**PATTERN RECOGNITION ALGORITHM TO DETECT SUSPICIOUS ACTIVITIES**

**A PROJECT REPORT**

**Submitted by**

**KAVIN S V (715519104018)**

**ROOPAKUMAR S (715519104041)**

**SHYAM GANESH T (715519104048)**

**VIKHAS S G (715519104058)**

***in the partial fulfilment of award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH**

**COIMBATORE - 641 062**

**ANNA UNIVERSITY: CHENNAI - 600 025**

**MAY – 2023**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“PATTERN RECOGNITION ALGORITHM TO DETECT SUSPICIOUS ACTIVITIES ”** is the Bonafide work ofKavin S V(715519104018), Roopakumar S (715519104041), Shyam Ganesh T (715519104048), Vikhas S G (715519104058), who carried out the project work under my supervision.

**-------------------------**  **-------------------------**

**SIGNATURE**  **SIGNATURE**

Dr. Manimegalai R Dr. Mahavishnu V C

**HEAD OF THE DEPARTMENT SUPERVISOR**

Professor and Head Assistant Professor (Senior Grade)

Computer Science and Engineering Computer Science and Engineering

PSG Institute of Technology and PSG Institute of Technology and

Applied Research, Applied Research,

Coimbatore – 641 062 Coimbatore – 641 062

**Submitted for the project viva-voce Examination held on 18/05/2023**

**-------------------------------**   **----------------------------------**

**INTERNAL EXAMINER**  **EXTERNAL EXAMINER**

**ACKNOWLEDGMENT**

First and foremost, we express our heartfelt gratitude to our honorable Managing Trustee, **SHRI. L GOPALAKRISHNAN** for his invaluable advice and moral support. We would also like to express our deepest gratitude to our beloved Principal, **Dr. G. CHANDRAMOHAN, B.E.(Hons), M.Tech, Ph.D,** for his overwhelming support and encouragement on this project.

I take this opportunity to extend my humble gratitude to the Secretary of our institution, **Dr. P. V. MOHANRAM, B.E.(Hons), M.Tech, Ph.D.**

We are greatly indebted to **Dr. R. MANIMEGALAI, M.E, Ph.D** Head of the Department, Computer Science and Engineering for her guidance and continuous support which was instrumental in the completion of this project.

We extend our thanks to our guide, **Dr. Mahavishnu V C**, **M.E, Ph.D** Assistant Professor (Senior Grade) for his technical support and constant supervision without which we could not have completed this project study.

Finally, we would also like to whole heartedly thank our project coordinator **Ms. S. HEMKIRAN, M.E, (Ph.D)** Assistant Professor (Selection Grade) and **Lt. V. VILASINI, M.E, (Ph.D)** Assistant Professor (Senior Grade), for carrying out reviews smoothly and the valuable feedback provided at each step of the project development.

**Kavin S V**

**Roopakumar S**

**Shyam Ganesh T**

**Vikhas S**

**ABSTRACT**

The rise in shootings, knife attacks, terrorist attacks, and other such incidents worldwide has made it imperative to detect suspicious activities in public places. Video surveillance is instrumental in achieving this, and with the advent of advanced technologies like artificial intelligence, machine learning, and deep learning, it is now possible to differentiate between normal and suspicious behavior.

Detecting suspicious activity in an academic environment can be quite challenging, as human behavior is unpredictable. To overcome this challenge, a deep learning approach is employed to identify suspicious or normal activity and alert the appropriate authority in the event of suspicion. This monitoring is done through consecutive frames extracted from the video, and simple installation of traditional CCTV is not enough as it requires continuous human monitoring.

Thus, there is a need for a fully-automated security system that recognizes anomalous activities in real-time and provides instant help to victims. To address this need, we have proposed a system that examines and detects suspicious human actions from real-time CCTV footage with the help of machine learning techniques, generating an alert if abnormal activity is observed.

**LIST OF TABLES**

**TABLE NO.**  **TITLE**   **PAGE NO.**

6.1 Result of model 36

**LIST OF FIGURES**

**FIGURE NO.**   **TITLE**   **PAGE NO.**

1.1 Input – Output of Neurons 3

1.2 Layers of a Neural Network 3

4.1 Neuron 17

4.2 CNN 1D 18

4.3 CNN 2D 18

4.4 Input to feature map 19

4.5 CNN 3D 19

4.6 Padding 20

4.7 Pooling 21

4.8 Flattening 21

4.9 CNN Architecture 21

4.10 ReLu plot 22

4.11 Function graph ReLu 22

5.1 System Workflow 25

5.2 Pooling layer 27

5.3 Software Activation Function 27

5.4 ReLu Activation Function 28

6.1 Dataset Snippet 30

6.2 GUI for prediction model 30

6.3 Converting video to frames 31

6.4 Result of prediction 31

6.5 Model Training 32

6.6 Saving frames in jpg format 32

6.7 Saved images 33

6.8 Model Architecture 34

6.9 Accuracy function for keras 35

6.10 Loss function for keras based CNN 35

6.11 Accuracy function for Mobilenet V2 35

6.12 Loss function for Mobilenet V2 36

**LIST OF ABBREVIATIONS**

CPU Central Processing Unit

GPU Graphic Processing Unit

ANN Artificial Neural Networks

CNN Convolutional Neural Networks

API Application Program Interface

GUI Graphical User Interface

NN Neural Networks

ReLu Rectified Linear Unit

**LIST OF CONTENTS**

**CHAPTER** **TITLE**   **PAGE NO.**  **NO.**

**ABSTRACT**   **I**

**1** **INTRODUCTION** 1

1.1 INTRODUCTION 1

1.2 OBJECTIVE 1

1.3 PROBLEM STATEMENT 1

1.4 PROJECT OVERVIEW 2

1.5 SCOPE AND MOTIVATION 2

1.6 INTRODUCTION TO NEURAL 2

NETWORKS

1.6.1 BUILDING BLOCKS: NEURONS 2

1.6.2 COMBINING NEURONS INTO A 3

NEURAL NETWORK

1.6.3 DEEP LEARNING 4

1.7 NEURAL NETWORKS IN IMAGE 4

PROCESSING

**2**  **LITERATURE REVIEW** 6

2.1 EXISTING SYSTEMS 6

**3**  **SYSTEM DESCRIPTION** 12

3.1 KERAS 12

3.2 TENSORFLOW 12

3.3 NUMPY 12

3.4 MATPLOTLIB 13

3.5 PANDAS 13

3.6 MOBILENETV2 13

3.7 FLASK 14

3.8 OPENCV 14

**4**  **SYSTEM DESIGN** 16

4.1 COLLECTION OF DATA 16

4.2 PROPOSED METHODOLOGY 16

4.3 MODEL DEVELOPMENT 17

4.4 CHOICE OF NEURAL NETWORK USED 18

4.4.1 CNN ARCHITECTURE 18

4.4.2 CONVOLUTION WITH PADDING 19

4.4.3 POOLING AND FLATTENING 20

4.4.4 INTRODUCING NON-LINEARITY 22

4.4.5 IMAGE ENHANCEMENT 22

4.4.6. EDGE DETECTION 23

**5**  **SYSTEM IMPLEMENTATION 25**

5.1 SYSTEM FLOW 25

5.1.1 THE FEATURE EXTRACTOR 25

5.2 SETTING UP THE ENVIRONMENT 25

5.3 MODEL IMPLEMENTATION 26

**6**  **RESULT AND ANALYSIS 29**

6.1 TESTING THE NEURAL NETWORK 29

6.2 RESULT ANALYSIS 29

**7** **CONCLUSION AND FUTURE**  **ENHANCEMENTS 37**

7.1 CONCLUSION 37

7.2 FUTURE ENHANCEMENT 37

**REFERENCES 39**

**PLAGIARISM REPORT 44**

**APPENDIX 45**

**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION**

Surveillance is the act of monitoring and observing behaviour, activities, or information for the purpose of protecting, influencing, managing, or directing them. This can be done through various means such as CCTV cameras or interception of electronic information like phone calls and internet traffic. The captured data can be used to prevent crimes and also serves as valuable forensic evidence.

Human operators traditionally monitor CCTV feeds, but the development of image processing and computer vision has made it possible to recognize suspicious human activity from surveillance videos. Visual surveillance is necessary to monitor sensitive and public areas to prevent terrorism, theft, accidents, and other suspicious activities. However, it is difficult to continuously monitor public places, which is why an intelligent video surveillance system is required to categorize human activities as normal or unusual and generate alerts accordingly.

Anomalous activity refers to a person's abnormal behavior that causes harm to either themselves or their surroundings. To detect such behavior, one of the widely used methods is to train a model using videos of normal events and identify any suspicious events that do not fit the learned model. The process of training a surveillance system involves three phases: data preparation, model training, and inference. In this context, neural networks can be used to overcome the challenges. Recognition of anomalous human activity from surveillance video is an area of active exploration in image processing and computer visualization.

**1.2 OBJECTIVE**

The aim of this project is to monitor and detect the suspicious activities and report it to the authorities before the action is performed. To avoid unwanted riots and harm to someone, the suspicious activities are reported to the authorities.

**1.3 PROBLEM STATEMENT**

The model is developed to avoid human threats which are potentially affect the human lives by taking continuous monitoring and surveillance. It identifies and classifies the type of threat and reports it to the authorities as soon as possible.

**1.4 PROJECT OVERVIEW**

The detection and monitoring of suspicious activities could potentially be monitored using a variety of different sensors and drones. These activities are detected and classified based on the KERAS TENSORFLOW based CNN algorithm and MobileNet v2 which are used to train the models.

**1.5 SCOPE AND MOTIVATION**

Building a machine learning system to detect suspicious activity is not easy. In today's world, the variety of images lying out there is so vast that pointing out features and highlighting them is a hard task. Machine learning replaces this exploitation process with training of neural networks with enough data to avoid overfitting. This approach becomes convenient compared to the other approaches out there. With training, the model can differentiate easily between type of activities.

**1.6 INTRODUCTION TO NEURAL NETWORKS**

Neural networks are based on self-learning resulting from history which occurs within networks, which can derive results from a complex and seemingly unrelated set of information.

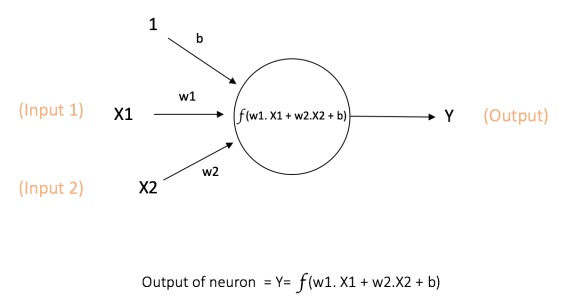
According to the neurocomputer's inventor, Dr. Robert Fecht-Nielson, neural networks consist of simple, interconnected elements that help process and assess the inputs given.

**1.6.1 BUILDING BLOCKS: NEURONS**

Living beings work out information in their brains, i.e., their cognitive centres. A brain consists of trillions of neurons (nerve cells), which transfer and process information through impulses. Artificial neural networks (ANN) mimic these natural structures.

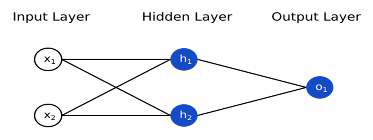
Neurons are the fundamental building blocks of a neural network as well. Neurons act as receptors and receive information, process it with mathematical operations, and produce the result.

Researchers in neural networks are driven by two desires, deepening the understanding of the human brain, as well as the development of computers that can solve conceptual and undefined problems.

Figure 1.1 Input - output of neurons

**1.6.2 COMBINING NEURONS INTO A NEURAL NETWORK**

There are n number of connected neurons.

Figure 1.2 – Layers of a Neural Network

The network in the figure has – input : 2, hidden : 1 with 2 neurons in it, and output : 1. which has a single neuron (o1). Here it can be seen that the o1 gets its input from the results of h1 and h2 - this makes it all a neural network. Any layer which is present between the input and output ones is called as hidden layer.

Before training the network, categorization in a way to quantify the network to analyse how good the network performs, is a must. This helps find the loss.

A better prediction = a lower loss.

So, the main aim is to minimize the loss.

**1.6.3 DEEP LEARNING**

Andrew gave a presentation in 2013 on "Deep Learning, Self-Taught Learning, and Unsupervised Feature Learning," where he explained how deep learning with traditional ANNs works. The main concept is to use brain simulation to enhance the learning algorithm and make it more user-friendly. Brain simulations can also bring about ground-breaking developments in machine learning and artificial intelligence.

The speaker believed that this was the best way to advance artificial intelligence. Deep learning is a type of neural network technology in which humans neurological system is reflected to process the data and form patterns for automated decision making. It is a subset of machine learning in AI that can learn and understand unstructured data. Deep learning can be referred to interchangeably as deep neural learning or deep neural networks.

The main goal is to learn the feature hierarchy from higher to lower levels. The levels are ANNs, built by connected neurons like in the human brain. This helps employ nonlinear approaches as opposed to linear processing of data. Since multiple levels are used in the learning process, it helps in learning complex problems and gives output without human dependency. For higher efficiency, deep learning requires huge amounts of data. With the advent of faster processing capabilities, larger neural networks can be worked on.

**1.7 NEURAL NETWORKS IN IMAGE PROCESSING**

Before moving on to neural networks, clarity is required in the field of solving problems. The first step is to understand and break it down into smaller tasks and identify the recurring parts to create a generalized function. The project mimics the same step in image processing too, the only difference being that each pixel need not be parsed. Some pixels may have a greater range of coverage, for example - a picture of a tree. It can be broken down into grids and if the result is a green grid, the probability of the nearby grids being green is high. Focusing on specific boundaries and identifying different features, however, is necessary.

Any neural network should eliminate the need for manually designing feature vectors. With Convolutional Neural Networks, the relevant features are automatically extracted.

Related Terminologies:

1. Locally Connected Networks

* It restricts the connection between the hidden and input layers, so each layer is connected to parts of the input layer rather than whole.

2. Convolutions

* Used to extract features.
* Done with convolutional masks.

3. Pooling

* The process of aggregation is called pooling.
* Since images have stationary properties, it can be compute using pooling.
* There are various aggregation methods policies like mean, average and max pooling.

They are composed of convolutional, sub-sampled, fully connected layers. Fully connected layers are discretionary.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EXISTING SYSTEMS**

The reliable system for identifying suspicious human activity behaviour using deep learning techniques was proposed by Buttar , Ahmed Mateen , Faisalabad , Bano , Akbar ,Muhammad Azeem ,Azeem. Akbar Lut.Fi, in their research “Toward trustworthy human suspicious activity detection from surveillance videos using deep learning". Since Continuous monitoring of surveillance videos in public areas for detecting abnormal events is a challenging task, development of automated video detection system is precious. In this model, A dataset comprising of diverse videos related to each activity was used. The frames were extracted from the original video, and features were calculated through the Inception v3 model, a variant of Convolutional Neural Network. This model exhibited superior performance and accuracy compared to other models.

Authors: Leela S, K V Sai Likhita, et al proposed that suspicious activity cause serious threaten to personal safety and identified Suspicious Human Activity Recognition and Alarming System. This system uses deep learning-based CNN algorithm which has disadvantages that it requires lot of time. A CNN was utilized to segregate and identify the spatial attributes at a particular time point in the input sequence which is a video format file in this work. An LSTM was used to detect temporal similarities within adjacent frames of the video. CNN models were primarily focused for image classification tasks. Feature learning is the stage where the model learns an internal representation of an image that is represented mathematically as a two-dimensional input. It has a 96% accuracy rating.

Suspicious Activity Detection System was proposed by Sumon et al. The Algorithm used in this system were Faster R-CNN and YOLOv3 technique. The main aim of this system is to automate video surveillance input images are not in expected way to feed as input, different image preprocessing methods are used in this system to improve the quality of the image. The advantage of this paper is usage of faster CNN and sematic based approach in place of standard dataset unavailability. But , this system got accuracy of only 57%.

Monali Ahire, Devarshi Borse, et al CNN architecture to detect anomaly and suspicious activities. This paper presents a self-evaluating system, which makes use of a highly professional video monitoring tool for surveying of potentially suspicious criminal activities that happens in and around shopping malls. In this paper, they have utilized HaarCascade method which is highly suitable for body detection but still it poses as complex in computational aspect. This system not only identify suspicious activity but also dangerous object with accuracy of nearly 92%.

Video surveillance systems are now more crucial than ever in ensuring safety and security. These systems use real-time applications that include behavior recognition to differentiate between normal and suspicious activities. Suspicious activities can pose a significant threat to individuals, which is why intelligent surveillance systems have implemented various approaches to address this issue. In a recent paper, Sumon Ghosh, Prasham Shah, Aditya Ghadge, Vaibhav Sanghavi, and Dr. Vaqar Ansari proposed a neural network model called Faster R-CNN and the YOLOv3 technique to detect such activities. The authors also introduced a deep neural network that can identify guns in images, which can be integrated into existing surveillance systems to mitigate threats to human lives. Additionally, an algorithm was developed to detect ATM loitering, exam cheating, and wall climbing. Their method for detecting abandoned baggage is computationally efficient, achieving an extremely low false alarm rate while effectively detecting most abandoned items.

Ms Archana R. Ghuge et al. have proposed an Advance Suspicious Activity Detection System, which aims to modernize security measures beyond the traditional video monitoring approach. The system performs two crucial functions: identifying and recognizing faces to help trace suspects involved in criminal activity and detecting suspicious activity through a trained network. The proposed system utilizes a Deep Neural Network (DNN) and the CNN technique, which significantly reduces processing time. The Faster-RCNN frame sampling technique is also utilized to classify images and count unique classes. This system is highly applicable to real-time video processing requirements [21].

A group of experts - C. Chalmers, P. Fergus, et al - have proposed a method for detecting suspicious activity using consumer-grade drones and convolutional neural networks with faster-region-based technology. Their approach involves video analysis, and they conducted two experiments to evaluate the trained model's performance. The first experiment measured the model's object detection performance using standard metrics such as mAP and IOU. In the second experiment, they assessed the model's inferencing capabilities by using it to analyze unseen test data from YouTube[18].

Digambar Kauthkar, et al proposed that an interesting topic of computer vision and image processing research is the detection of suspicious human activity and combat from video information. To train this system, deep learning and neural networks has been used. Applications for it in real life include gesture recognition, AR/VR, and gaming. The application of CNN in video classification is comparatively simple compared to the picture data domain. In this model, they extract a more number of images from the video. Hence one disadvantage of this system is if the video is large, creating frames will require more time. But, The advantage is that this model is suitable for CCTV images [15].

Video surveillance systems are now more crucial than ever in ensuring safety and security. These systems use real-time applications that include behavior recognition to differentiate between normal and suspicious activities. Suspicious activities can pose a significant threat to individuals, which is why intelligent surveillance systems have implemented various approaches to address this issue. In a recent paper, Sumon Ghosh, Prasham Shah, Aditya Ghadge, et al proposed a neural network model called Faster R-CNN and the YOLOv3 technique to detect such activities. The authors also introduced a deep neural network that can identify guns in images, which can be integrated into existing surveillance systems to mitigate threats to human lives. Additionally, an algorithm was developed to detect ATM loitering, exam cheating, and wall climbing. Their method for detecting abandoned baggage is computationally more efficient, achieving mostly low false alarm rate while effectively detecting most abandoned items. [25].

Suspicious Activity Detection and Tracking through Unmanned Aerial Vehicle Using Deep Learning Techniques was proposed by Madala Gayathri, et al This paper discusses the need for credible and effective surveillance to diminish the occurrence of crimes, which often result from late communication and a lack of authentic security or surveillance systems. The author proposes the idea of an Intelligent Unmanned Aerial Vehicle (UAV) inspired by various sources, capable of monitoring its surroundings constantly for suspicious or illegal activities. The UAV captures and processes the scene when it encounters suspicious activity, sends out an alert to the administrator, and waits for their command. The aim is to monitor the surroundings using a bird-shaped surveillance device with minimal human intervention, especially in places like ATMs and shopping malls, where a secur ity system is required to prevent heinous offenses

This paper discusses the use of a UAV device for crime detection in surveillance videos, with the advantage of using Raspberry Pi for efficient image processing at the device level. The proposed approach employs a CNN model for detecting suspicious activity and RNN for matching, improving performance. The previous approach using Arduino Uno was not efficient for image processing and required continuous transmission of frames to the administrator[10].

Ryosuke Kakiuchi, Dinh Tuan Tran, et al conducted a research study to evaluate the utilization of an night security drone that has an infrared camera and projector attached to it to detect human behavior at night time and provide necessary information. The study aimed to address the shortage of security guards in Japan. The AUD was tested in an experiment and proved to be effective in offering near to real time spectation and information projection from the atmosphere. The study suggests that in person security guards can be changed over with mechanized automated technology with the AUD can improve the situation in human power management in the future and lead to the development of better security methods.

This paper proposes the use of an aerial ubiquitous display (AUD) as a night security drone to address the problems of lack of labour and pain faced physically by security personnel in Japan. The AUD is equipped with an infrared camera and projector to detect human behaviour at night and project information to those in need. The experiment evaluated the effectiveness of the AUD in three scenarios, showing a 72.7% overall success rate in monitoring and projecting information. The study suggests the need for more accurate action detection models, increasing the altitude limit of behaviour detection, and improving self-localization estimation techniques[19].

The paper which was proposed by Tejashri Subhash Bora1, Monika Dhananjay Rokade2 finds solution to one of the most important applications of human suspicious activity (anomaly) detection. Inexpensive depth sensors have drawbacks such as less indoor usage, low resolution picture or image quality, and a highly grained output which are fed as input to the system. This information makes it difficult to estimate human pose from images. So they used neural networks to overcome these problems.

Detecting Suspicious Mankind Activity From Surveillance Video acted as an major Investigation when the domain is about computer visualization and image processing. The result of the proposed system can detect if any unusual actions occur or not. Also, most previous studies had poor accuracy in identifying abnormal behavior. Therefore, a better variant of CNN is deployed in this system for better outcomes[14].

Jefferson Ryan Medel, Andreas Savakis propose an convolutional long-short-term memory i.e. Conv-LSTM for end to end trustable network building for suspicious activity detection that can guess the upcoming video sequences from a small number of input frames given as input. This model uses composite structures and analyzes the impacts of conditioning on extracting features on highly meaningful representations which are selected based on accuracy of the prediction. The Conv-LSTM model has been evaluated under quality conditions and also under quantitative platforms and has successfully evaluated to be an highly confidential tool for predicting video sequences that are yet to come using patterns and modeling purposes. It is shown quantitative analysis that their model performs competitively with edge computing based anomaly detection methods on several datasets.

The Faster-RCNN explains the frame sampling technique which is a useful tool for real-time video processing. It successfully addresses the image classification and unique class counting research questions in the paper. C. Chalmers, P. Fergus, et al proposed the use of CNN and people affordable domestic grade drones for video analysis to detect animals. In the first experiment, they used a faster RCNN to offer a standardized object detection performance metrics such as mAP and IOU for the trained model. The second experiment evaluated the inferencing capabilities of the trained model using test data that can not be interpreted from YouTube. [9].

Multimodal Fusion for Suspicious Activity Detection in Smart Environments. This literary piece talks about detecting suspicious activities in smart environments through the use of multimodal fusion techniques. The authors suggest a framework that integrates data from different sensors, such as video cameras, audio microphones, and motion detectors, to capture a comprehensive representation of activities. By combining information from various modalities, the proposed approach boosts detection accuracy and minimizes false alarms. The study evaluates the framework on a large-scale dataset gathered from smart home environments, demonstrating the effectiveness of multimodal fusion in identifying a broad range of suspicious activities, including intrusion, theft, and abnormal behavior. The paper also discusses the challenges and opportunities in implementing such a system in real-world scenarios. Overall, this literature provides valuable insights into utilizing multimodal fusion to enhance the efficiency and effectiveness of suspicious activity detection in smart environments.

Graph-based Suspicious Activity Detection in Social Networks . The literature delves into the use of graph-based techniques for identifying suspicious activities in social networks. The authors introduce a method that models social interactions as graphs, with individuals as nodes and connections as relationships. This approach analyses the structural properties of these graphs to identify anomalous patterns that suggest suspicious behavior. The study provides a thorough evaluation of this method across several real-world social network datasets, showcasing its ability to detect various types of suspicious activities including fraud, terrorism, and cyberbullying. Additionally, the paper compares the proposed method with other state-of-the-art techniques, highlighting its accuracy and scalability advantages. These findings contribute towards the development of robust techniques for detecting suspicious activities in social networks, allowing for proactive measures to be taken to mitigate potential risks [7].

**CHAPTER 3**

**SYSTEM DESCRIPTION**

**3.1 KERAS**

A high-level python NN framework and API. TensorFlow is the base library used. It enables easy and fast experimentations, prototyping. It supports CNN and RNN and also runs seamlessly on the CPU and GPU. Keras is an open-source Python deep learning package. It offers a high-level API for creating and training neural networks, making it simple to design models for a variety of machine learning tasks.

Keras simplicity of use is one of its primary advantages. It provides a straightforward and user-friendly interface for creating and training neural networks, allowing developers to concentrate on the logic of their models rather than low-level implementation concerns. Keras also supports many different neural network topologies, such as feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and others.

**3.2 TENSORFLOW**

TensorFlow is a software library that operates on data flow graphs, specifically for numerical operations. Its nodes correspond to mathematical operations while its edges represent multidimensional arrays. It supports one or more CPUs or GPUs and was originally designed for machine learning and neural network research but can be used for other domains as well.

**3.3 NUMPY**

It is developed for scientific calculations and operations in Python. Uses an n-dimensional array structure, derived objects (such as masked arrays and matrices). Basically, it is handy when you need to operate routines requiring fast operations for mathematical, logical or shape manipulation problems. It is an np array object.

NumPy is a popular Python numerical computing package that offers strong data structures for encoding multi-dimensional arrays and matrices. Its primary data structure, the ndarray, allows for the fast and flexible storage and manipulation of massive datasets, while a variety of mathematical functions enable operations such as element-wise arithmetic, linear algebra, and statistical analysis. NumPy is well-known for its efficiency, with computations completed significantly quicker than with pure Python code, and its memory management capabilities enable dealing with extremely big datasets. It's extensively used in scientific computing for things like data analysis, visualization, machine learning, and scientific simulations, and it's the foundation for other Python libraries like Pandas and SciPy.

**3.4 MATPLOTLIB**

Matplotlib is a Python-based open-source library for creating visualizations that are static, animated, and interactive. Although it is mostly written in Python, some parts are written in C, Objective-C, and Javascript for platform compatibility. This comprehensive library can be used in various user interfaces such as Python scripts, IPython shells, Jupyter notebooks, web applications, and GUI toolkits. Additionally, it supports labels and texts formatted in LaTeX.

**3.5 PANDAS**

Pandas is a toolkit for Python that specializes in data manipulation and analysis. This toolkit offers data structures that effectively process and organize data, along with tools for data cleansing, transformation, and aggregation. The two fundamental data structures in Pandas are Series and DataFrame. A Series is a single-dimensional array that has labels and can include any data type, such as integers, floats, and texts. On the other hand, a DataFrame is a two-dimensional table that has rows and columns, and each cell can have a distinct data type.

Pandas also has extensive indexing and filtering features, allowing you to easily choose and alter subsets of data. It also includes features for dealing with missing data, merging and connecting databases, and altering data. One of Pandas' primary features is its ability to effectively manage big datasets. It supports reading and writing data from a wide range of file formats, including CSV, Excel, SQL databases, and others. It also has tools for working with time-series data, which makes it a popular choice for financial and economic research.

**3.6 MobileNetV2**

Google introduced MobileNetV2 in 2018, a neural network model for image categorization and object recognition workloads. It is an upgraded version of the original MobileNet architecture, which was built for mobile and embedded devices.

MobileNetV2 employs a combination of depthwise separable convolutions and linear bottlenecks to minimize the network's computational complexity and memory requirements while retaining high accuracy. Depthwise separable convolutions convert a regular convolution into a depthwise convolution followed by a pointwise convolution, lowering the number of parameters and calculations required. Linear bottlenecks minimize the number of computations even further by utilizing 1x1 convolutions with a bottleneck layer in between. The MobileNetV2 model can be fine-tuned for a variety of tasks, including image classification, object detection, and semantic segmentation. It has achieved high accuracy on benchmark datasets such as ImageNet and COCO, while requiring significantly fewer computations and memory than other state-of-the-art models.

**3.7 FLASK**

Flask is a popular Python web framework that allows developers to quickly and simply create online apps. Flask has a number of tools and functionality for creating web APIs, including RESTful APIs that may be accessed by other applications.

A Flask API is typically made up of a number of endpoints that may be accessed through HTTP queries. Each endpoint is a function that handles a certain URL and HTTP method and delivers a response in a format chosen by the user, such as JSON or XML. Flask has a number of tools for interacting with HTTP requests and responses, such as request and response objects, which can be used to retrieve data sent in requests and construct responses to send back to clients. Flask also includes capabilities for authentication, authorisation, and input validation, as well as support for web standards like CORS and HTTP caching.

Flask's adaptability and simplicity are two of its primary advantages. Flask enables developers to quickly and simply design APIs with little boilerplate code. It also interfaces well with other Python modules and tools, making extension and customization simple.

**3.8 OPENCV**

OpenCV (Open Source Computer Vision) is a machine learning and computer vision library that was first released in 2000. It offers a variety of image and video processing tools and techniques, as well as object detection and tracking and machine learning.

Machine learning methods are also included in OpenCV for applications like as classification, grouping, and regression. These techniques can be applied to a variety of tasks, including image identification, face detection, and gesture recognition. OpenCV is widely used in the computer vision and machine learning sectors, and has been utilized in a variety of applications including robots, augmented reality, and self-driving cars. It's also commonly utilized in the entertainment sector for things like special effects and computer-generated imagery (CGI).

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 COLLECTION OF DATA**

The dataset for this project consists of images of suspicious activities. The images were originally of different dimensions and were resized so that a uniform dataset was formed. Training sets are data sets used to build the models. To train the model, specific features are selected from the training set. The model then incorporates these features. Testing the model's predictions on the test set is a measure of how well it performs. Before running the program, the images weren't divided into training sets and testing sets. An initial set of classified scanned images is all that is required, which is later split - based on the components required in the code module.

**4.2 PROPOSED SYSTEM**

Keras is a user-friendly and efficient interface for addressing machine learning issues. It is the high-level API of TensorFlow that emphasizes contemporary deep learning. The framework offers crucial components and foundations for developing and delivering machine learning solutions with a rapid iteration speed. Keras is also a versatile platform that can accommodate cutting-edge research concepts. Keras follows the principle of gradual disclosure of intricacy.

In RCNN, there are 3 main networks. First is pretrained CNN (ResNet101) which is used to generate a feature map of images. The feature map is the output from the last fully connected layer from a CNN which contains important features of the image like edges and shapes. The key concept in RPN is anchor. Anchor is used to put bounding box with fixed size to reshaped image which is used as a reference for objection localization.

The MobileNet v2 design incorporates an inverted residual structure that utilizes narrow bottleneck layers for both input and output in the residual block. This stands in contrast to traditional residual models which rely on expanded representations in the input. In addition, MobileNet v2 implements lightweight depthwise convolutions to filter features in the intermediate expansion layer. Suspicious activities are identified and categorized using a CNN algorithm based on KERAS TENSORFLOW. The resulting models are trained using MobileNet v2.

**4.3 MODEL DEVELOPMENT**

Neural networks were used to develop the model. Various numbers of nodes and neural networks can be used.

Biological neurons have a cell body, dendrites to receive signals, an axon to send messages to other neurons. Artificial neurons have multiple input channels, a processing stage, and one output, also shown in the figure–

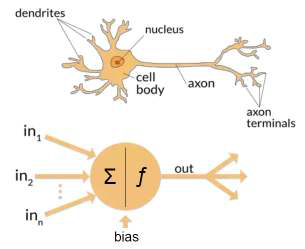


Figure 4.1 -Neuron

Layers of an ANN include:

**Input Layer**

Receives signals, information from external environments in the form of samples and patterns. These inputs can be normalised within the range done by the activation function. Therefore, better precision can be achieved for mathematical operations which are performed in the layers of the network.

**Hidden Layer**

The hidden layer goes between the input and output layers. It extracts the pattern associated with any process and is responsible for major processing.

**Output layer**

The hidden layer goes between the input and output layers.

**4.4 CHOICE OF NEURAL NETWORK USED**

The Neural Network algorithm used in our project is Convolutional Neural Networks (CNNs). CNNs are mainly proved very effective in image processing especially in image recognition and classification. They are mainly used for classifying the features in the image like faces, objects, traffic signs etc.

Generally speaking, it refers to patterns using the terms "kernel", "filter" or "feature detector" in CNN terminology. Using corresponding filters, the image data with CNN extracts the signal features and predictions based on the signals.

**4.4.1 CNN ARCHITECTURE**

In the one-dimensional case, to facilitate the detection of the point in which the value changed, the filter [-1,1] should be used.

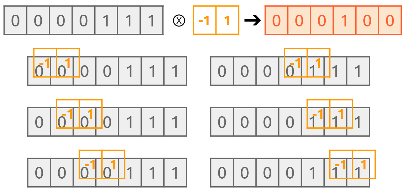
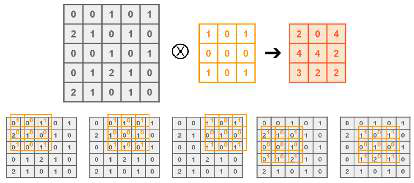


Figure 4.2 CNN 1D

With the convolution filter, it can pass it from left to right to calculate its value. Similarly, for 2-dimensions, the convolution can be calculated as below figure.

Figure 4.3 – CNN 2D

Convolution is done on two objects which produce the resultant object as one object modified by another. So, by this the feature of the input image can be detected. This is referred to as a ‘feature map’. The below is an example.

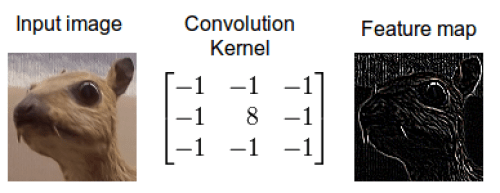


Figure 4.4 – Input to feature map

For 3-dimensional cases 2-dimensional convolution is applied, layer by layer.

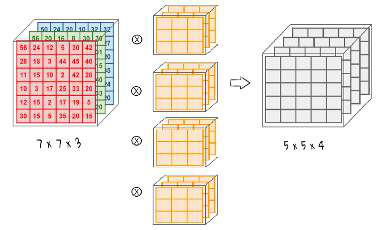
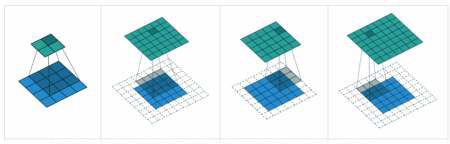
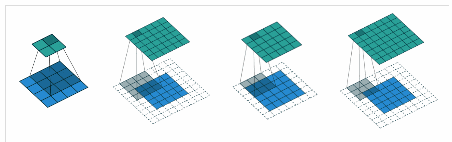


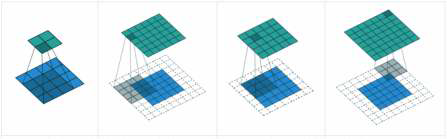
Figure 4.5 – CNN 3D

Every image data consists of 3 basic colours. They are red, green and blue(RGB). For a 7X7 image it will have 7X7X3 data.

**4.4.2 CONVOLUTION WITH PADDING AND STRIDE**

Each pixel is assigned with different weight, it is not mandatory that they must be the same. Apply convolution to the required number of times. If applied too frequently - data maybe be lost fast. Padding is adding additional pixels in the boundary of the image matrix.





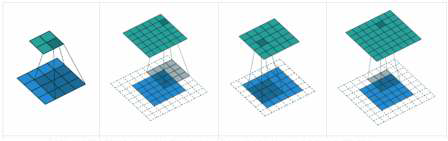


Figure 4.6 - Padding

In the figure 4.6 first image, given - 4x4 input image and a 3x3 filter. There is no padding. After applying it can get the resultant matrix as 2x2, i.e., the output is Downsized.

A filter can move more than one pixel horizontally or vertically. This type of averaging is possible because these steps are called strides.

**4.4.3 POOLING AND FLATTENING**

There are two more layers other than convolution layers called pooling and flattening layers.

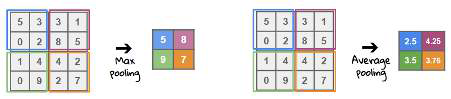


Figure 4.7 - Pooling

The pooling method includes maximum pooling and average pooling. In maxim pooling the individual value within the window is selected. In average pooling the average value within the panel is selected. This removes the unwanted data and fetches more meaningful data. Flattening converts the data into one dimension and gives it as an input for the next layer. The output data is converted into a long vector using a flattening method. One layer that is completely connected to the pixels of the other is the fully-connected layer.

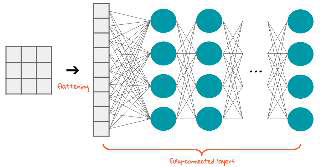


Figure 4.8 - Flattening

The final architecture is given in the figure 4.9 –

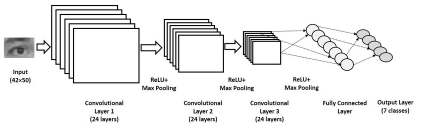


Figure 4.9 – CNN Architecture

**4.4.4 INTRODUCING NON-LINEARITY**

The weighted input of the node can be transformed into an activation node by applying an activation function to it.

Rectified linear activation functions are the most frequently used. This function is also referred to as a rectified linear activation unit. Its input is direct if it is positive. Otherwise, it will output 0. It gives a better performance. If it is used in hidden layers it is called a rectified network. ReLU does element wise operation. It introduces non linearity in CNN.

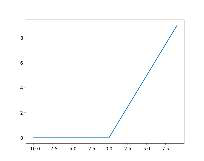


Figure 4.10 – ReLu plot

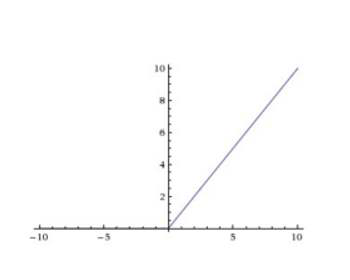


Figure 4.11 – Function graph ReLu

**4.4.5 IMAGE ENHANCEMENT**

Images are enhanced through contrast enhancement, spatial filtering, density slicing, and FCC after they are taken from the original data. Blurring the image not only weeds out details but it also smoothens the edges. After this step, the image is enhanced to a sharper state and edge detection is carried out.

Image enhancement is a key aspect of improving the quality and clarity of images, which in turn, aids in accurate image recognition. Image recognition systems use the visual features and patterns in images to identify objects or scenes. Unfortunately, real-world images can suffer from various issues such as noise, blur, low contrast, and uneven illumination, which can negatively impact image recognition algorithms.

Image enhancement techniques help to pre-process

images and mitigate these issues, resulting in better recognition. Spatial domain methods and frequency domain methods are two broad categories of image enhancement techniques. Spatial domain methods work directly on the pixel values of the image and include techniques like histogram equalization, contrast stretching, and spatial filtering. Frequency domain methods, on the other hand, work on the frequency components of the image and use techniques such as high-pass filtering and deconvolution.

Advanced techniques like image super-resolution can enhance low-resolution images by generating higher-resolution versions for more detailed information, which further aids in recognition. In summary, image enhancement techniques are crucial for improving image quality, reducing noise and blur, and enhancing relevant features for more reliable and effective image recognition systems.

**4.4.6. EDGE DETECTION**

Digital images have localized changes of intensity known as edges. Edges can be defined as a boundary between two disjoint areas surrounded by a string of connected pixels. These kinds of edges exist:

● Horizontal

● Vertical

● Diagonal

By detecting regions of discontinuity, Edge Detection can segment the image. This process consists of several techniques, including image morphology, pattern recognition, and feature extraction.

A texture is used to limit the amount of data in an image, preserve its structural properties, and indicate the look of one region and how it transitions into another. The texture reduces the overall data in an image while preserving its structural properties.Two types of Edge Detection Operators includes Gradient – based operator and Gaussian – based operator.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

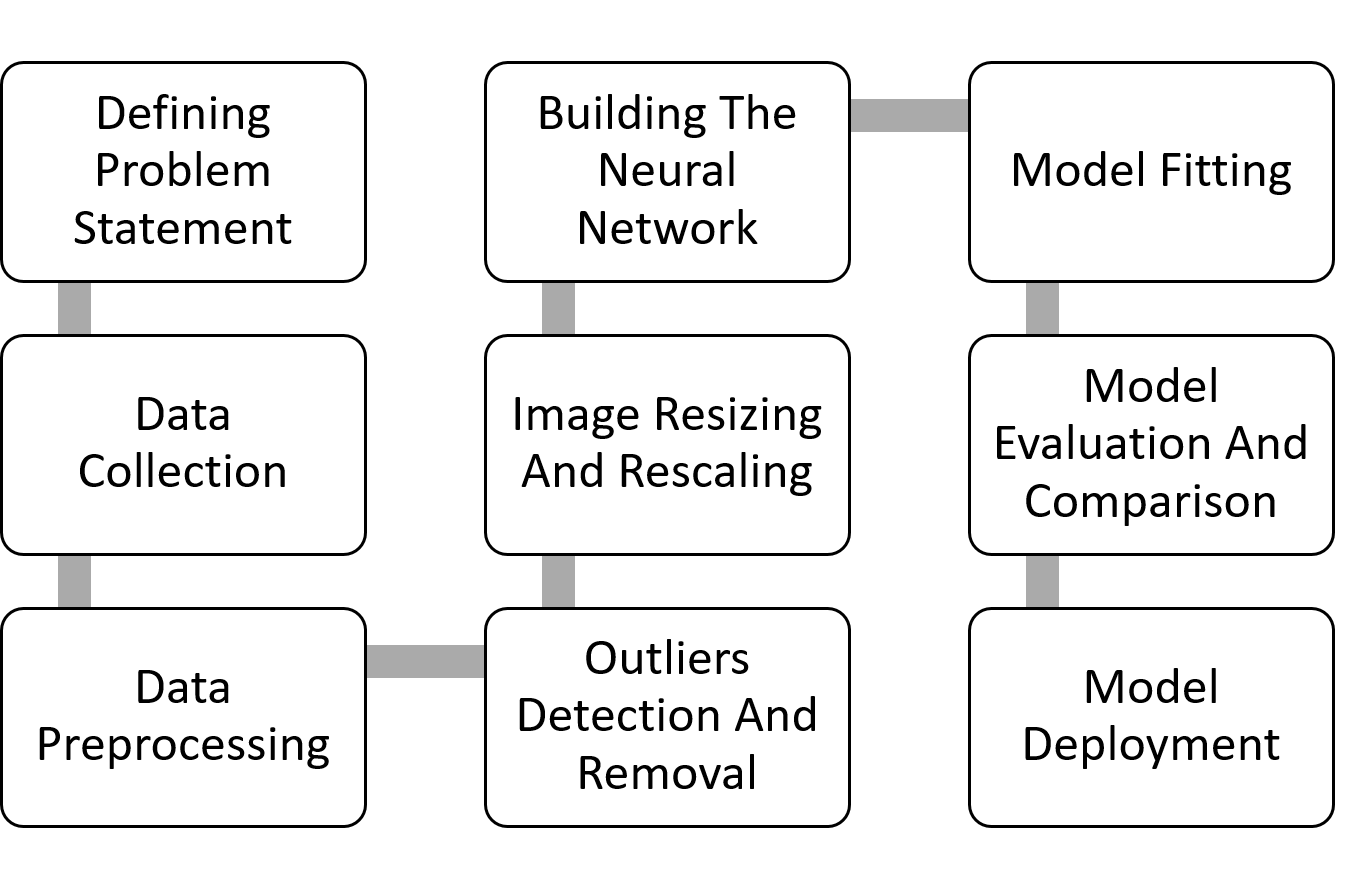
**5.1 SYSTEM FLOW**

The CNN model and MobileNetv2 model are created by training the data sets. The given data set is split. The model is trained, tested and optimized to give the best accuracy. The model is saved.

The model which is more productive and gives more accuracy is implemented into drones.

**5.1.1 FEATURE EXTRACTION**

The workflow of model for classification of suspicious activities and alerting users is shown in the Figure 5.1. Datasets were collected by web scrapping method and then the model is trained using below 2 different models and model deployment were made using openai tool.

  
Fig 5.1 System Workflow

**5.2 SETTING UP THE ENVIRONMENT**

Google colab is used for running the program. Also, for the initial test and run it is sufficient for us to have a google colab account, as it is easily accessible and already has all the important libraries installed on cloud. The dataset needs to be loaded to train the model on the platform at the time of running.

**5.3 MODEL IMPLEMENTATION**

1. The problem statement is defined and the suitable data for the statement is searched.
2. The Dataset is scrapped from the internet and the dataset is classified under 6 classes, i.e. 6 activities of suspicion stored in six separate directories.
3. The dataset is clearly studied, and the preprocessing stages are undertaken.
4. Manually the dataset is analyzed and the images that are considered to be outliers are removed explicitly.
5. All the images are resized into a fixed size of 256 x 256 pixels.
6. The image format of all the images are converted to png, for training purposes.
7. The dataset is loaded, and the neural network is trained with the images.
8. 80% of the images are deployed for training purposes, 10% are separated for validation and 10% are left behind untouched during training for testing purposes.
9. Convolutional layers and Pooling layers are used alternatively with 6 dense layers and ReLu function is used as an activation function in these layers.
10. Softmax activation function is used as an activation function in the output layer.
11. Adam optimizer is used for optimization purposes with accuracy as the metrics, and the model is trained for 50 epochs.
12. The accuracy of the model is measured.
13. The dataset is trained with Mobilenetv2 model with some additional optimizations for 50 epochs and the accuracy is measured.
14. The models are compared, and the model with better accuracy is chosen and deployed.

Convolutional Neural Networks (CNNs) are an effective tool for recognizing items, classes, and categories in images by analyzing their patterns. These networks can also be useful for categorizing audio, time-series, and signal data. The most significant advantage of CNNs is that they can identify important features without any human input. Additionally, CNNs are highly skilled at classifying and identifying visual content. Furthermore, CNNs have the advantage of weight sharing, which is another key benefit.

The pooling layers reduce the size of the hidden layer by merging the outputs of neuron clusters from the previous layer into one neuron in the next layer. On the other hand, the convolution layer extracts features from the data matrix, while the pooling layer only decreases the size of the input matrix.

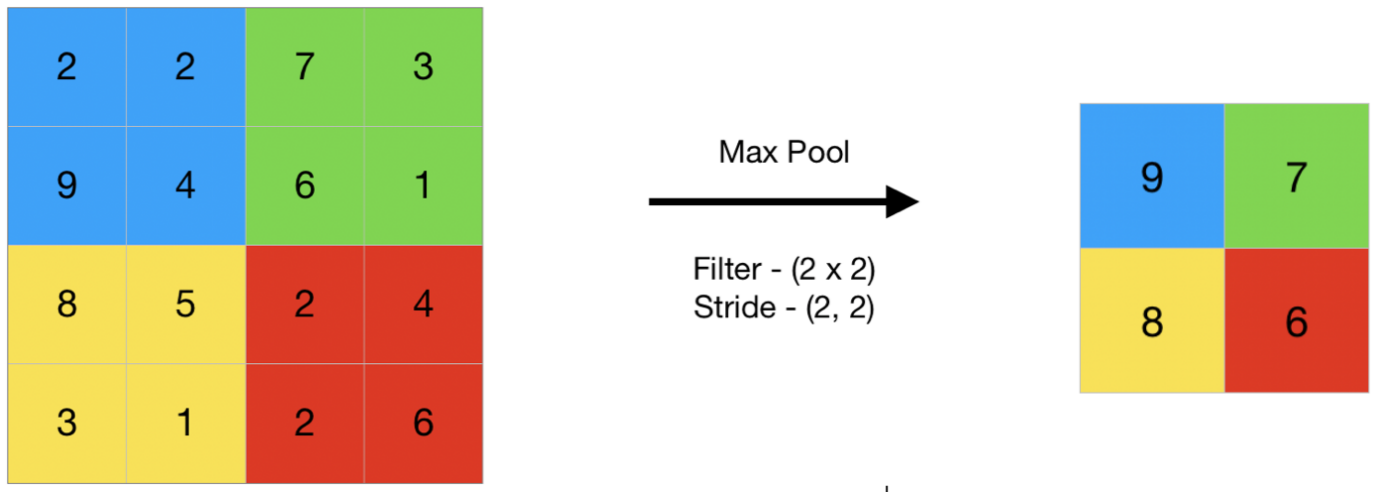


Figure 5.2 – Pooling layer

A neural network can be fine-tuned and its performance improved with the help of an optimizer, which is a function or algorithm. It accomplishes this by modifying the learning rates and weights. The optimization process is crucial because deep learning models are complex and have millions of parameters. It is essential to select the right optimization algorithm that suits your application. One of the most popular algorithms is the Adam optimizer, which has been updated as a benchmark for deep learning. This algorithm is preferred because it is easy to construct, faster, requires less memory, and needs less adjustment than other optimization techniques.

An activation function regulates the level of activation of a neuron. It determines whether the input received by the network from the neuron is useful during the prediction process by employing simple mathematical methods. The softmax activation function converts the unprocessed outputs of the neural network into a probability distribution over the input classes, essentially a vector of probabilities.

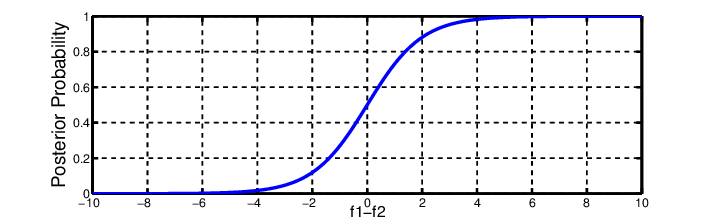


Figure 5.3 – Softmax Activation Function

A rectified linear unit (ReLU) is an activation function that solves the vanishing gradients issue and allows deep learning models to be non-linear. It interprets the feature of its case that is conclusive. This is one of the most popular deep learning activation functions.

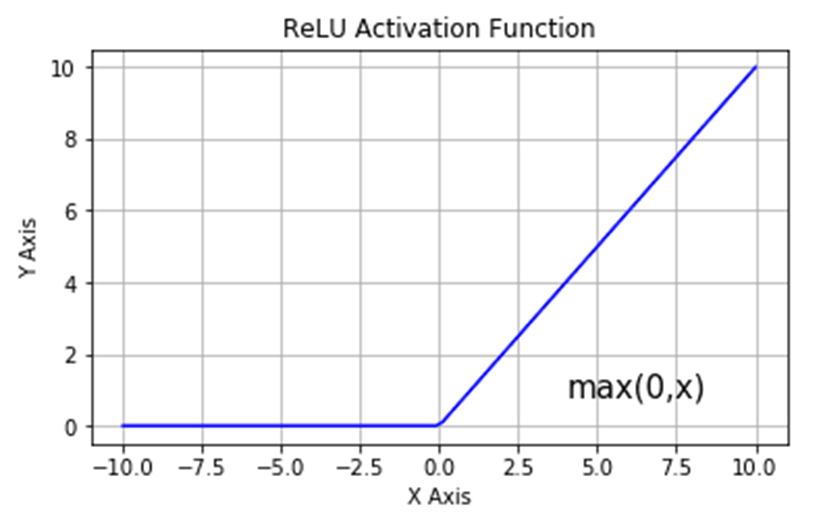


Figure 5.4 – Relu Activation Function

A loss function is a simple tool that measures the effectiveness of your algorithm in replicating the data it is trained on. It provides a higher value if your predictions are completely off the mark, whereas a lower value indicates that your predictions are accurate to a certain extent.

A model's loss value reflects its performance at each optimization cycle. The algorithm's performance is measured using an accurate metric that is easy to understand. The accuracy of a model is frequently evaluated and expressed as a percentage after adjusting its input parameters.

**CHAPTER 6**

**RESULT AND ANALYSIS**

**6.1 TESTING THE NEURAL NETWORK**

System testing is an intriguing anomaly for software engineers. To test the generated programme, the engineer creates a number of test cases. When problems are found, the developer must get past the presumptions that the newly created programme is "correct" and get rid of such presumptions before testing.

There are a number of helpful testing objectives. The programme is run repeatedly during testing to look for flaws. Unknown errors will be uncovered by a successful test scenario. Errors are found in a test scenario with a high likelihood.

The environment is initially examined for compliance with all criteria. After a model has been trained and tested, various datasets of various sizes are used. They are incorporated into the model, which produces outcomes and determines correctness. The environment is initially checked for adherence to all requirements.

Various datasets of different sizes are employed after a model has been trained and tested. They are included in the model, which generates results and establishes accuracy. We assess loss functions in the model in an effort to avoid overfitting while keeping in mind the importance of a non-recurrence loss in each training cycle.

The number of epochs required for a specific dataset is also determined by our test. The best detecting techniques are then employed after testing the components.

**6.2 RESULT ANALYSIS**

Various tests have been conducted on the system, continually improving the accuracy and precision with each test. Several images were also used to check the model's accuracy. It does have included a series of screenshots showing how the model was tested, analysed, and run.

**A snippet of the dataset**

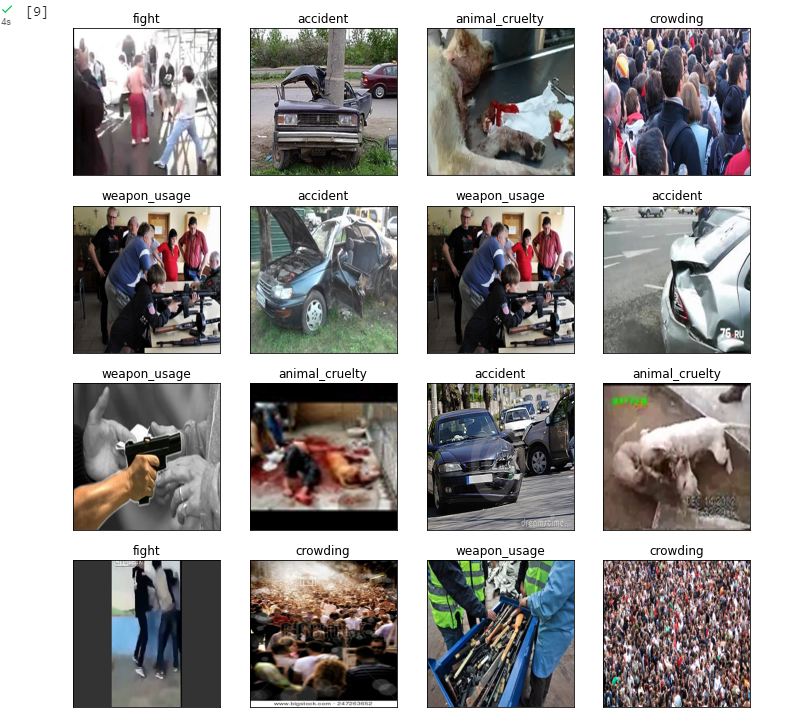
****

Figure 6.1 – dataset snippet

The figure 6.1 depicts the dataset which has 6 different classes.

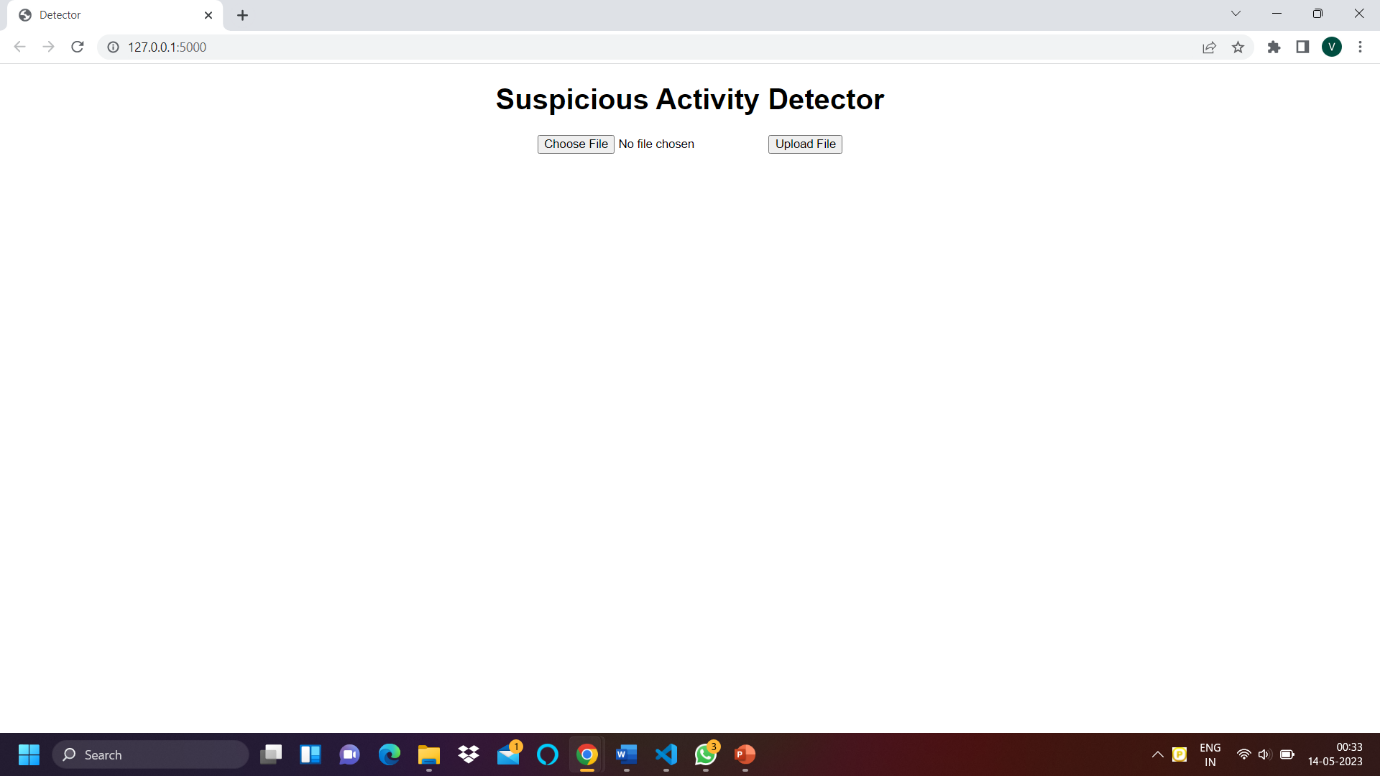


Figure 6.2 – GUI for prediction model

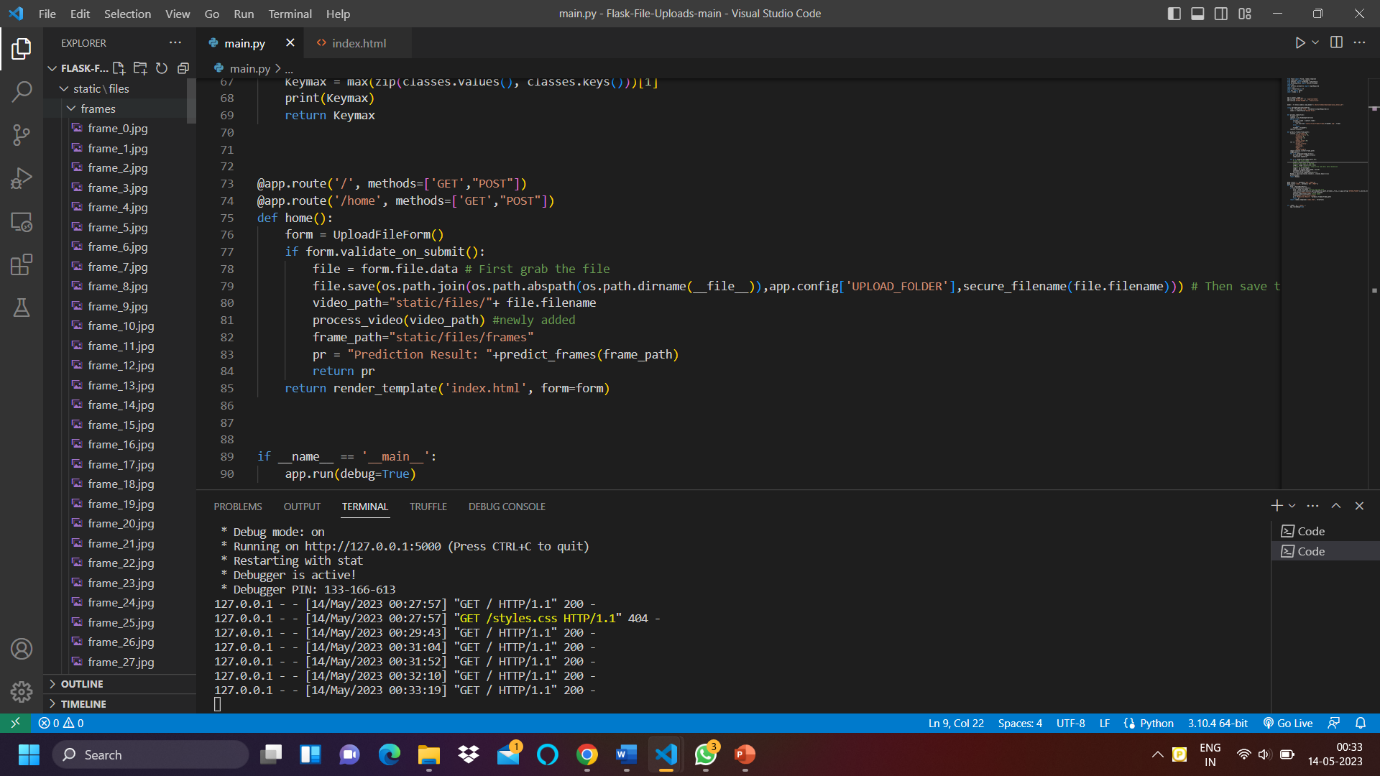


Figure 6.3 – Converting video to frames

The Figure 6.2 represent the REST api connection using FLASK and it has upload feature to predict the video input and Figure 6.3 shows the video converted into frames of jpg format.

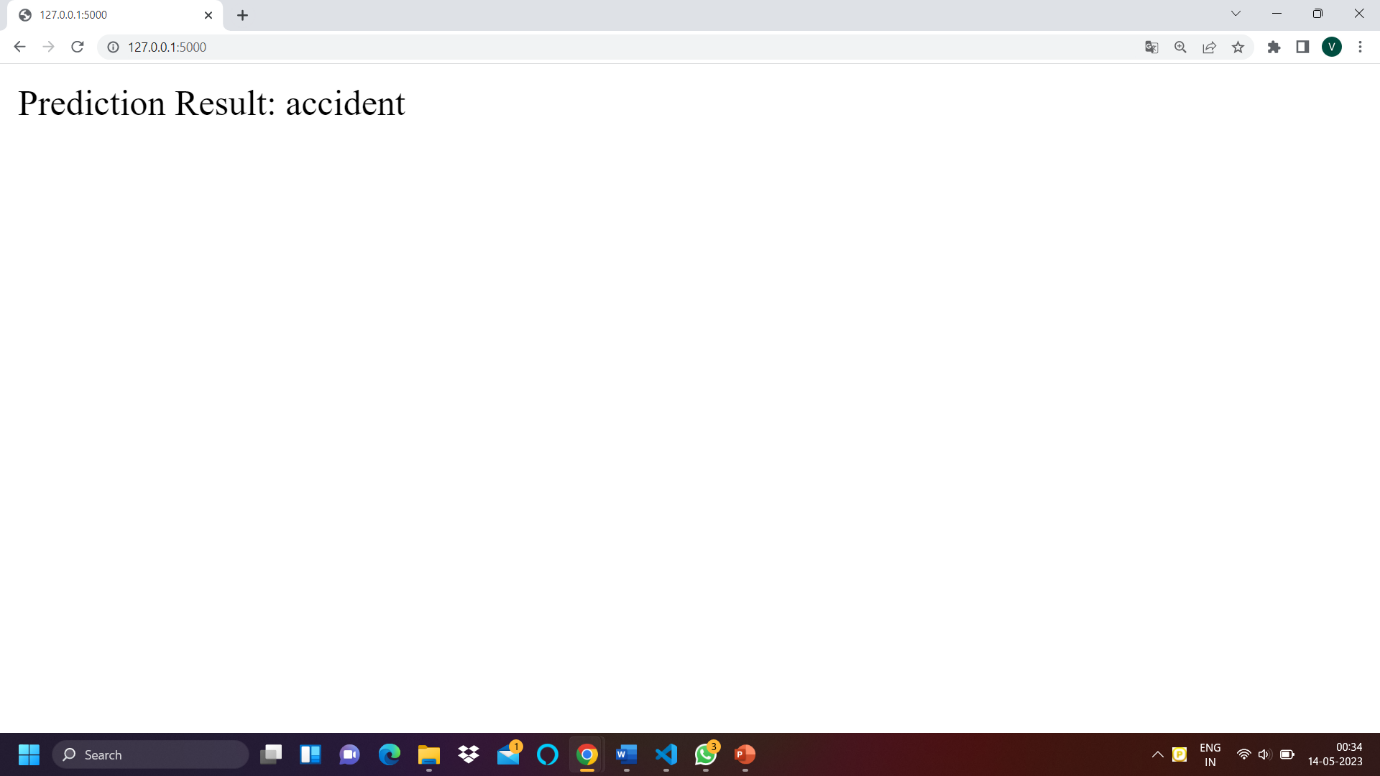


Figure 6.4 – Result of Prediction

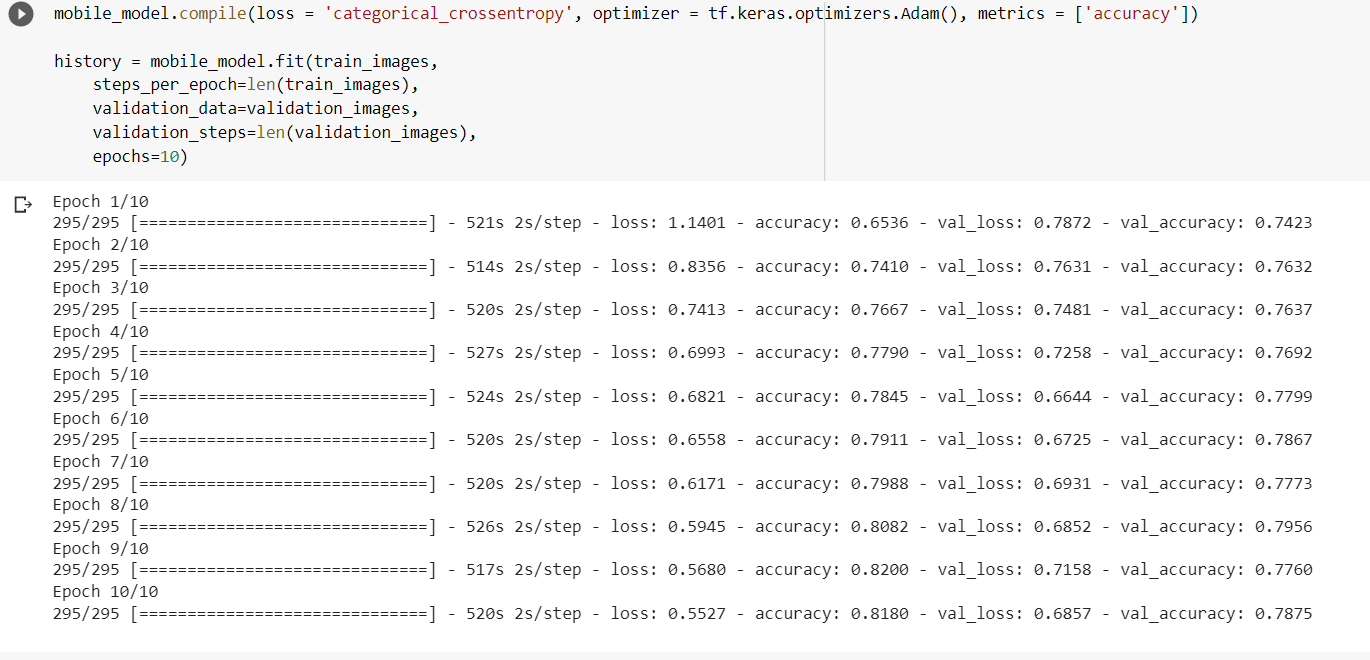


Figure 6.5 – Model Training

The Figure 6.4 depicts the result of prediction after successful upload of video and Figure 6.5 shows the training of the model for 10 epochs.



Figure 6.6 –Saving frames in jpg format from video

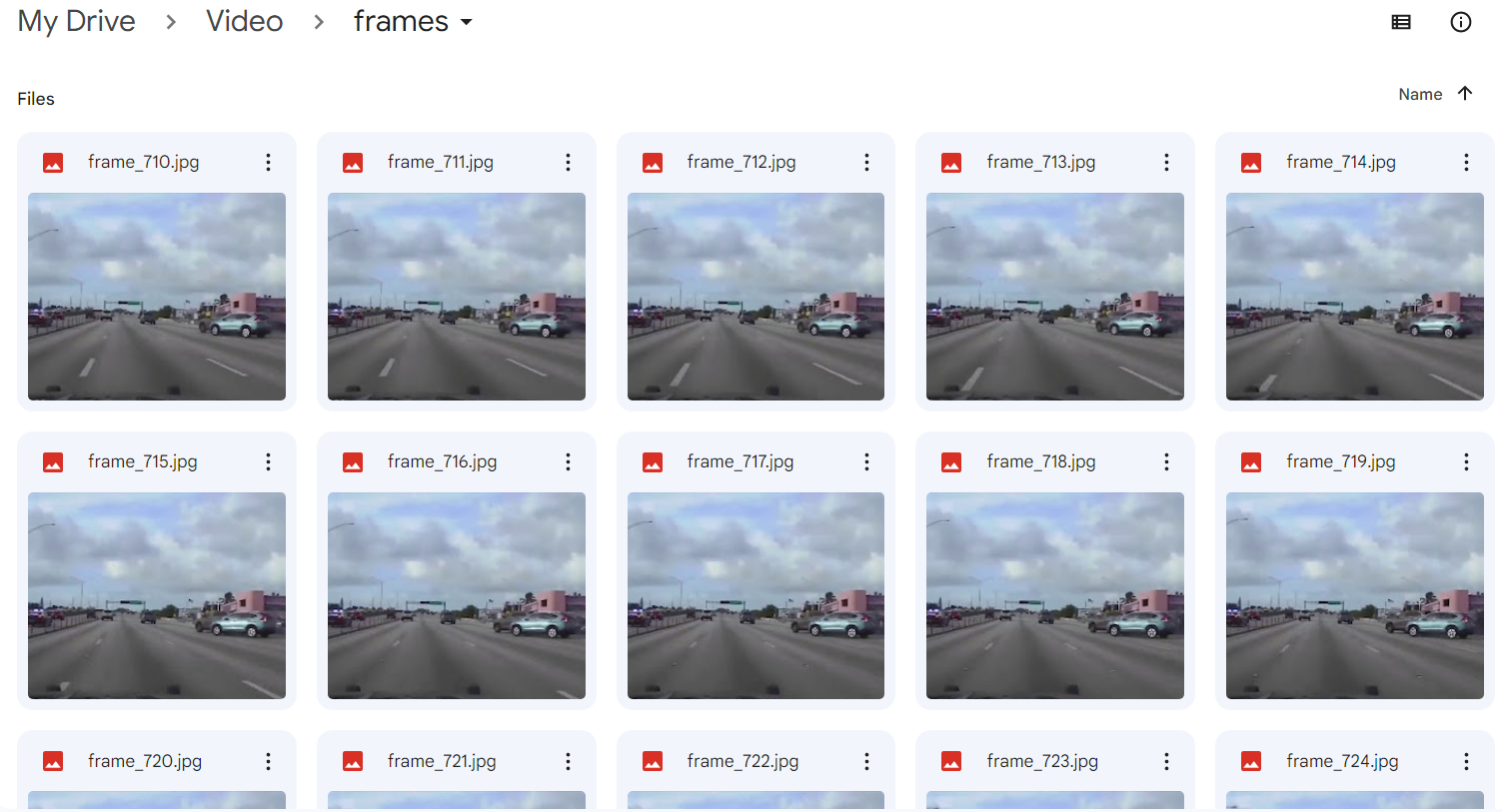


Figure 6.7 – Saved images

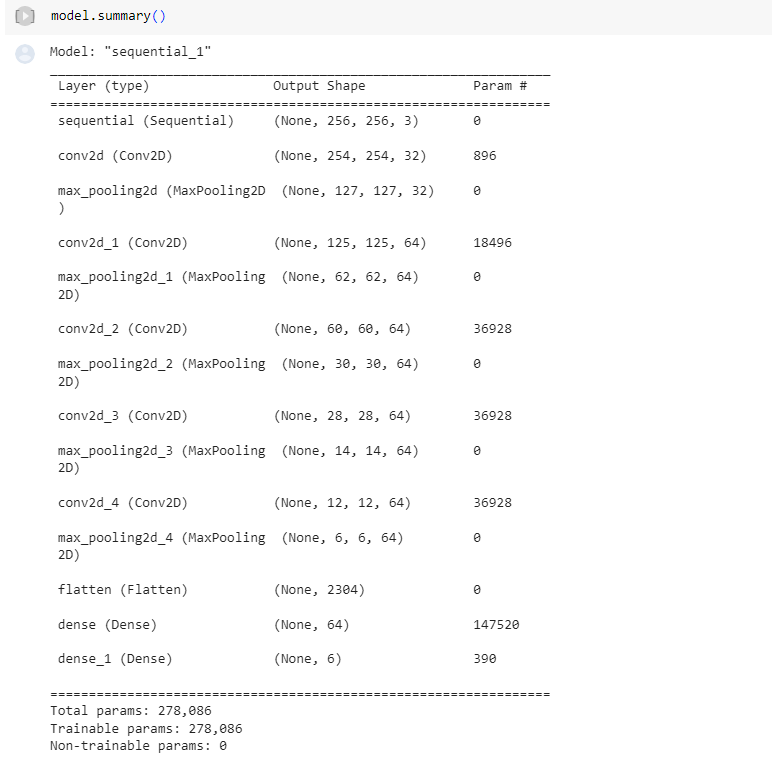


Figure 6.8 – Model Architecture

The Figure 6.6 and Figure 6.7 shows the saved frames which are made from video input and Figure 6.8 shows the TensorFlow based keras model.

The below figures depicts the accuracy and loss functions of the two models. Here is the table which includes the comparison of different models on the test data for the performance metrics.

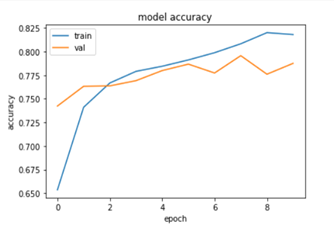


Figure 6.9 - Accuracy function for keras based CNN

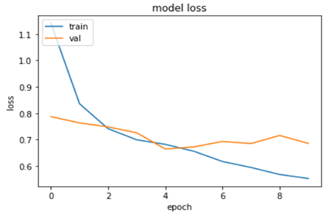


Figure 6.10 - Loss function for keras based CNN



Figure 6.11 - Accuracy function for mobilenetV2 based CNN

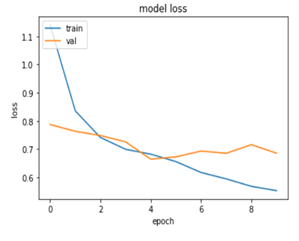


Figure 6.12 - Loss function for mobilenetV2 based CNN

Table 6.1 - Result of model

|  |  |
| --- | --- |
| **MODEL** | **MODEL ACCURACY** |
| MobileNetv2 with 20 epochs | 0.36 |
| MobileNetv2 with 50 epochs | 0.52 |
| CNN with 20 epochs | 0.78 |
| CNN with 50 epochs | 0.95 |

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**7.1 CONCLUSION**

The research discusses a method for detecting suspicious activity using drones and the Convolutional Neural Network model for feature extraction and network for action classification. There are several techniques available for detecting suspicious activity, each with its own advantages and limitations. By using multiple techniques, it is possible to enhance the accuracy of detection. However, deep learning-based models, specifically the Convolution Neural Network model, are more efficient and precise in classification and also improve accuracy while reducing false positives. The proposed classifier is based on CNNs, which compare the input and training data to achieve the best results.

Furthermore, this particular method has the capacity to not only anticipate but also make predictions regarding an expanded range of dubious behaviours that may occur within various public or private locations. The versatility of this model is such that it can be applied across a wide array of scenarios, providing training that specifically aligns with the unique suspicious activities associated with each particular circumstance. By implementing the process of identifying the suspicious individual in conjunction with the observed suspicious activities, significant enhancements can be made to the overall performance and effectiveness of the model.

**7.2 FUTURE ENHANCEMENT**

This project can be further improved by approaching the problem from another direction, increasing the accuracy and possibly proposing whether the detected Suspicious activity is effective. The proposed model identifies 6 different classes as of now, it can be improved by targeting a greater number of suspicious activities.

Currently our existing dataset has some limited data’s, an increased number of images can be incorporated, particularly by extracting images from the drone recordings capturing instances of suspicious activity. However, obtaining such footage is currently a challenge for students. Nevertheless, with the backing and support of the administration, there is a strong possibility of getting drone footage depicting criminal activities that have transpired in the recent past. This collaborative effort would undoubtedly contribute significantly to the advancement and refinement of the model.

**REFERENCES**

[1] Dwivedi, N., Singh, D.K, et al. A novel approach for suspicious activity detection with deep learning. Multimed Tools Appl (2023). https://doi.org/10.1007/s11042-023-14445-7[2] Gupta, A., Tickoo, A., et al. (2023). Unusual Activity Detection Using Machine Learning. In Proceedings of International Conference on Recent Trends in Computing. Lecture Notes in Networks and Systems, vol 600. Springer, Singapore. https://doi.org/10.1007/978-981-19-8825-7\_47[3] Bagane, P., Krishna, et al. (2023). Unsupervised Machine Learning for Unusual Crowd Activity Detection. Proceedings of International Conference on Recent Trends in Computing. Lecture Notes in Networks and Systems, vol 600. Springer, Singapore. https://doi.org/10.1007/978-981-19-8825-7\_70[4] Dhanush Kumar, A., Shushruth Reddy, et al. (2022). Abnormal Activity Detection Using Deep Learning. Intelligent Sustainable Systems. Lecture Notes in Networks and Systems, vol 333. Springer, Singapore. https://doi.org/10.1007/978-981-16-6309-3\_63[5] Sharma, M., Garg, D.K. (2022). Human Activity Detection-Based Upon CNN with Pruning and Edge DetectionAdvances in Intelligent Computing and Communication. https://doi.org/10.1007/978-981-19-0825-5\_2[6] Prabha, B., Manivannan, et al. (2022). Human Abnormal Activity Detection in the ATM Surveillance Video. Evolution in Signal Processing and Telecommunication Networks. Lecture Notes in Electrical Engineering, vol 839. Springer, Singapore. https://doi.org/10.1007/978-981-16-8554-5\_5[7] Visuwasam, L.M.M., Kalpana, et al. (2022).Proceedings of the International Conference on Cognitive and Intelligent Computing. Cognitive Science and Technology. Springer, Singapore. https://doi.org/10.1007/978-981-19-2350-0\_80[8] Agarwal, M., Parashar,et al. (2022). Suspicious Activity Detection in Surveillance Applications Using Slow-Fast Convolutional Neural Network.Advances in Data Computing, Communication and Security. Lecture Notes on Data Engineering and Communications Technologies, vol 106. Springer, Singapore. https://doi.org/10.1007/978-981-16-8403-6\_59[9] de Oliveira, W.G., Filho, et al. (2022). Driver Behavior Analysis: Abnormal Driving Detection Using MLP Classifier Applied to Outdoor Camera ImagesIntelligent Systems Design and Applications. ISDA 2021. Lecture Notes in Networks and Systems, vol 418. Springer, Cham. https://doi.org/10.1007/978-3-030-96308-8\_106

[10] Kaur, G., et al (2022). Violence Detection in Videos Using Deep Learning: A Survey Advances in Information Communication Technology and Computing. Lecture Notes in Networks and Systems, vol 392. Springer, Singapore. https://doi.org/10.1007/978-981-19-0619-0\_15[11] Pawade, A., Anjaria, R., et al. (2021). Suspicious Activity Detection for Security Cameras. Applications of Advanced Computing in Systems. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-33-4862-2\_22[12] Thombare, P., Gond, et al (2021). Artificial Intelligence for Low Level Suspicious Activity Detection. Applications of Advanced Computing in Systems. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-33-4862-2\_23[13] Pawar, K., Attar, V. (2021). Automated Surveillance Model for Video-Based Anomalous Activity Detection Using Deep Learning Architecture. Innovations in Computational Intelligence and Computer Vision. Advances in Intelligent Systems and Computing, vol 1189. Springer, Singapore. https://doi.org/10.1007/978-981-15-6067-5\_36[14] Saidon, M.S., Mustafa et al. (2021). Automatic People Counting System Using Aerial Image Captured by Drone for Event Management. Intelligent Manufacturing and Mechatronics. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-16-0866-7\_4[15] Vallathan, G., John, A., Thirumalai, C. et al. Suspicious activity detection using deep learning in secure assisted living IoT environments. Supercomputer 77, 3242–3260 (2021). https://doi.org/10.1007/s11227-020-03387-8[16] Gorave, A., Misra, et al. (2020). Suspicious Activity Detection Using Live Video Analysis. Proceeding of International Conference on Computational Science and Applications. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-15-0790-8\_21[17] Castellano, G., Castiello, C., et al. (2020). Multi-view Convolutional Network for Crowd Counting in Drone-Captured Images. Computer Vision – ECCV 2020 Workshops. ECCV 2020. Lecture Notes in Computer Science(), vol 12538. Springer, Cham. https://doi.org/10.1007/978-3-030-66823-5\_35[18] Iqbal, N., Saad Missen, et al. (2019). On Video Based Human Abnormal Activity Detection with Histogram of Oriented Gradients. Handbook of Multimedia Information Security: Techniques and Applications. Springer, Cham. https://doi.org/10.1007/978-3-030-15887-3\_21

[19] Kalech, M., Shlomo, A., et al. (2019). Temporal Pattern-Based Malicious Activity Detection in SCADA Systems (Brief Announcement). Cyber Security Cryptography and Machine Learning. CSCML 2019. Lecture Notes in Computer Science(), vol 11527. Springer, Cham. https://doi.org/10.1007/978-3-030-20951-3\_26[20] Fraś, M., Bednarz, M. (2017). Simple Rule-Based Human Activity Detection with Use of Mobile Phone Sensors. Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part II. Advances in Intelligent Systems and Computing, vol 522. Springer, Cham. https://doi.org/10.1007/978-3-319-46586-9\_4[21] Senthilkumar, T., Archana , et al. (2016). Suspicious Human Activity Detection in Classroom Examination. Computational Intelligence, Cyber Security and Computational Models. Advances in Intelligent Systems and Computing, vol 412. Springer, Singapore. https://doi.org/10.1007/978-981-10-0251-9\_11[22] Shin, H., Choi, K., et al. (2016). Security Analysis of FHSS-type Drone Controller.Information Security Applications. WISA 2015. Lecture Notes in Computer Science(), vol 9503. Springer, Cham. https://doi.org/10.1007/978-3-319-31875-2\_20[23] Zhao, Y., Qiao, et al. (2015). Abnormal Activity Detection Using Spatio-Temporal Feature and Laplacian Sparse Representation. Neural Information Processing. ICONIP 2015. Lecture Notes in Computer Science(), vol 9492. Springer, Cham. https://doi.org/10.1007/978-3-319-26561-2\_49[24] Yussiff, AL., Yong, SP.,et al. (2014). Detecting People Using Histogram of Oriented Gradients: A Step towards Abnormal Human Activity Detection. Advances in Computer Science and its Applications. Lecture Notes in Electrical Engineering, vol 279. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41674-3\_159[25] Dimoulas, C.A., Avdelidis, et al. Joint Wavelet Video Denoising and Motion Activity Detection in Multimodal Human Activity Analysis: Application to Video-Assisted Bioacoustic/Psychophysiological Monitoring. EURASIP J. Adv. Signal Process. 2008, 792028 (2007). https://doi.org/10.1155/2008/792028

[26] Mateen Buttar, Ahmed & Bano, Mahnoor & Azeem Akbar, et al. (2023). Toward trustworthy human suspicious activity detection from surveillance videos using deep learning. Soft Computing. 1-13. 10.1007/s00500-023-07971-x.

[27] Leela S, K V Sai Likhita, et al. Suspicious Human Activity Recognition and Alarming System. Ijraset Journal For Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2022.44900

[28]  Sumon Ghosh, Prasham Shah, et al. Suspicious Activity Detection. Ijraset Journal For Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2022.47186

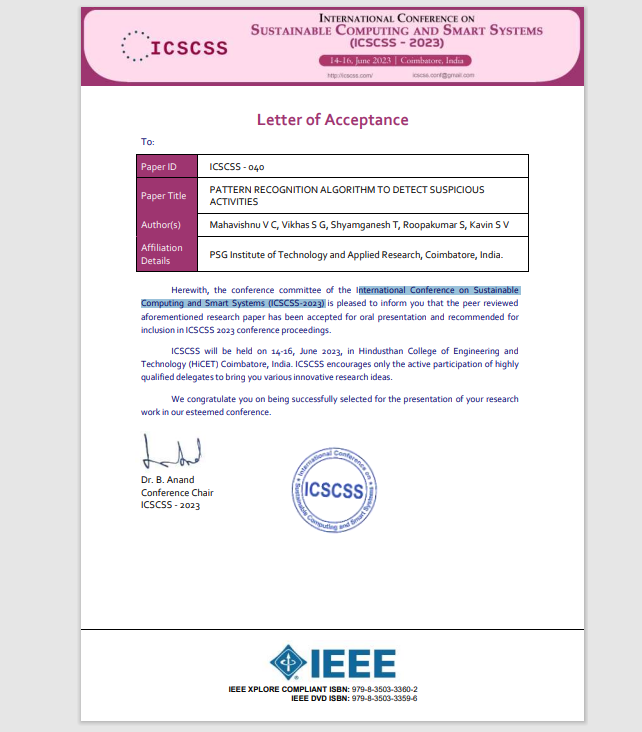
[29]  Deshmukh, Shubham & Fernandes, et al. (2022). Suspicious and Anomaly Detection. 10.48550/arXiv.2209.03576.

**PUBLICATION DETAILS**

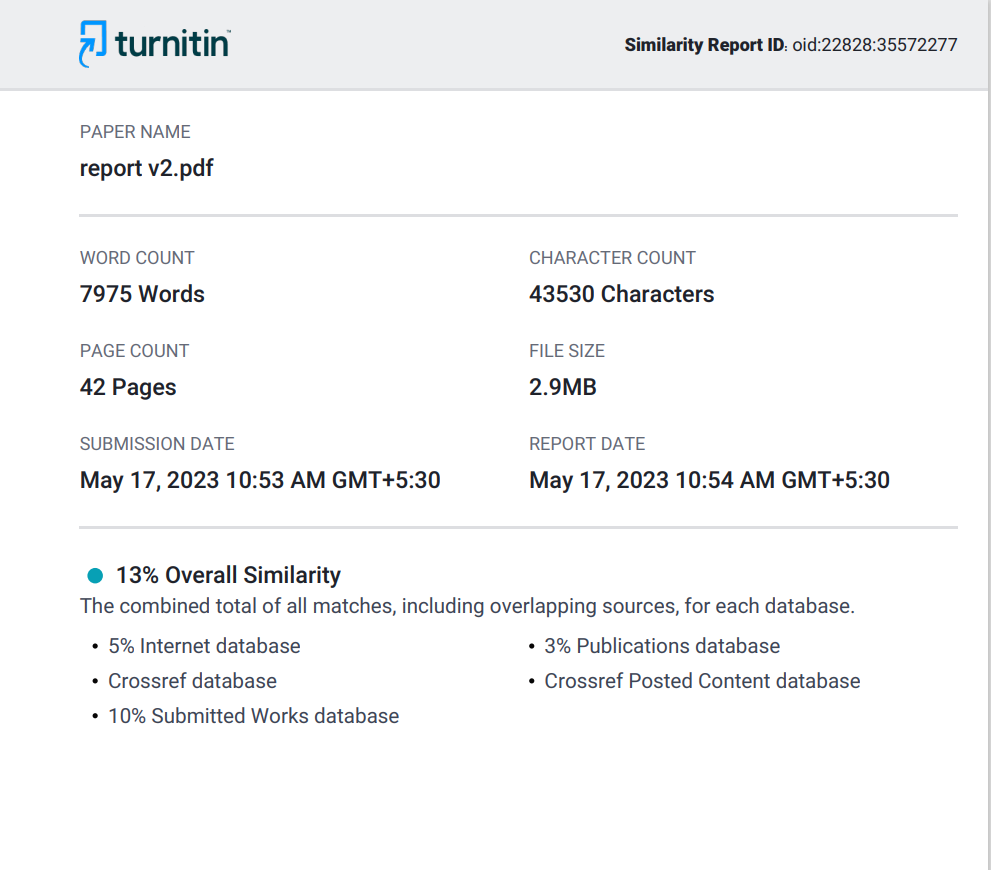
**Authors:** Dr.Mahavishnu V C, Kavin S V, Roopakumar S, Shyamganesh T, Vikhas S G.

PATTERN RECOGNITION ALGORITHM TO DETECT SUSPICIOUS ACTIVITIES.

International Conference on Sustainable Computing and Smart Systems (ICSCSS-2023).



**PLAGIARISM REPORT**



**APPENDIX**

#MobilenetV2 model

import os

import warnings

warnings.filterwarnings("ignore")

from google.colab import drive

drive.mount('/content/drive')

%cd '/content/drive/My Drive'

#Imports

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers,models

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, Flatten, MaxPooling2D, Dense, Dropout, GlobalAveragePooling2D

from tensorflow.keras import optimizers, losses

import seaborn as sns

import matplotlib.pyplot as plt

# System libraries

from pathlib import Path

import os.path

# Metrics

from sklearn.metrics import classification\_report, confusion\_matrix

import itertools

image\_dir = Path(data)

# Get filepaths and labels

filepaths = list(image\_dir.glob(r'\*\*/\*.JPG')) + list(image\_dir.glob(r'\*\*/\*.jpg')) + list(image\_dir.glob(r'\*\*/\*.png')) + list(image\_dir.glob(r'\*\*/\*.PNG'))

labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)

labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels

image\_df = pd.concat([filepaths, labels], axis=1)

import PIL

from pathlib import Path

from PIL import UnidentifiedImageError

path = Path("project").rglob("\*.png")

for img\_p in path:

    try:

        img = PIL.Image.open(img\_p)

    except PIL.UnidentifiedImageError:

            print(img\_p)

image\_df

| **Filepath** | **Label** |
| --- | --- |
| **0** | project/weapon\_usage/undefined-1575555697-924-... | weapon\_usage |
| **1** | project/weapon\_usage/73d96b1b494d8e0563255b1f9... | weapon\_usage |
| **2** | project/weapon\_usage/tim.png | weapon\_usage |
| **3** | project/weapon\_usage/B0yn-hiqtcan8730246 (1).png | weapon\_usage |
| **4** | project/weapon\_usage/f1fd2da9063bf2a9bdcaa313a... | weapon\_usage |
| **...** | ... | ... |
| **630** | project/accident/mqdefault (6).png | accident |
| **631** | project/accident/367x232.png | accident |
| **632** | project/accident/HYiDlz38MYbVJdnCAiILOfFy8fzlw... | accident |
| **633** | project/accident/5852564.png | accident |
| **634** | project/accident/380-333x1000.png | accident |

635 rows × 2 columns

# Display 16 picture of the dataset with their labels

random\_index = np.random.randint(0, len(image\_df), 16)

fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(10, 10),

                        subplot\_kw={'xticks': [], 'yticks': []})

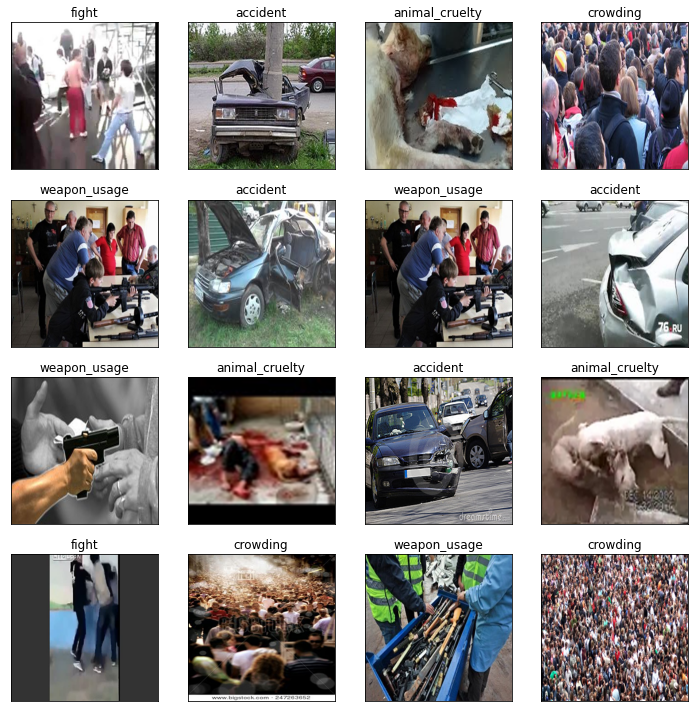
for i, ax in enumerate(axes.flat):

    ax.imshow(plt.imread(image\_df.Filepath[random\_index[i]]))

    ax.set\_title(image\_df.Label[random\_index[i]])

plt.tight\_layout()

plt.show()



train\_datagen = ImageDataGenerator(rescale=1./255,rotation\_range = 40, width\_shift\_range = 0.2, height\_shift\_range = 0.2,

                                  shear\_range = 0.2, zoom\_range = 0.2, horizontal\_flip = True, fill\_mode = 'nearest',

    validation\_split=0.2) # set validation split

train\_images = train\_datagen.flow\_from\_directory(

    data,

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    subset='training') # set as training data

validation\_images = train\_datagen.flow\_from\_directory(

    data , # same directory as training data

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    subset='validation') # set as validation data

# Load the pretained model

mobile\_model = Sequential()

pretrained\_model = tf.keras.applications.MobileNetV2(

    input\_shape=(224, 224, 3),

    include\_top=False,

    weights='imagenet',

    pooling='avg'

)

pretrained\_model.trainable = False

mobile\_model.add(pretrained\_model)

mobile\_model.add(Flatten())

mobile\_model.add(Dense(512, activation='relu'))

mobile\_model.add(Dropout(0.2))

mobile\_model.add(Dense(6, activation='softmax'))

mobile\_model.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

mobilenetv2\_1.00\_224 (Funct (None, 1280) 2257984

ional)

flatten (Flatten) (None, 1280) 0

dense (Dense) (None, 512) 655872

dropout (Dropout) (None, 512) 0

dense\_1 (Dense) (None, 10) 5130

flatten\_1 (Flatten) (None, 10) 0

dense\_2 (Dense) (None, 512) 5632

dropout\_1 (Dropout) (None, 512) 0

dense\_3 (Dense) (None, 6) 3078

=================================================================

Total params: 2,927,696

Trainable params: 669,712

Non-trainable params: 2,257,984

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

mobile\_model.compile(

    optimizer='adam',

    loss=tf.keras.losses.CategoricalCrossentropy(from\_logits=False),

    metrics=['accuracy']

)

history = mobile\_model.fit(train\_images,batch\_size=32,validation\_data=validation\_images,epochs=25,verbose=1)

Epoch 1/25

16/16 [==============================] - 52s 3s/step - loss: 1.7296 - accuracy: 0.3131 - val\_loss: 1.6395 - val\_accuracy: 0.4355

Epoch 2/25

16/16 [==============================] - 41s 3s/step - loss: 1.5532 - accuracy: 0.4716 - val\_loss: 1.4514 - val\_accuracy: 0.5081

Epoch 3/25

16/16 [==============================] - 37s 2s/step - loss: 1.3786 - accuracy: 0.5049 - val\_loss: 1.3038 - val\_accuracy: 0.4919

Epoch 4/25

16/16 [==============================] - 40s 3s/step - loss: 1.2272 - accuracy: 0.4892 - val\_loss: 1.2106 - val\_accuracy: 0.5000

Epoch 5/25

16/16 [==============================] - 40s 3s/step - loss: 1.1409 - accuracy: 0.4658 - val\_loss: 1.1142 - val\_accuracy: 0.5000

Epoch 6/25

16/16 [==============================] - 40s 3s/step - loss: 1.1019 - accuracy: 0.4990 - val\_loss: 1.1174 - val\_accuracy: 0.5000

Epoch 7/25

16/16 [==============================] - 40s 3s/step - loss: 1.0927 - accuracy: 0.4736 - val\_loss: 1.0948 - val\_accuracy: 0.5081

Epoch 8/25

16/16 [==============================] - 41s 2s/step - loss: 1.1229 - accuracy: 0.4932 - val\_loss: 1.0797 - val\_accuracy: 0.4919

Epoch 9/25

16/16 [==============================] - 39s 2s/step - loss: 1.0519 - accuracy: 0.5068 - val\_loss: 1.0988 - val\_accuracy: 0.5000

Epoch 10/25

16/16 [==============================] - 39s 2s/step - loss: 1.0548 - accuracy: 0.5108 - val\_loss: 1.0867 - val\_accuracy: 0.5000

Epoch 11/25

16/16 [==============================] - 40s 3s/step - loss: 1.0566 - accuracy: 0.5068 - val\_loss: 1.1503 - val\_accuracy: 0.4758

Epoch 12/25

16/16 [==============================] - 39s 2s/step - loss: 1.0995 - accuracy: 0.4932 - val\_loss: 1.0991 - val\_accuracy: 0.4839

Epoch 13/25

16/16 [==============================] - 38s 2s/step - loss: 1.0464 - accuracy: 0.5186 - val\_loss: 1.0680 - val\_accuracy: 0.5081

Epoch 14/25

16/16 [==============================] - 39s 2s/step - loss: 1.0306 - accuracy: 0.5127 - val\_loss: 1.1227 - val\_accuracy: 0.4839

Epoch 15/25

16/16 [==============================] - 43s 3s/step - loss: 1.0327 - accuracy: 0.4971 - val\_loss: 1.1442 - val\_accuracy: 0.4758

Epoch 16/25

16/16 [==============================] - 39s 2s/step - loss: 1.0194 - accuracy: 0.5108 - val\_loss: 1.0699 - val\_accuracy: 0.5000

Epoch 17/25

16/16 [==============================] - 35s 2s/step - loss: 1.0494 - accuracy: 0.5068 - val\_loss: 1.0834 - val\_accuracy: 0.5000

Epoch 18/25

16/16 [==============================] - 39s 2s/step - loss: 1.0396 - accuracy: 0.5010 - val\_loss: 1.1220 - val\_accuracy: 0.4839

Epoch 19/25

16/16 [==============================] - 36s 2s/step - loss: 1.0122 - accuracy: 0.5088 - val\_loss: 1.1087 - val\_accuracy: 0.5081

Epoch 20/25

16/16 [==============================] - 39s 2s/step - loss: 1.0614 - accuracy: 0.5010 - val\_loss: 1.1243 - val\_accuracy: 0.4919

Epoch 21/25

16/16 [==============================] - 39s 2s/step - loss: 1.0189 - accuracy: 0.5284 - val\_loss: 1.0966 - val\_accuracy: 0.4919

Epoch 22/25

16/16 [==============================] - 39s 2s/step - loss: 1.0430 - accuracy: 0.4912 - val\_loss: 1.1326 - val\_accuracy: 0.4839

Epoch 23/25

16/16 [==============================] - 39s 2s/step - loss: 1.1370 - accuracy: 0.5068 - val\_loss: 1.1044 - val\_accuracy: 0.4919

Epoch 24/25

16/16 [==============================] - 39s 2s/step - loss: 1.0482 - accuracy: 0.4814 - val\_loss: 1.0918 - val\_accuracy: 0.4919

Epoch 25/25

16/16 [==============================] - 36s 2s/step - loss: 1.0260 - accuracy: 0.4971 - val\_loss: 1.1221 - val\_accuracy: 0.5000

#Accuracy

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

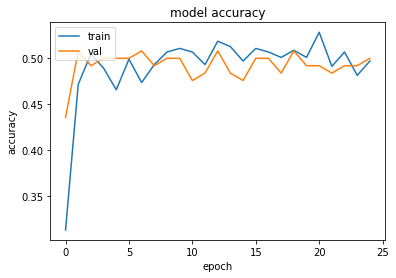
plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()



# loss

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

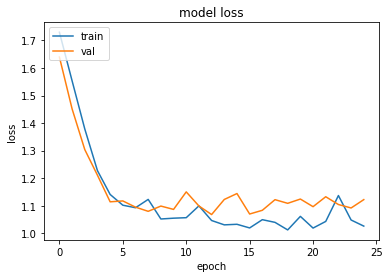
plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()



mobile\_model.save("mobile\_unsus.h5")

#keras based model

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras import models,layers

from google.colab import drive

drive.mount('/content/drive')

%cd '/content/drive/My Drive'

ds = tf.keras.preprocessing.image\_dataset\_from\_directory("project",shuffle=True,image\_size=(256,256),batch\_size=32)

Found 635 files belonging to 6 classes.

classes = ds.class\_names

classes

['accident', 'animal\_cruelty', 'crowding', 'explosion', 'fight', 'weapon\_usage']

train = train.cache().prefetch(buffer\_size=2).shuffle(10)

vali = vali.cache().prefetch(buffer\_size=2).shuffle(10)

test = test.cache().prefetch(buffer\_size=2).shuffle(10)

prepro = tf.keras.Sequential([

    layers.experimental.preprocessing.Resizing(256,256),

    layers.experimental.preprocessing.Rescaling(1.0/255)

])

inpu = (32,256,256,3)

model = models.Sequential([

    prepro,

    layers.Conv2D(filters=32,kernel\_size=(3,3),activation="relu",input\_shape=inpu),

    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64,kernel\_size=(3,3),activation="relu"),

    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64,kernel\_size=(3,3),activation="relu"),

    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64,kernel\_size=(3,3),activation="relu"),

    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64,kernel\_size=(3,3),activation="relu"),

    layers.MaxPooling2D((2,2)),

    layers.Flatten(),

    layers.Dense(64,activation='relu'),

    layers.Dense(6,activation="softmax")

])

model.build(input\_shape=inpu)

model.compile(

    optimizer='adam',

    loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=False),

    metrics=['accuracy']

)

history=model.fit(train,batch\_size=32,validation\_data=vali,epochs=10,verbose=1)

Epoch 1/10

16/16 [==============================] - 71s 4s/step - loss: 0.0226 - accuracy: 0.9961 - val\_loss: 0.5427 - val\_accuracy: 0.8983

Epoch 2/10

16/16 [==============================] - 72s 4s/step - loss: 0.0143 - accuracy: 0.9961 - val\_loss: 0.4920 - val\_accuracy: 0.9322

Epoch 3/10

16/16 [==============================] - 71s 4s/step - loss: 0.0073 - accuracy: 1.0000 - val\_loss: 0.6232 - val\_accuracy: 0.9322

Epoch 4/10

16/16 [==============================] - 71s 4s/step - loss: 0.0037 - accuracy: 1.0000 - val\_loss: 0.6771 - val\_accuracy: 0.9153

Epoch 5/10

16/16 [==============================] - 71s 5s/step - loss: 0.0020 - accuracy: 1.0000 - val\_loss: 0.6106 - val\_accuracy: 0.9153

Epoch 6/10

16/16 [==============================] - 71s 4s/step - loss: 0.0012 - accuracy: 1.0000 - val\_loss: 0.6408 - val\_accuracy: 0.8983

Epoch 7/10

16/16 [==============================] - 73s 5s/step - loss: 6.8284e-04 - accuracy: 1.0000 - val\_loss: 0.6830 - val\_accuracy: 0.8983

Epoch 8/10

16/16 [==============================] - 71s 4s/step - loss: 5.1086e-04 - accuracy: 1.0000 - val\_loss: 0.6852 - val\_accuracy: 0.9153

Epoch 9/10

16/16 [==============================] - 72s 4s/step - loss: 4.5301e-04 - accuracy: 1.0000 - val\_loss: 0.6953 - val\_accuracy: 0.9153

Epoch 10/10

16/16 [==============================] - 72s 4s/step - loss: 4.0846e-04 - accuracy: 1.0000 - val\_loss: 0.7115 - val\_accuracy: 0.8983

model.summary()

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

sequential (Sequential) (None, 256, 256, 3) 0

conv2d (Conv2D) (None, 254, 254, 32) 896

max\_pooling2d (MaxPooling2D (None, 127, 127, 32) 0

)

conv2d\_1 (Conv2D) (None, 125, 125, 64) 18496

max\_pooling2d\_1 (MaxPooling (None, 62, 62, 64) 0

2D)

conv2d\_2 (Conv2D) (None, 60, 60, 64) 36928

max\_pooling2d\_2 (MaxPooling (None, 30, 30, 64) 0

2D)

conv2d\_3 (Conv2D) (None, 28, 28, 64) 36928

max\_pooling2d\_3 (MaxPooling (None, 14, 14, 64) 0

2D)

conv2d\_4 (Conv2D) (None, 12, 12, 64) 36928

max\_pooling2d\_4 (MaxPooling (None, 6, 6, 64) 0

2D)

flatten (Flatten) (None, 2304) 0

dense (Dense) (None, 64) 147520

dense\_1 (Dense) (None, 6) 390

=================================================================

Total params: 278,086

Trainable params: 278,086

Non-trainable params: 0

model.save("unsus\_keras.h5")

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

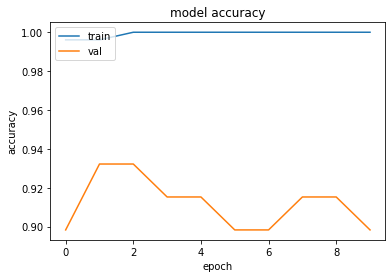
plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()



model.evaluate(test)

2/2 [==============================] - 5s 1s/step - loss: 0.9275 - accuracy: 0.9375

[0.927470326423645, 0.9375]