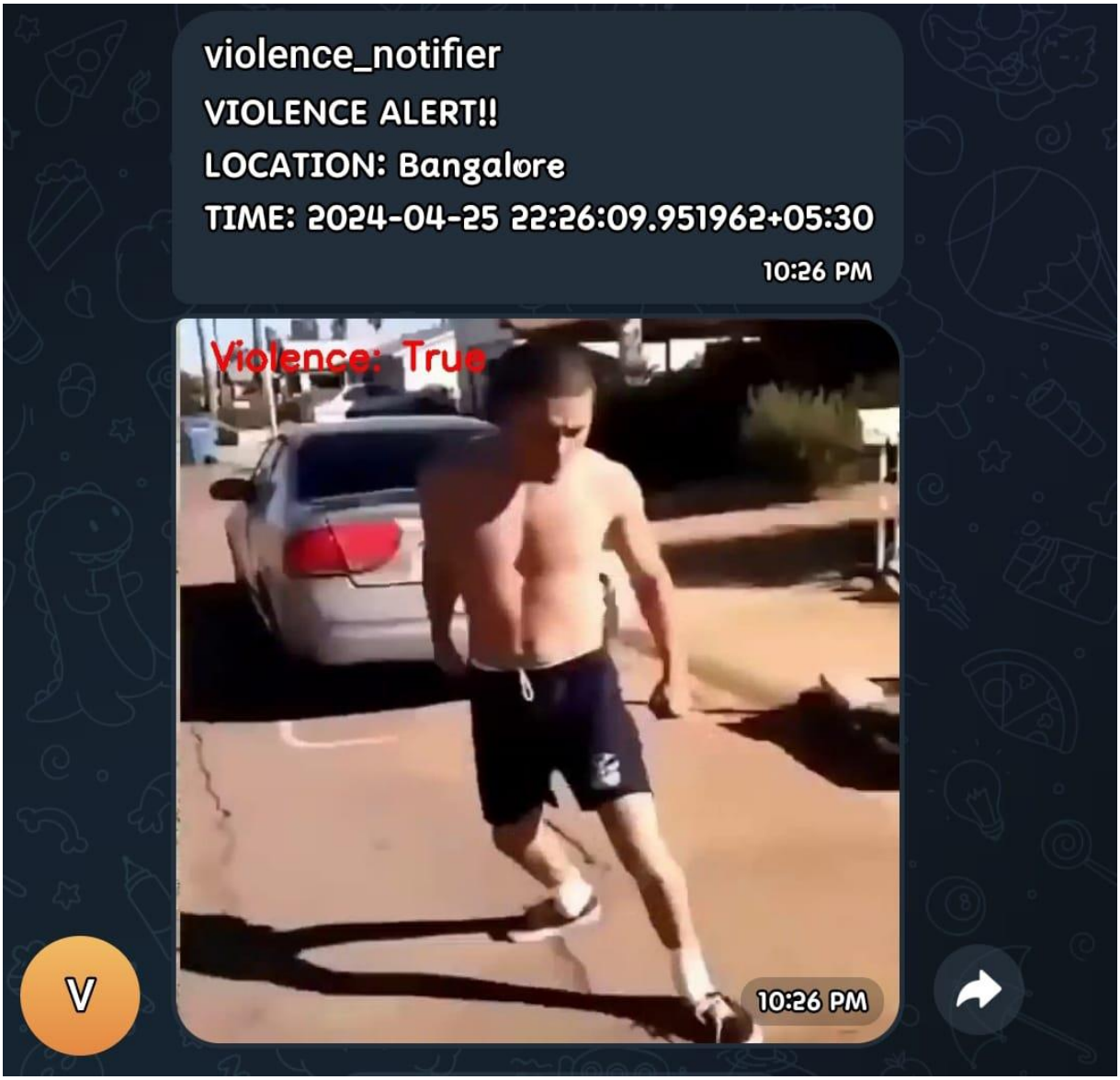
1m 41s completed at 10:36 PM

```
def print_results(video, limit=None):
        trueCount = 0
        imageSaved = 0
        filename = 'savedImage.jpg'
        finalImage = 'finaImage.jpg'
        sendAlert = 0
        location = "Bangalore"
        #fig=plt.figure(figsize=(16, 30))

        print("Loading model ...")
        model = load_model('/content/modelnew.h5')
        Q = deque(maxlen=128)
        vs = cv2.VideoCapture(video)
        writer = None
        (W, H) = (None, None)
        count = 0
        while True:
            # read the next frame from the file
            (grabbed, frame) = vs.read()

            # if the frame was not grabbed, then we have reached the end
            # of the stream
            if not grabbed:
                break

            # if the frame dimensions are empty, grab them
            if W is None or H is None:
                (H, W) = frame.shape[:2]

            # clone the output frame, then convert it from BGR to RGB
            # ordering, resize the frame to a fixed 128x128, and then
            # perform mean subtraction


            output = frame.copy()
```


print_results(V_path)
Loading model ...
1/1 [==============================] - 3s 3s/step
Violence: True
1m 41s completed at 10:36PM


print_results(NV_path)
Loading model ...
Violence: False
2012-11-09 18:35:48
IPC
2012-11-09 18:35:48

5.5 Alert System (Telegram Bot):



6. CONCLUSION

Violence scene detection in real-time is a challenging problem due to the diverse content and large variations quality. In this research, we use the MobileNet v2 model to offer an innovative and efficient technique for identifying violent events in real-time surveillance footage. The proposed network has a good recognition accuracy in typical benchmark datasets, indicating that it can learn discriminative motion saliency maps successfully. It's also computationally efficient, making it ideal for use in time-critical applications and low-end devices. Here, we had also shown the working of an alert system that is integrated with the pretrained model. In comparison to other state-of the-art approaches, this methodology will give a far superior option. Future Scope: This model could be upgraded to work in multiple cameras connected by a single network in a concurrent fashion. A short video of the violent activity could be incorporated along with the alert message.

7. REFERENCES

[1] Ahmed, N., Nait-ali, A., & Omar, M. (2016). Real-time human detection and tracking for video surveillance. 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 98-103. doi: 10.1109/avss.2016.7738023

[2] Cong, Y., Yuan, J., Liu, J., & Thalmann, D. (2016). Real-time abnormal event detection in crowded scenes. ACM Transactions on Multimedia Computing, Communications, and Applications, 12(4), 1-21. doi: 10.1145/2973641

[3] Liu, H., Huang, Y., Liu, F., & Zeng, G. (2019). A machine learning based framework for pedestrian detection in surveillance videos. IEEE Access, 7, 34845-34856. doi: 10.1109/access.2019.2901573

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (2018). Deep learning-based object detection in video surveillance: a review. Journal of Visual Communication and Image Representation, 55, 273-280. doi: 10.1016/j.jvcir.2018.09.009 Saikia, R., Ahmed, S. I., & Das, D. (2019).

[5] Real-time multi-camera vehicle tracking using deep learning. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-6. doi: 10.1109/icccnt45670.2019.8945149 Communication and Image Representation, 55, 273-280. doi: 10.1016/j.jvcir.2018.09.009

[6]https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification8febb490e61c