जामिया मिल्लिया इस्लामिया جامعہ ملیہ اسلامیہ
Jamia Millia Islamia

A

PROJECT REPORT

ON

"CLOUD ENABLED VIOLENCE DETECTION SYSTEM"

Submitted by

MOTASIM	(23MCS014)
JUNED KHAN	(23MCS010)

Under the supervision of

Prof. Mohd Amjad,

&

Dr. Danish Raza Rizvi

Master Of Technology

In

Computer Engineering

Faculty of Engineering & Technology

Jamia Millia Islamia

New Delhi

April' 2024

TABLE OF CONTENTS

1. **OBJECTIVE**:

This project aims to develop a Cloud-Enabled Multimedia Evidence Capture System using Raspberry Pi cameras that can automatically capture pictures or videos at accident scenes and securely store them on the cloud. Implement cloud storage and processing capabilities to securely store captured images or videos, ensuring data integrity and accessibility from anywhere. Then we use some important machine learning algorithms to analyze captured data, such as identifying types of accidents, assessing severity, and recognizing patterns for predictive analysis. capable of handling a high volume of image or video data from multiple sources concurrently without performance degradation. The devices used to capture the accident such as Raspberry Pi, cameras, to enhance the system's capabilities and data accuracy.

2. INTRODUCTION:

Accident management is a critical aspect of public safety and infrastructure maintenance. Traditional methods of documenting accidents, such as written reports and photographs, have limitations in terms of accuracy, efficiency, and accessibility. To address these challenges, a cloud-enabled image and video capture system can significantly enhance accident management processes. This system incorporates cutting-edge technologies like Internet of Things (IoT) devices, cloud computing, and artificial intelligence (AI) to streamline accident documentation, analysis, and reporting.

The system employs IoT devices such as cameras, and sensors installed in vehicles and accident-prone areas or the public places where the chances of suspicious activities are more. These devices continuously capture high-resolution images and video-clips, along with relevant data like location, and time can also be integrated. Cloud-Based Storage and Processing All captured media and data are securely stored in the cloud, ensuring centralized access and data integrity. Cloud computing enables scalable storage solutions, reducing the need for physical storage infrastructure and facilitating seamless data retrieval for computation and the prediction tasks.

3. LITERATURE REVIEW:

Violence detection systems have become increasingly important in various domains such as security, public safety, and surveillance in order to track the real time situation. Traditional methods relying on human surveillance and manual monitoring suffer from limitations in accuracy and real-time response. As a result, there is a growing interest in automated systems leveraging advanced technologies like computer vision, machine learning and neural networks.

Cloud computing has emerged as a powerful platform for surveillance systems, offering scalability, real-time processing, and remote monitoring capabilities which can store large amount of data that can be further use as a dataset. Several studies have explored the integration of cloud computing in surveillance applications, demonstrating its effectiveness in handling large volumes of data and supporting real-time analytics.

Raspberry Pi, with its low cost and versatility, has gained popularity as a platform for embedded systems. Researchers have utilized Raspberry Pi for various applications, including surveillance and security. Its capabilities for image processing, data acquisition and the portable nature by which it can be integrated in a bigger hardware makes it well-suited for implementing violence detection systems.

Camera modules play a crucial role in surveillance systems, providing visual data for analysis. The data captured by the Camera module can also be used as a dataset for any trained model, to perform prediction, and monitor the real time situation.

The integration of Raspberry Pi, camera modules, and Amazon Aws Services like S3 Bucket offers promising opportunities for enhancing violence detection systems. By leveraging the computational power of Raspberry Pi for on-device processing and offloading heavier tasks to the cloud, these systems can achieve real-time analysis, remote monitoring, and scalability.

4. TECHNICAL DETAILS:

4.1 HARDWARE REQUIREMENTS:

1) Samsung Evo Micro SD:

Figure 4.1.1: Samsung Evo Micro SD

Brand	Samsung
Flash Memory Type	Micro SD, SDXC, Micro SDXC, SDHC
Memory Storage Capacity	64 GB
Compatible Devices	Smartphone
Colour	Orange, White
Special Feature	Ultra High Speed
Read Speed	48
Item Weight	0.01 Kilograms
Hardware Interface	microSDHC, microSDXC
Secure Digital Association Speed Class	Class 10

2) Raspberry Pi 3B:

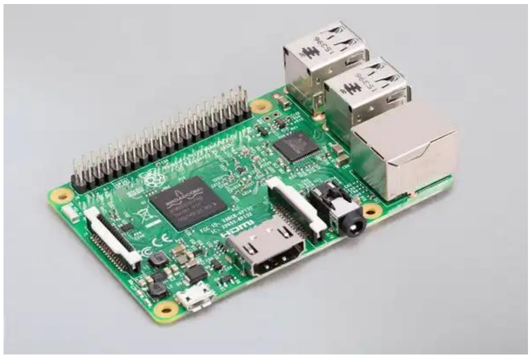


Figure 4.1.2.1:
Raspberry Pi 3B Upper view

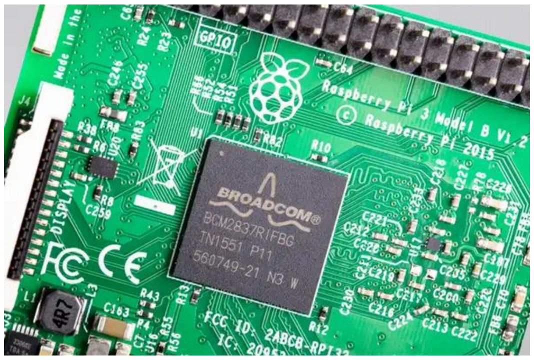


Figure 4.1.2.2:
Raspberry Pi 3B chip view

Figure 4.1.2.3:
Raspberry Pi 3B Side view

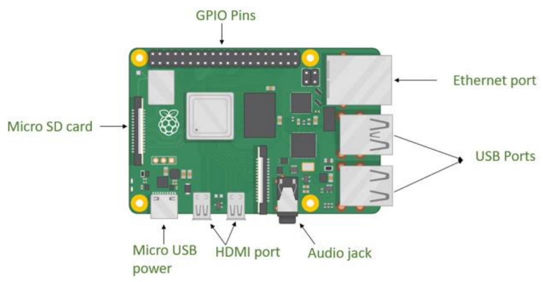


Figure 4.1.2.4:
Raspberry Pi 3B Ports

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT).

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2 ports.
- 4 Pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi camera.
- DSI display port for connecting a Raspberry Pi touchscreen display.

- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source up to 2.5A.

3) Raspberry Pi Camera V2.1:

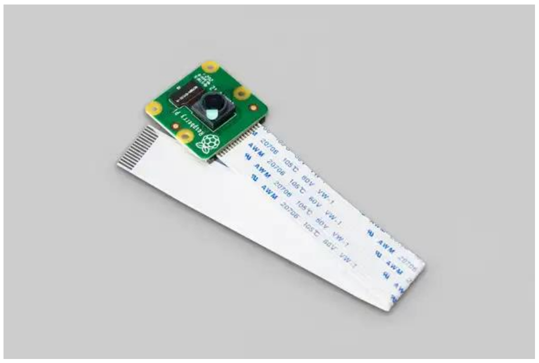


Figure 4.1.3.1:
Camera V2.1 complete view

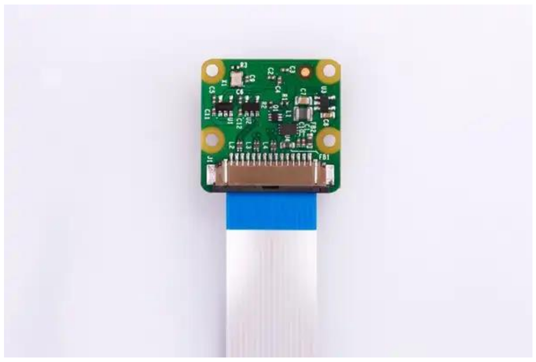


Figure 4.1.3.2:
Camera V2.1 back view

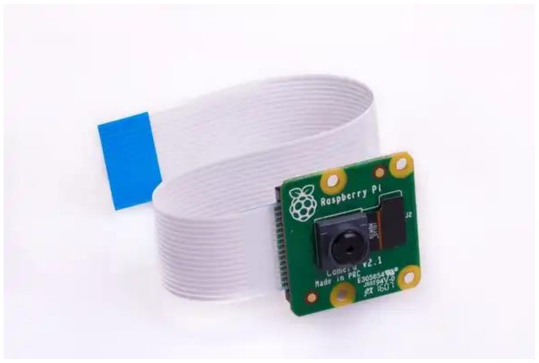


Figure 4.1.3.3:
Camera V2.1 front view

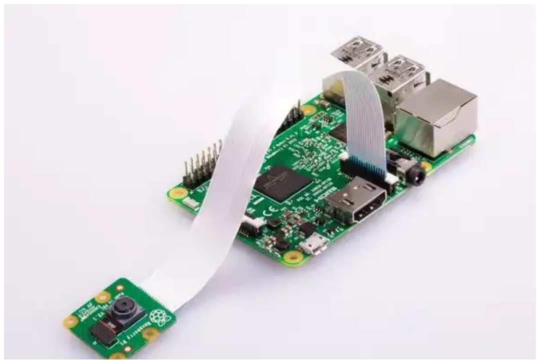


Figure 4.1.3.4:
Camera V2.1 attached with Raspberry Pi

- The Raspberry Pi Camera Module 2 replaced the original Camera Module in April 2016. The v2 Camera Module has a Sony IMX219 8-megapixel sensor (compared to the 5-megapixel OmniVision OV5647 sensor of the original camera).
- The Camera Module 2 can be used to take high-definition video, as well as stills photographs.
- It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.
- The camera works with all models of Raspberry Pi 1, 2, 3 and 4. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.

4) Ethernet Cable:

Figure 4.1.4.1: Ethernet Cable

To connect the Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on the Raspberry Pi to an Ethernet socket on the wall or on your internet router. You don't need to do this if you want to use wireless connectivity, or if you don't want to connect to the internet.

5) USB B-Type Cable:

Figure 4.1.5.1: USB B-Type Cable

- Size: **Around 1.2 Meters**
- Color: **Black / White**

Brand	MI
Connector Type	USB Type A, USB Type B, Micro USB

Cable Type	USB
Compatible Devices	Tablet, Smartphone
Special Feature	Fast Charging

4.2 TECHNOLOGY USED / ALGORITHM USED:

4.2.1 CNN:

Neural networks are a subset of machine learning, and they are at the heart of deep learning algorithms. They are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

There are various types of neural nets, which are used for different use cases and data types. For example, recurrent neural networks are commonly used for natural language processing and speech recognition whereas convolutional neural networks (ConvNets or CNNs) are more often utilized for classification and computer vision tasks. Prior to CNNs, manual, time-consuming feature extraction methods were used to identify objects in images. However, convolutional neural networks now provide a more scalable approach to image classification and object recognition tasks, leveraging principles from linear algebra, specifically matrix multiplication, to identify patterns within an image. That said, they can be computationally demanding, requiring graphical processing units (GPUs) to train models.

<u>How do convolutional neural networks work?</u>

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer

- Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

Convolutional layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

Note that the weights in the feature detector remain fixed as it moves across the image, which is also known as parameter sharing. Some parameters, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters which affect the volume size of the output that need to be set before the training of the neural network begins. These include:

1. The **number of filters** affects the depth of the output. For example, three distinct filters would yield three different feature maps, creating a depth of three.

2. Stride is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater is rare, a larger stride yields a smaller output.

3. Zero-padding is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output. There are three types of padding:

Valid padding: This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.

Same padding: This padding ensures that the output layer has the same size as the input layer.

Full padding: This type of padding increases the size of the output by adding zeros to the border of the input.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

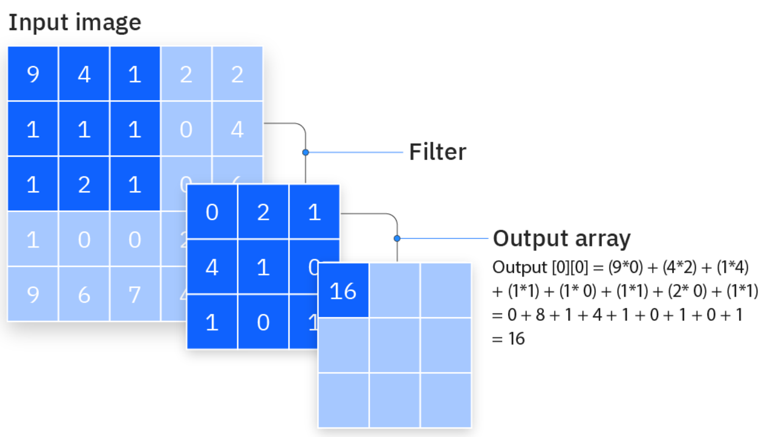


Figure 4.2.1.1: CNN Convolutional layer

<u>Additional convolutional layer</u>

As we mentioned earlier, another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within the receptive fields of prior layers. As an example, let’s assume that we’re trying to determine if an image contains a bicycle. You can think of the bicycle as a sum of parts. It is comprised of a frame, handlebars, wheels, pedals, et cetera. Each individual part of the bicycle makes up a lower-level pattern in the neural net, and the combination of its parts represents a higher-level pattern, creating a feature hierarchy within the CNN. Ultimately, the convolutional layer converts the image into numerical values, allowing the neural network to interpret and extract relevant patterns.

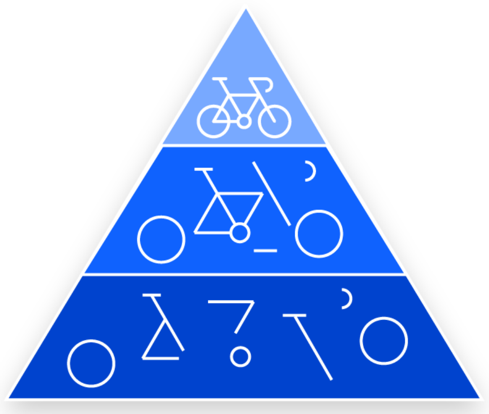


Figure 4.2.1.2: Sample work of Pooling layer

Pooling layer

Pooling layers, also known as down-sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling:

- **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
- **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

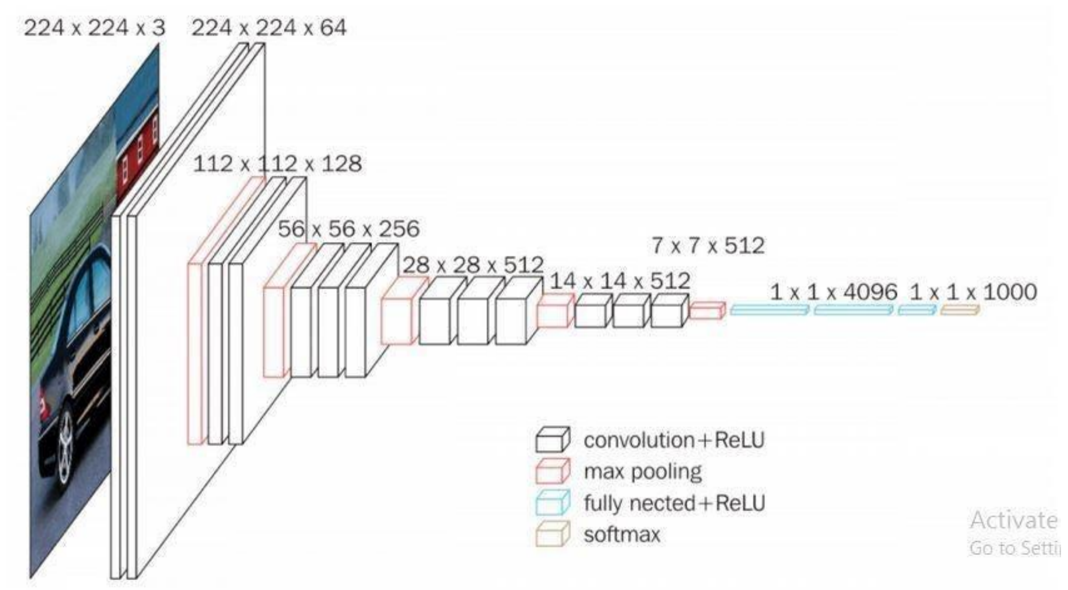
4.2.2 VGG16:

A convolutional neural network is also known as a ConvNet, which is a kind of artificial neural network. A convolutional neural network has an input layer, an output layer, and various hidden layers. VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date. The creators of this model evaluated the networks and increased the depth using an architecture with very small (3 × 3) convolution filters, which showed a significant improvement on the prior-art configurations. They pushed the depth to 16–19 weight layers making it approx — 138 trainable parameters.

What is VGG16 used for?

VGG16 is object detection and classification algorithm which is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

VGG16 Architecture



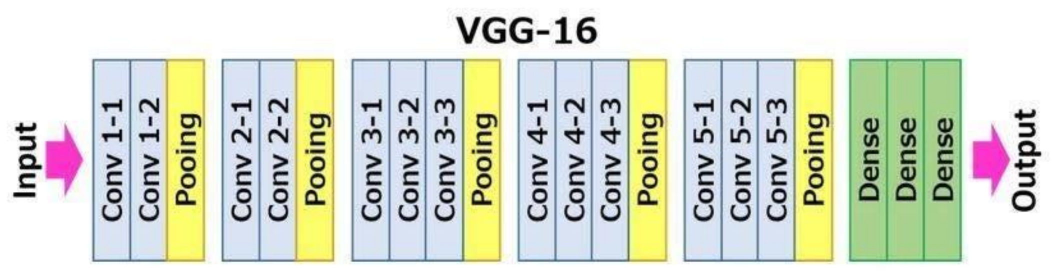
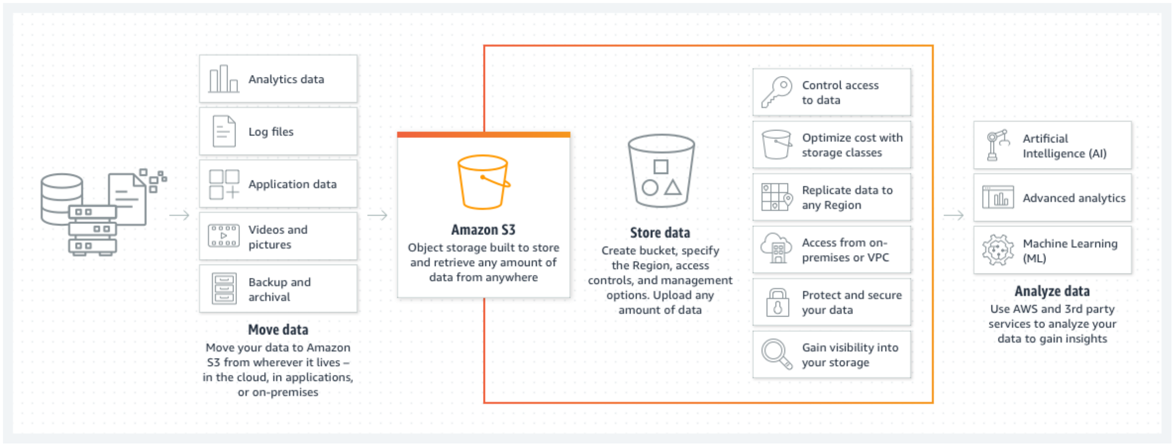


Figure 4.2.2.1: Layer architecture of VGG16

- The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.
- VGG16 takes input tensor size as 224, 244 with 3 RGB channels.
- Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.
- The convolution and max pool layers are consistently arranged throughout the whole architecture.
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.
- Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

4.2.3 AMAZON S3 BUCKET:

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.



Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

Figure 4.2.3.1: Working of Amazon S3 Bucket

4.3 LIBRARY INSTALLED:

4.3.1 Keras API:

Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details. The creation of framework can be of the following two types −

- Sequential API
- Functional API

There are mainly eight steps to create deep learning model in Keras −

- Loading the data
- Preprocess the loaded data
- Definition of model
- Compiling the model
- Fit the specified model
- Evaluate it
- Make the required predictions
- Save the model

Keras is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch.

Keras is:

- **Simple –** but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
- **Flexible –** Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- **Powerful –** Keras provides industry-strength performance and scalability: it is used by organizations including NASA, YouTube, or Waymo.

5. PROPOSED MODEL:

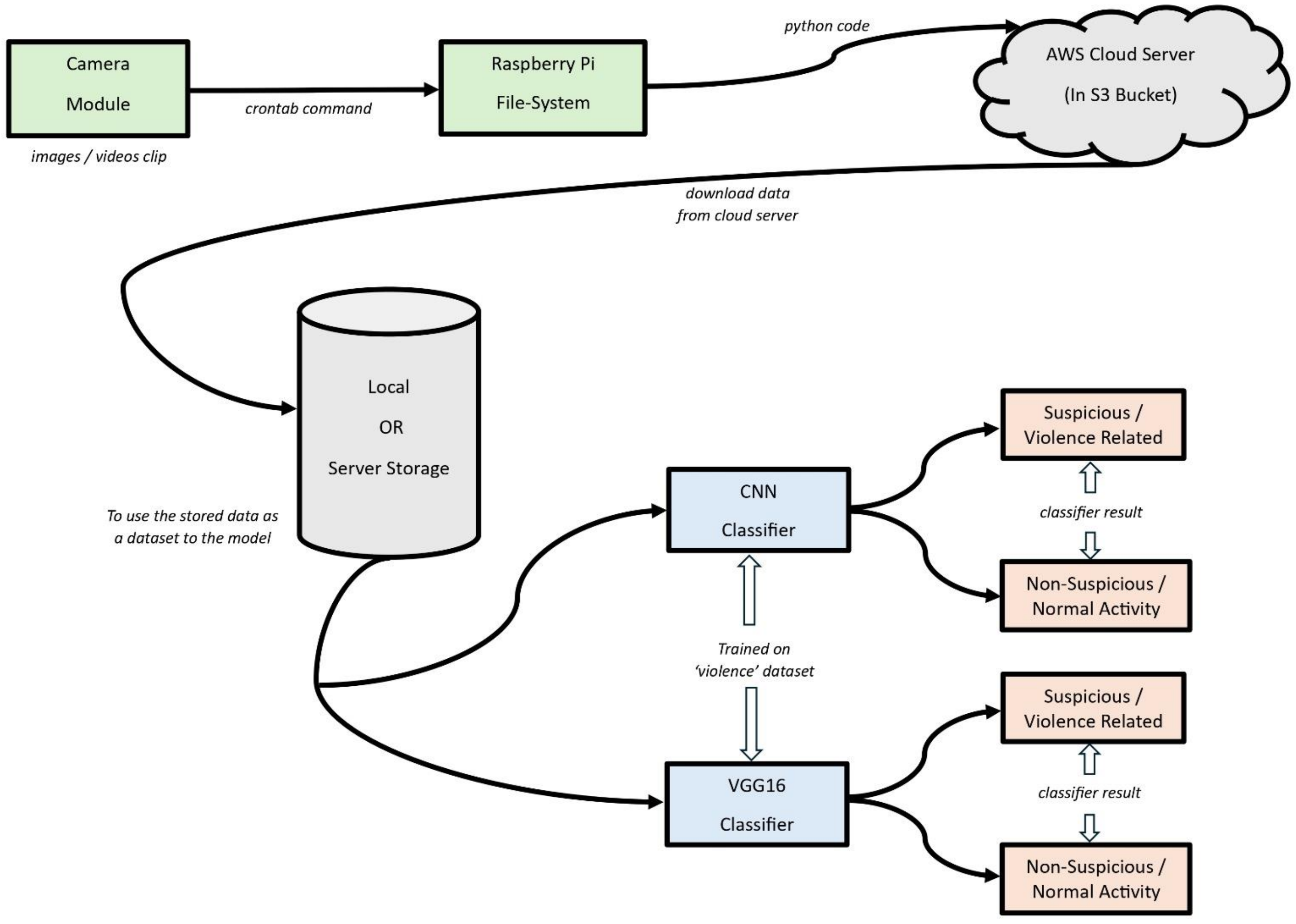

Camera Module
images / videos clip
crontab command
Raspberry Pi File-System
python code
AWS Cloud Server (In S3 Bucket)
download data from cloud server
Local OR Server Storage
To use the stored data as a dataset to the model
CNN Classifier
Trained on 'violence' dataset
VGG16 Classifier
Suspicious / Violence Related
classifier result
Non-Suspicious / Normal Activity
Suspicious / Violence Related
classifier result
Non-Suspicious / Normal Activity

Figure 5.1: Diagram representing proposed model.

The idea is, after the successful arrangement of all the hardware components, that is the connection of camera module with the Raspberry Pi, along with internet access. Once all the manual arrangements are done, the camera module will automatically start capturing the images and video clips after every 3-5 secs of duration and save the captured data over the AWS S3 bucket.

Two Neural Networks are prepared, that will perform the classification tasks, one is the CNN, and another one is theVGG16, both the models are trained on the pre-collected dataset related to violence, once the models are trained, they are good to do the prediction tasks on the data that we have collected over the AWS cloud.

The result of these classifier will classify whether the captures image or video belong to any suspicious or violence related activity or not.

6. CONNECTION OF COMPONENTS:

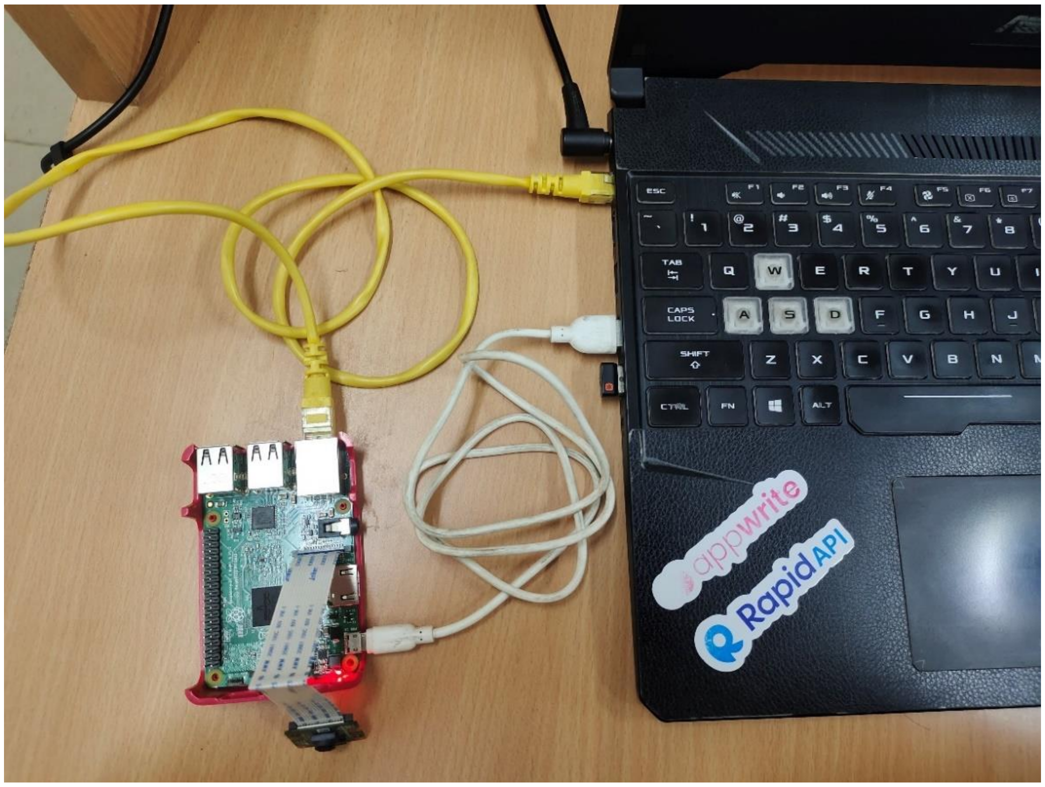


Figure 6.1: Complete hardware connection

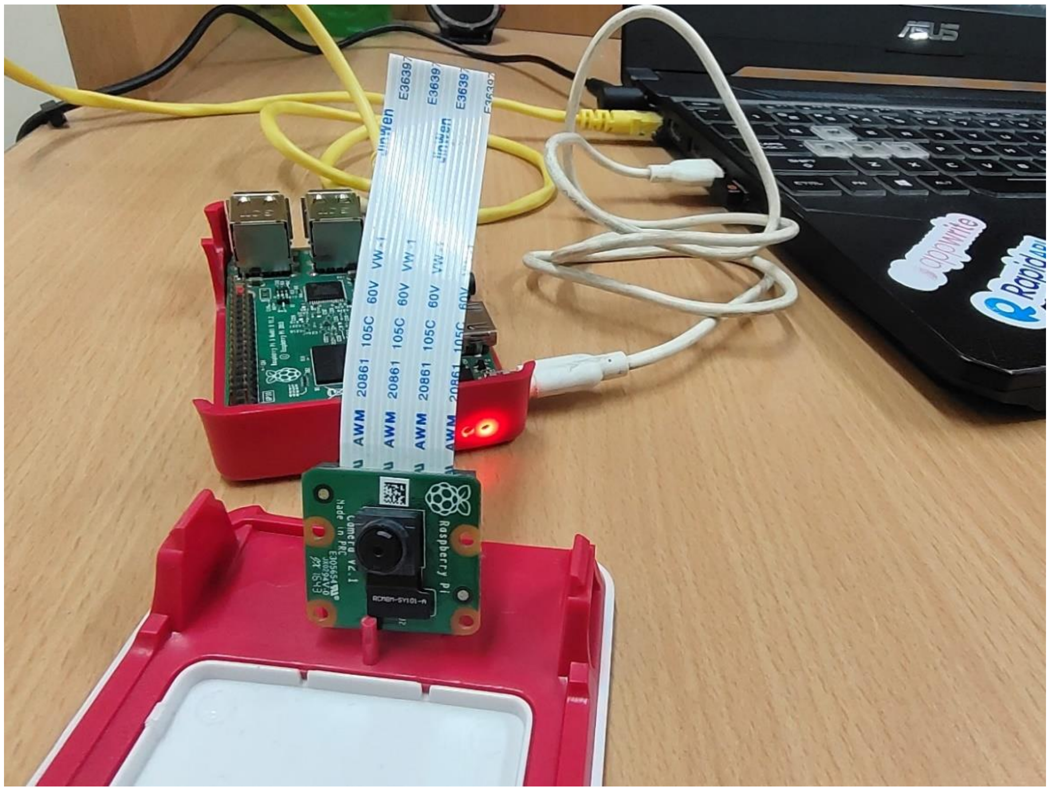


Figure 6.2: Camera module in hardware connection

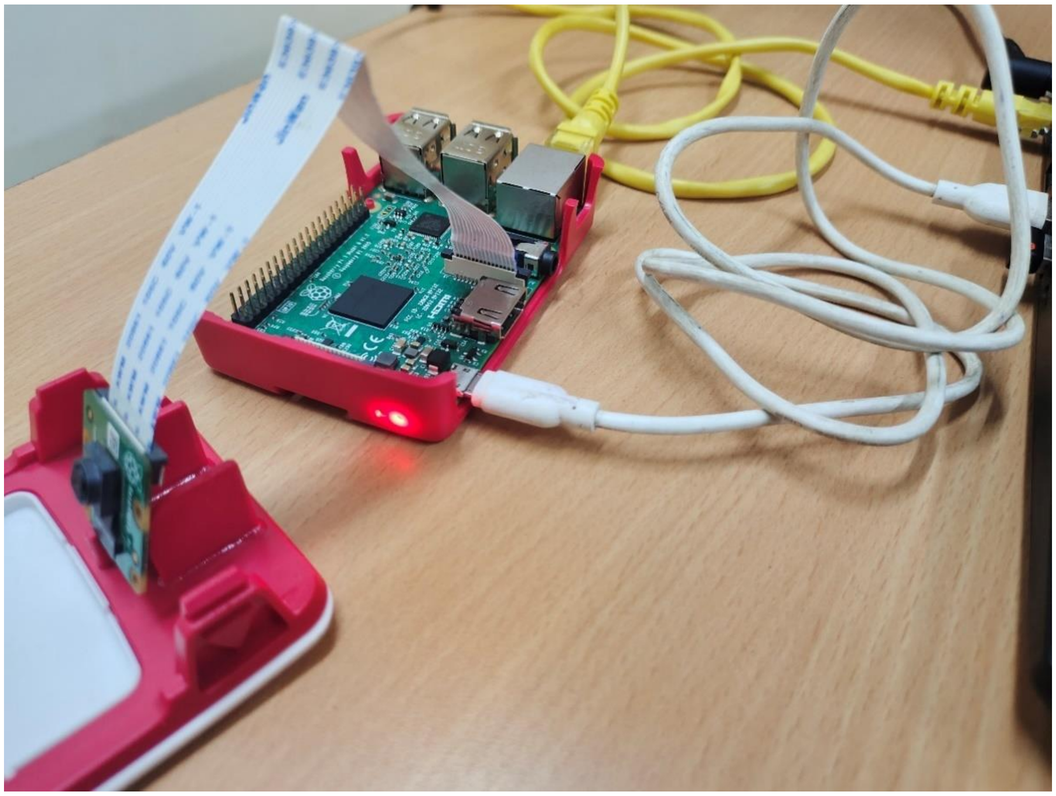


Figure 6.3: Ribbon cable connection

6.1 HARDWARE SETUP:

- **Raspberry Pi Configs:**

  Raspberry Pi 3 Model B V1.2, Raspberry Pi 2015
- **Camera Module:**

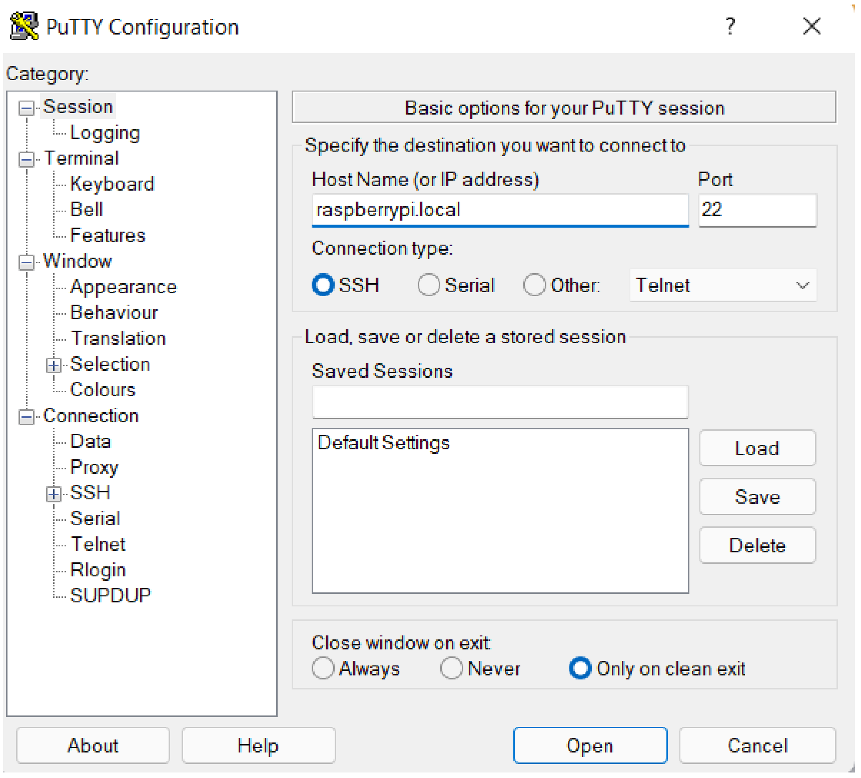
  Raspberry Pi Camera V2.1
- **Step 01:**

  Arrange SD Card or around 32, or 64 GB, so that after loading the OS furthermore software can also be loaded in the Raspberry Pi.
- **Step 02:**
  - Install the Raspberry Pi OS Installer (known as **imager**).

    Use Edit option while selecting the OS version, and give details for username and password, and for Wi-Fi connection.
  - Install **Putty**, so stablish a connection every time with Raspberry Pi hardware.

- Install **VNC-Viewer**, in order the GUI of Raspberry Pi OS, however every task can also be performed from the command line or the terminal itself, but for an ease it is recommended.
- Install **WinSCP**, this will allow to alter or access the file system of the Raspberry Pi hardware. Make it convenient to copy and paste data, files or codes to or from the Raspberry PI OS.

- **Step 03:**
  - Create an empty ssh file with no extension and save in the booted SC Card.
  - Inject the SD Card into the Raspberry Pi hardware.
- **Step 04:**
  - Connect Raspberry Pi hardware to your computer, using the LAN or Ethernet cable.
  - Give power to the Raspberry Pi hardware using the B-Type charger.
- **Step 05:**
  - Open Putty software, type *raspberrypi.local* and hit connect.


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): raspberrypi.local

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

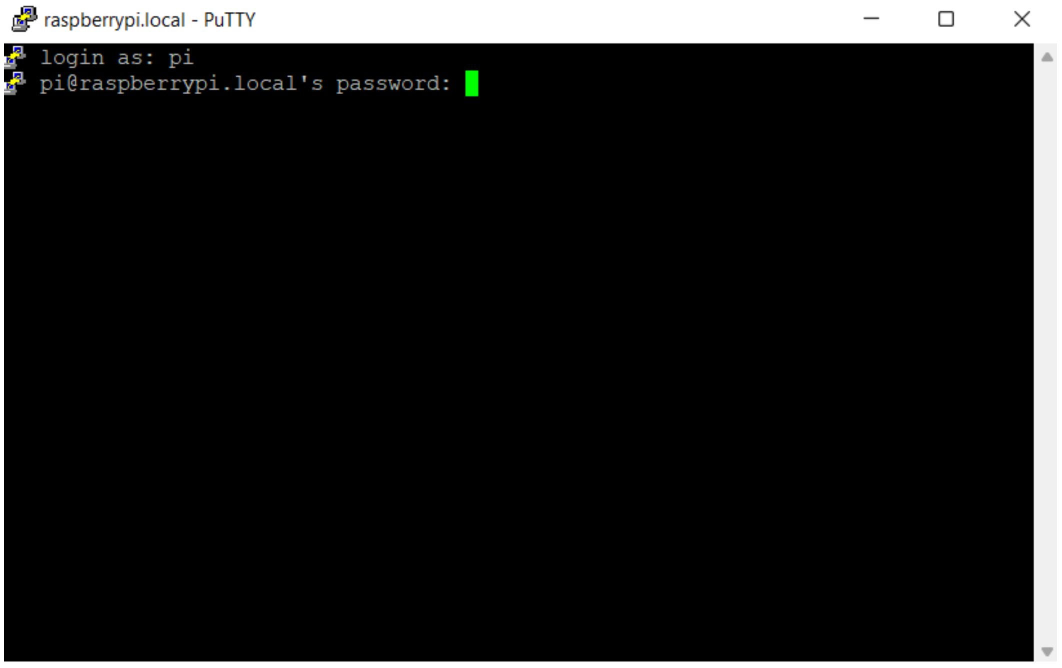
Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Figure 6.1.1: puTTy configuration with Raspberry Pi hardware

- Give the same username and password, as provided earlier for the authentication and logging into the Raspberry Pi OS.



```
login as: pi
pi@raspberrypi.local's password:
```

Figure 6.1.2: login into Raspberry Pi hardware

- **<u>Commands used</u>:**
  - **sudo vncserver-virtual**: To connect with the VNC Server (GUI).
  - **sudo apt-get install idle**: To install Python Idle into the Raspberry OS (For coding purpose).
  - **sudo apt-get update && sudo apt-get upgrade**: To update and upgrade the required dependencies.
  - **sudo raspi-config**: To get the configuration dialog box, enable the camera module from the interfacing options.
  - **sudo raspistill -o 'filename'.jpg**: To capture an image and save in the current directory.
  - **vcgencmd get_camera**: To check whether camera module is detected or NOT.
  - **sudo modprobe bcm2835-v412**: If camera module is NOT detected.

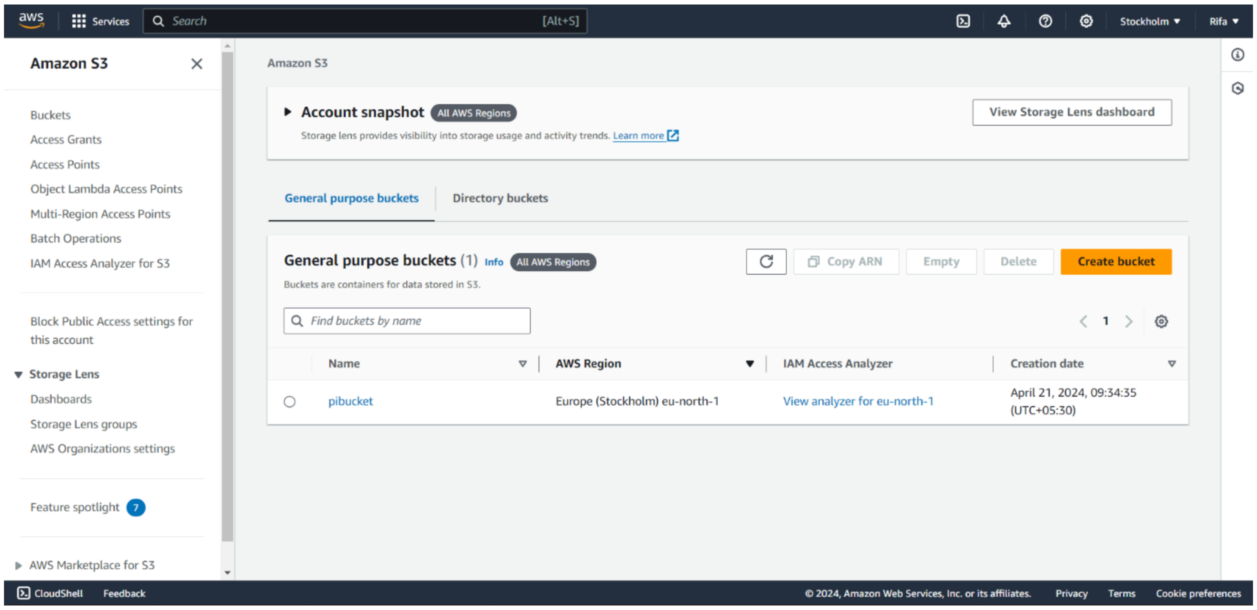
7. DESCRIPTION OF SOLUTION IMPLEMENTED / STEPS OF PROCEDURE:

- **STEP 01:**

Stablish a seamless connection with the camera module in the Raspberry pi hardware in order to capture the images and record the live footage, so that it can be used for detection tasks.

- **STEP 02:**

Make a connection with the cloud architecture (AWS in use), Every simple capture and video will be directly uploaded into the cloud and will form a large database of real-world activities in the multimedia format.



Amazon S3

Storage lens provides visibility into storage usage and activity trends. Learn more

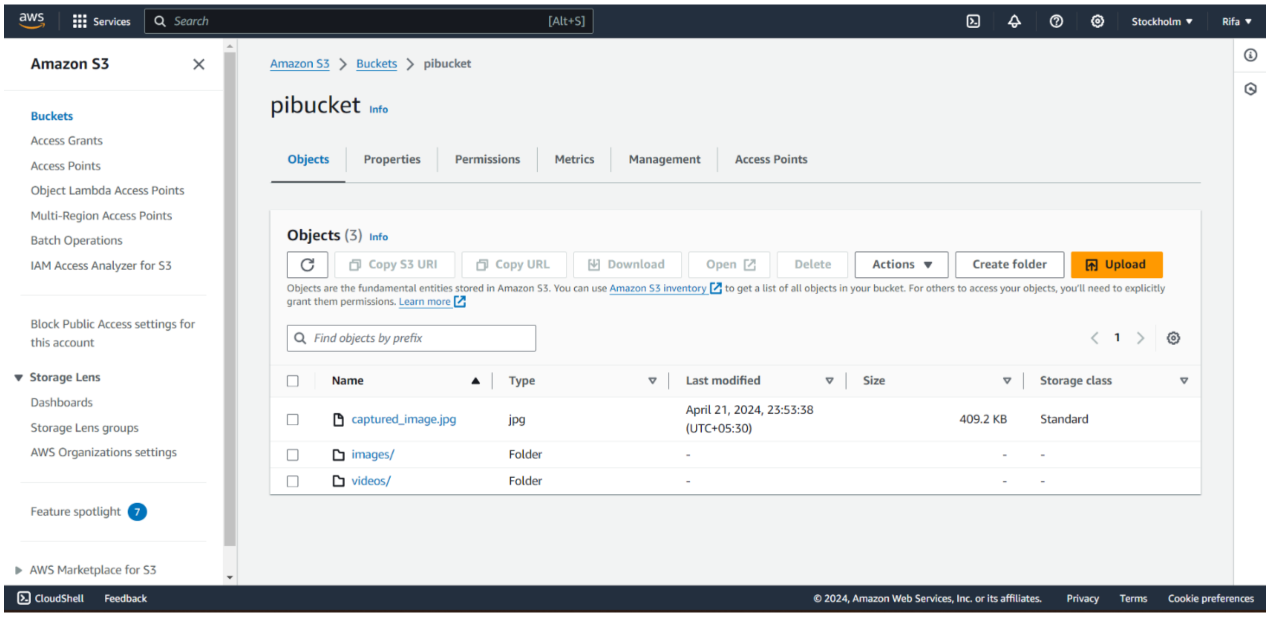
General purpose buckets (1) All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
pibucket	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	April 21, 2024, 09:34:35 (UTC+05:30)

© 2024, Amazon Web Services, Inc. or its affiliates.

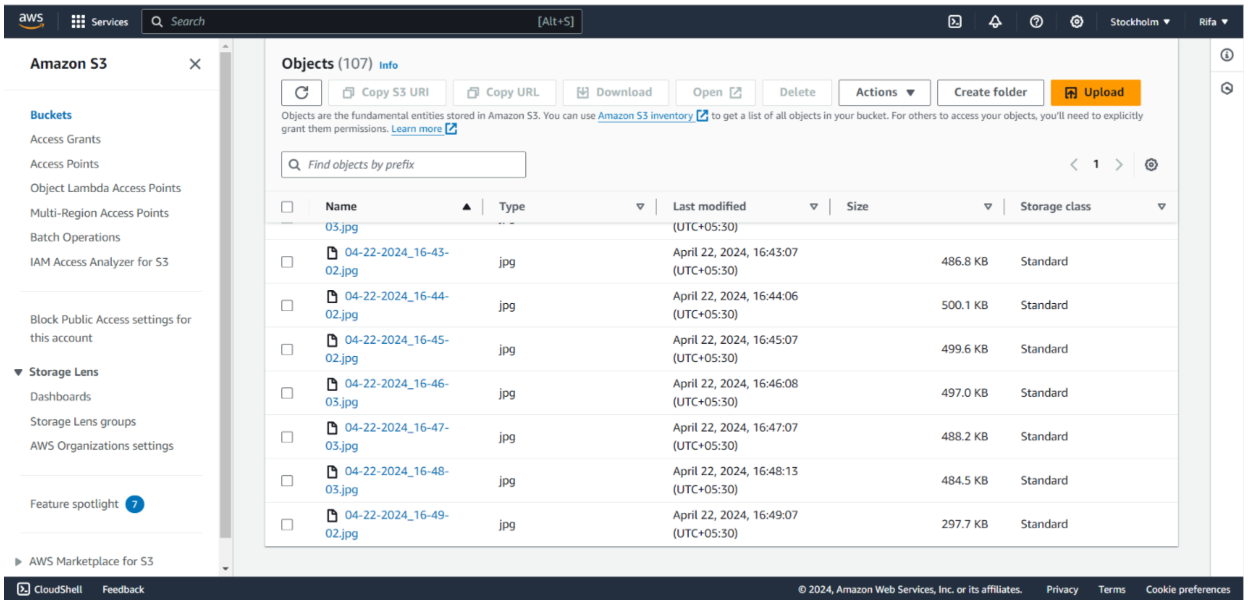
Figure 7.1: Amazon S3 buckets list



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
captured_image.jpg	jpg	April 21, 2024, 23:53:38 (UTC+05:30)	409.2 KB	Standard
images/	Folder	-	-	-
videos/	Folder	-	-	-

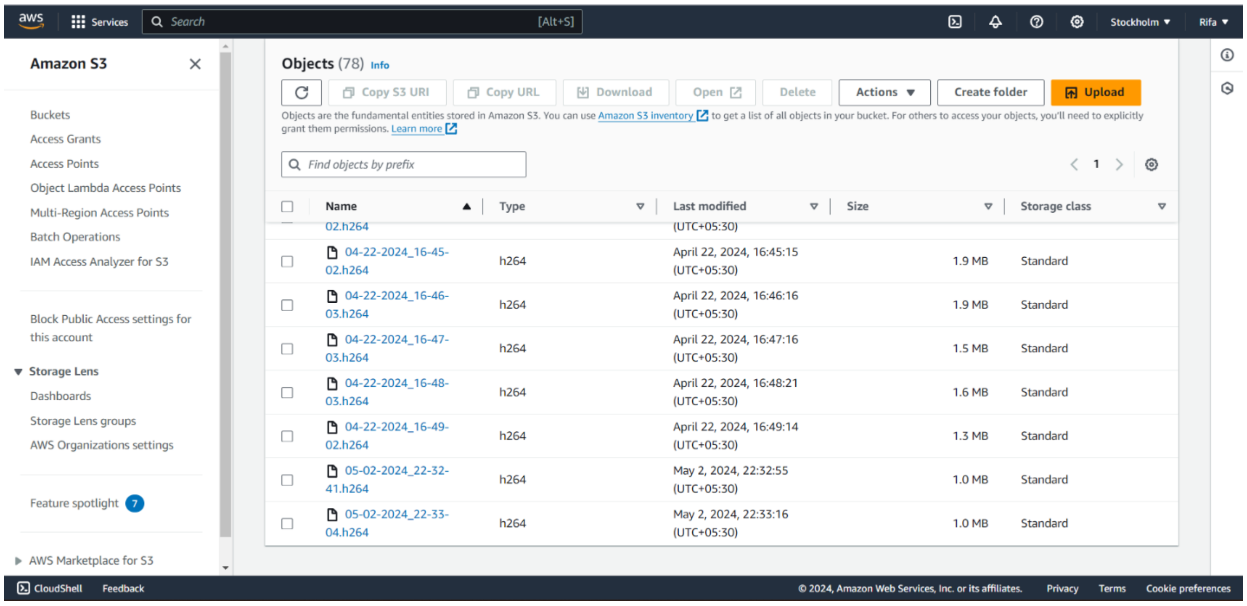
Figure 7.2: Inside S3 pibucket



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
03.jpg		(UTC+05:30)		
04-22-2024_16-43-02.jpg	jpg	April 22, 2024, 16:43:07 (UTC+05:30)	486.8 KB	Standard
04-22-2024_16-44-02.jpg	jpg	April 22, 2024, 16:44:06 (UTC+05:30)	500.1 KB	Standard
04-22-2024_16-45-02.jpg	jpg	April 22, 2024, 16:45:07 (UTC+05:30)	499.6 KB	Standard
04-22-2024_16-46-03.jpg	jpg	April 22, 2024, 16:46:08 (UTC+05:30)	497.0 KB	Standard
04-22-2024_16-47-03.jpg	jpg	April 22, 2024, 16:47:07 (UTC+05:30)	488.2 KB	Standard
04-22-2024_16-48-03.jpg	jpg	April 22, 2024, 16:48:13 (UTC+05:30)	484.5 KB	Standard
04-22-2024_16-49-02.jpg	jpg	April 22, 2024, 16:49:07 (UTC+05:30)	297.7 KB	Standard

Figure 7.3: images directory in S3 pibucket



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
02.h264		(UTC+05:30)		
04-22-2024_16-45-02.h264	h264	April 22, 2024, 16:45:15 (UTC+05:30)	1.9 MB	Standard
04-22-2024_16-46-03.h264	h264	April 22, 2024, 16:46:16 (UTC+05:30)	1.9 MB	Standard
04-22-2024_16-47-03.h264	h264	April 22, 2024, 16:47:16 (UTC+05:30)	1.5 MB	Standard
04-22-2024_16-48-03.h264	h264	April 22, 2024, 16:48:21 (UTC+05:30)	1.6 MB	Standard
04-22-2024_16-49-02.h264	h264	April 22, 2024, 16:49:14 (UTC+05:30)	1.3 MB	Standard
05-02-2024_22-32-41.h264	h264	May 2, 2024, 22:32:55 (UTC+05:30)	1.0 MB	Standard
05-02-2024_22-33-04.h264	h264	May 2, 2024, 22:33:16 (UTC+05:30)	1.0 MB	Standard

Figure 7.4: videos directory in S3 pibucket

- **STEP 03:**
  - Use the CNN model, and train it on a large dataset (Kaggle dataset) to detect the new multimedia data, so that it classifies it to the violence related activity or a normal activity.
  - A pretrained model like, vgg16, resnet50, inception, or any other algorithm trained on the ImageNet dataset can also be used for classification of the image data.
  - Comparison among the several models on the basis of accuracy, and precision will better give us the best model to classify.
- **STEP 04:**

  Any form or arming or alerting system can be further implemented after any suspicious or violence related activity has been detected by the model.

8. PROGRAM CODE:

8.1 CAMERA.PY:

This script will automatically start capturing the images and short video clips of around 3-4 secs after the successful connection of mentions hardware.

```
import time
import picamera
import boto3
import os
import datetime as dt
from botocore.exceptions import NoCredentialsError

def capture_image(file_path):
    with picamera.PiCamera() as camera:
        # Adjust the resolution as per need.
        camera.resolution = (1024, 768)

        CURRENT_DATE = dt.datetime.now().strftime('%m/%d/%Y %H:%M:%S')
        camera.annotate_text = CURRENT_DATE

        camera.capture(file_path)

def capture_video(file_path, duration = 3):
    with picamera.PiCamera() as camera:
        # Adjust the resolution as per need.
        camera.resolution = (1024, 768)
        camera.start_recording(file_path)
        camera.wait_recording(duration)
        camera.stop_recording()

def upload_to_s3(file_path, bucket_name, object_name):
    aws_access_key_id = "AKIA6KGA7GO62DXDR5F6"
    aws_secret_access_key = "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"
    s3 = boto3.client('s3', aws_access_key_id = aws_access_key_id, aws_secret_access_key = 
aws_secret_access_key)

    try:
        s3.upload_file(file_path, bucket_name, object_name)
        print(f"File uploaded to S3: s3://{bucket_name}/{object_name}")
    except FileNotFoundError:
        print("The file was not found.")
    except NoCredentialsError:
        print("Credentials not available or incorrect.")

def main():
    # Change the path to your desktop path.
    data_path = '/home/pi/Desktop/captured_data/'
    # Replace with your bucket name.
    bucket_name = 'pibucket'

    IMAGE_NAME = dt.datetime.now().strftime('%m-%d-%Y_%H-%M-%S')
    VIDEO_NAME = dt.datetime.now().strftime('%m-%d-%Y_%H-%M-%S')

    # Capture image.
    image_file_path = data_path + IMAGE_NAME + ".jpg"
    capture_image(image_file_path)
    upload_to_s3(image_file_path, bucket_name, 'images/{}'.format(IMAGE_NAME + ".jpg"))

    # Capture video.
    video_file_path = data_path + VIDEO_NAME + ".h264"
    capture_video(video_file_path)
    upload_to_s3(video_file_path, bucket_name, 'videos/{}'.format(VIDEO_NAME + ".h264"))

    # Removing the captured data, from local directory, after uploading.
    # os.remove(data_path + IMAGE_NAME +'.jpg')
    # os.remove(data_path + VIDEO_NAME +'.h264')
```

```
if __name__ == "__main__":
    main()
```

8.2 CNN MODEL:

- Link to file containing the complete code along with output: G-Drive
- Link to Google Colab showing the saved runtime space along with code: G-Colab

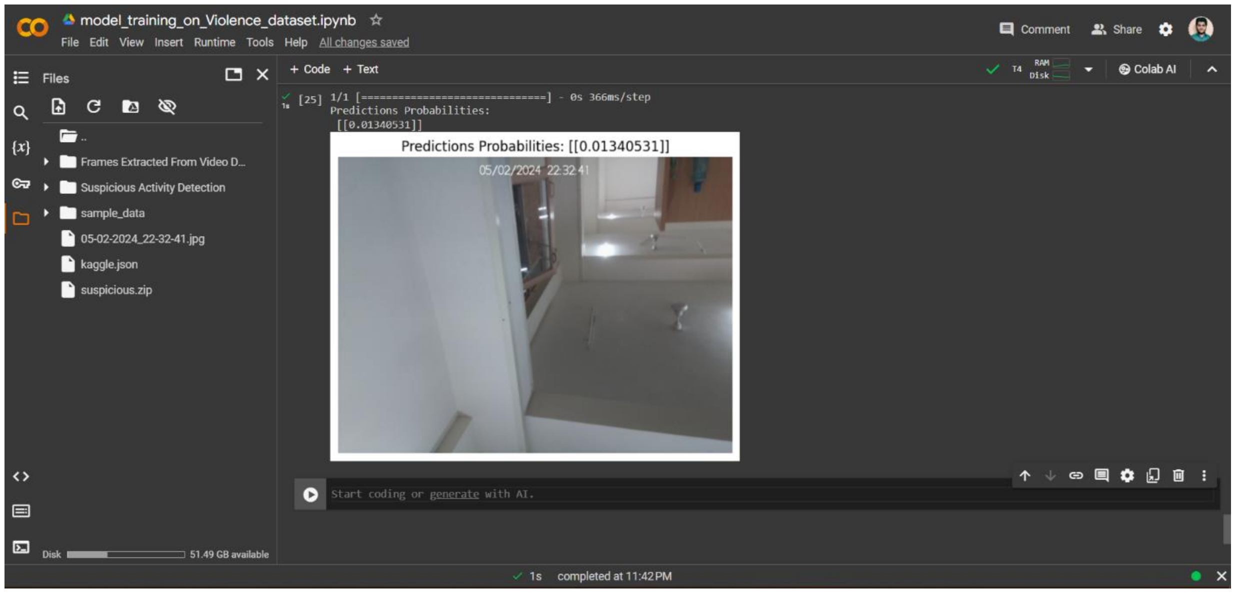
8.3 VGG16 MODEL:

- Link to file containing the complete code along with output: G-Drive
- Link to Google Colab showing the saved runtime space along with code: G-Colab

9. RESULTS:

9.1 CNN MODEL:

```
Validation Loss:     0.0006173738511279225
Validation Accuracy: 0.9998052716255188000
```



```
[25] 1/1 [==============================] - 0s 366ms/step
Predictions Probabilities:
 [[0.01340531]]
```

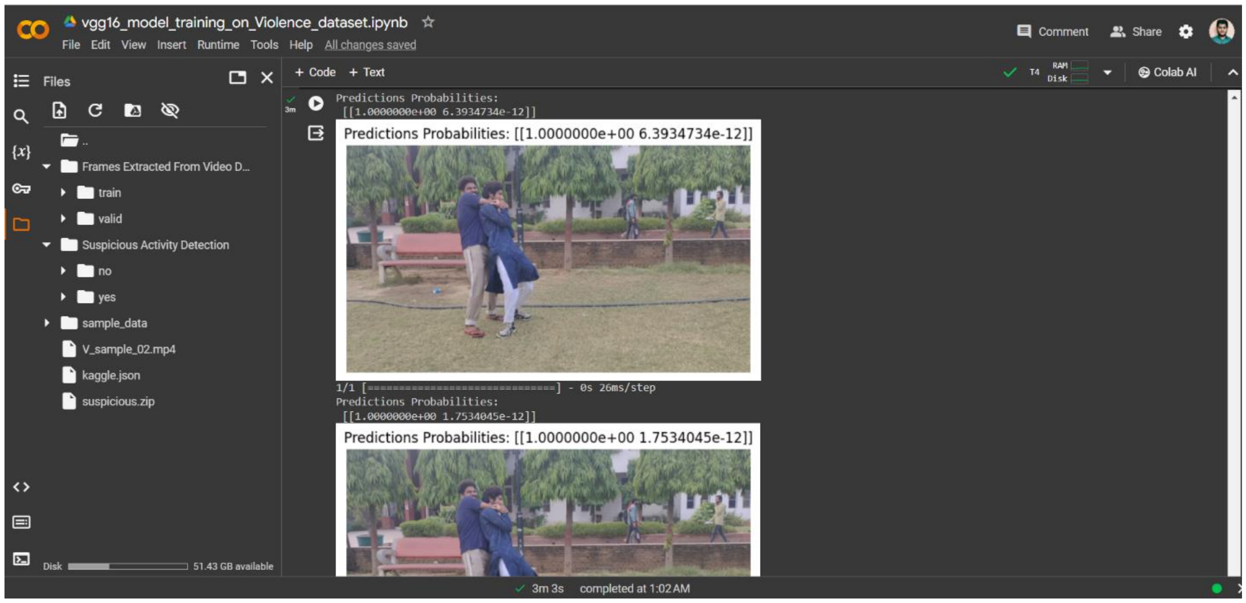
Predictions Probabilities: [[0.01340531]]

model_training_on_Violence_dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
Frames Extracted From Video D...
Suspicious Activity Detection
sample_data
05-02-2024_22-32-41.jpg
kaggle.json
suspicious.zip
05/02/2024 22:32:41
Start coding or generate with AI.
Disk 51.49 GB available
1s completed at 11:42 PM

Figure 9.1.1: Result of CNN model
when run on the non-violence type real time data.

9.2 VGG16 MODEL:

```
Validation Loss:     0.0019463554490357637
Validation Accuracy: 0.9998067617416382000
```



```
Predictions Probabilities:
 [[1.0000000e+00 6.3934734e-12]]
```

Predictions Probabilities: [[1.0000000e+00 6.3934734e-12]]

```
1/1 [==============================] - 0s 26ms/step
Predictions Probabilities:
 [[1.0000000e+00 1.7534045e-12]]
```

Predictions Probabilities: [[1.0000000e+00 1.7534045e-12]]

vgg16_model_training_on_Violence_dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
Frames Extracted From Video D...
train
valid
Suspicious Activity Detection
no
yes
sample_data
V_sample_02.mp4
kaggle.json
suspicious.zip
Disk 51.43 GB available
3m 3s completed at 1:02 AM

Figure 9.2.1: Result of VGG16 model
when run on the violence type real time data.

10. MAJOR PROBLEMS IDENTIFIED:

The major problems that are encountered during the development phase of the system in order to run and get the desired working from the system are as follows:

- Each hardware component should be compatible to every other so that the transfer of data can be done smoothly.
- There should be a proper internet connection with the Raspberry Pi hardware, so that the continuous forming of dataset in the form of images and video clips can be done.
- The captured images and video clips format should be compatible with the model dataset to perform the prediction task.
- Camera module should be properly connected to the Raspberry Pi hardware, sometimes this might happen that the Raspberry Pi hardware does not detects the Pi Camera module. In this case the data will not be collected.
- To predict the class of the collected data, which will classify whether the video clip belongs to the Suspicious or Violence related activity or not, the complete video clip should be broken down into the number of frames.
- In such case each frame is treated as the individual image, and the prediction is performed over all the frames of video.
- To which class the maximum number of video frames belong will be computed and the complete video clip is considered that it belongs to that specific class only.

11. FUTURE SCOPE:

Some of the key factors that will boost the overall system performance in the future are as follows:

- **Advanced AI Algorithms**: Continued research and development in AI algorithms could lead to more sophisticated image and video analysis capabilities. This includes improving detection accuracy, enhancing anomaly detection algorithms, and developing AI models to predict potential accidents based on historical data and real-time environmental factors on.
- **Predictive analytics**: Predictive analytics can be incorporated into the system to enable proactive accident prevention strategies. By analyzing patterns, trends, and hazards, the system can quickly issue warnings or recommendations to reduce the likelihood of accidents, such as identifying high-risk areas or predicting vehicle behavior profiles.
- **Autonomous Vehicle Integration**: Autonomous vehicles are equipped with a plethora of sensors, cameras, LiDAR, and GPS systems that continuously gather data about their surroundings, including road conditions, nearby vehicles, pedestrians, and obstacles. By integrating these vehicles with the cloud-enabled system, real-time data exchange and communication channels are established. This allows autonomous vehicles to share critical data related to accidents, road hazards, and traffic conditions with the centralized cloud platform.
- **Blockchain enhances data integrity:** The use of blockchain technology can enhance data integrity, transparency, and security in accidental data management. By creating accident data, blockchain can prevent tampering, facilitate trusted data sharing among stakeholders, and simplify insurance claims processing.
- **Environmental sensors and IoT extensions:** Adding environmental sensors to IoT devices can provide additional contextual information, such as weather conditions, road surface conditions, and increased vehicle density This provision of data enhances accident analysis and helps to understand the external factors contributing to accidents.
- **Research and industry collaborations:** Collaborate with research institutes, industry partners, government agencies, and nonprofits to promote innovation, knowledge sharing, and technology adoption Collaboration can conduct pilot projects, field tests, and real-world applications to drive robust system optimization and improvement.

12. REFERENCES:

I. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

II. https://www.raspberrypi.com/products/camera-module-v2/

III. https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started/3

IV. https://www.ibm.com/topics/convolutional-neural-networks

V. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

VI. https://www.tutorialspoint.com/tensorflow/tensorflow_keras.htm

VII. https://opensource.com/resources/raspberry-pi

VIII. https://aws.amazon.com/s3/