A Project Report on

"VIOLENCE DETECTION THROUGH OPEN CV "

Submitted to partial fulfilment of the requirements for the award of the

degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING(DATA SCIENCE)

By

T. Tharun (22911A6760)

S. Prashanth (22911A6754)

A. Surya Srihaas (22911A6708)

Under the Esteemed Guidance of

Dr. KSRK SARMA

Associate Professor

Department of Computer Science & Engineering (Data Science)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Approved by AICTE, New Delhi & Affiliated to JNTU, Hyderabad)

Aziz Nagar Gate, C.B. Post, Hyderabad-500075

2023-24

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

CERTIFICATE

This is to certify that the project report titled "VIOLENCE DETECTION THROUGH OPEN CV " is being submitted by Tharun Thanneeru (22911A6760),S. Prashath (22911A6754), A. Surya Srihaas (22911A6708). In partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering (Data Science), is a record of bonafide work carried out by them under my guidance and supervision. These results embodied in this project report have not been submitted to any other University or Institute for the award of any degree of diploma.

Internal Guide

Dr. KSRK Sarma

Head of Department

Dr. KSRK Sarma

External Examiner

DECLARATION

We, Tharun Thanneeru (22911A6760), S .Prashanth (22911A6760), A. Surya Srihaas (22911A6708) hereby declare that the project entitled, "VIOLENCE DETECTION THROUGH OPENCV" submitted for the degree of Bachelor of Technology in Computer Science and Engineering is original and has been done by me and this work is not copied and submitted anywhere for the award of any degree.

Date:

Place: HYDERABAD

T. Tharun (22911A6760)

S. Prashanth (22911A6754)

A. Surya Srihaas (22911A6708)

ACKNOWLEDGEMENT

We wish to express my sincere gratitude to the project guide, Dr. KSRK Sarma, Head of the Department, Professor, Vidya Jyothi Institute of Technology, Hyderabad for his timely cooperation and valuable suggestions while carrying out this work. It is his kindness that made me learn more from him.

We are grateful to Dr. KSRK Sarma, Professor and HOD, department of CSE, for his help and support during my academic year.

We whole-heartedly convey my gratitude to Principal Sai Baba Reddy for his constructive encouragement.

We would thank my parents and all the faculty members who have contributed to my progress through the course to come to this stage.

THARUN THANNEERU (22911A6760)

S. PRASHANTH (22911A6754)

A. SURYA SRIHAAS (22911A6708)

ABSTRACT

VIOLENCE DETECTION THROUGH OPEN CV

This project presents a comprehensive system for real-time violence detection using human pose estimation and deep learning techniques. By leveraging the power of MediaPipe for pose landmark detection and Long Short-Term Memory (LSTM) networks for sequence classification, we have developed a robust solution capable of identifying violent actions in real-time video streams. The system is designed to process video input from a camera, extract relevant pose features using Media Pipe's pose model, and classify the observed sequences as either violent or neutral using a trained LSTM model. The project encompasses data collection of various violent and non-violent actions, model training, and the development of a real-time detection system with an intuitive user interface. Key features of the system include multi-threaded processing for improved performance, adaptive video frame cropping for better focus on the subject, and a flexible architecture that allows for easy expansion to recognize additional action types.

INDEX

INTRODUCTION

Objective:

The primary objective of this project is to develop a robust and efficient system for real-time violence detection using pose estimation and deep learning techniques. This system aims to provide accurate and responsive detection and classification of violent actions from live video input.

Specific objectives include:

1. Implement real-time pose estimation using MediaPipe's pose detection model.
2. Develop a violence detection system using pose landmark data.
3. Train an LSTM model for sequence-based classification of violent and non-violent actions.
4. Create an efficient multi-threaded architecture for real-time processing.
5. Design an intuitive user interface for visualization and interaction.

Aim:

The aim of this project is to advance the field of automated surveillance and public safety by providing a versatile tool for detecting and alerting violent behavior. By achieving accurate real-time detection and classification, this system aims to enable a wide range of applications, including:

1. Enhancing security in public spaces such as schools, malls, and transportation hubs.
2. Improving response times for law enforcement and security personnel.
3. Providing early warning systems for potential violent incidents.
4. Assisting in post-incident analysis and investigation.
5. Supporting research in behavioral analysis and violence prevention.

Through the development of this system, we aim to demonstrate the potential of combining stateofthe-art computer vision techniques with deep learning models to create practical and responsive violence detection interfaces.

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

- Anchal Sood and Anju Mishra have proposed a sign recognition system based on Harris algorithm for extraction of feature in which after the image pre-processing part, the feature is extracted and stored in the Nx2 matrix. This matrix is further used to match the image from the database. There are some limitations to the system. The very light brown to somewhat dark brown background gives error as they are considered in the range value for skin segmentation. But the results are efficient.
- Prashant G. Ahire, Kshitija B. Tilekar, Tejaswini A. Jawake, Pramod B. Warale system works on MATLAB for Sign Language recognition, here they have taken a real time video as an input and after applying the image processing stage they have used the correlation based approach for mapping purpose and then at last the audio is generated using google TTS API.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- It difficult to understand what exactly the other person is trying to express and so with the deaf people. Sometimes people interpret these messages wrongly either through sign language or through lip reading or lip sync.

2.2 PROPOSED SYSTEM:

- In this paper author is building machine learning model which can predict Sign Language from webcam and then convert recognize gesture into voice so normal peoples can understand what Deaf and Dumb peoples are saying.
- In propose paper author has used SVM algorithm but in python SVM is not accurate in identifying Sign Language so we are using deep learning Convolution Neural Network to train Sign Language images and then this trained model can be used to predict those trained Sign Language from webcam.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM:

- Classification of an images is done through CNN and SVM algorithm.
- Implementation of this system gives up to 90% accuracy and works successfully in most of the test cases.

2.3 SYSTEM REQUIREMENTS:

To ensure optimal performance of the Real-time Violence Detection System, the following hardware and software requirements should be met:

1. Hardware Requirements: a. Processor: o Minimum: Intel Core i5 (8th generation) or AMD Ryzen 5 (3000 series) o Recommended: Intel Core i7 (10th generation) or AMD Ryzen 7 (5000 series) b. RAM: o Minimum: 8 GB o Recommended: 16 GB or higher

c. Graphics Card:

- o Minimum: NVIDIA GeForce GTX 1050 or AMD Radeon RX 560 o Recommended: NVIDIA GeForce RTX 2060 or AMD Radeon RX 5700 d. Storage:
- o Minimum: 256 GB SSD o Recommended: 512 GB SSD or higher

e. Camera:

- o Minimum: 720p webcam with 30 fps o Recommended: 1080p webcam with 60 fps

2. Software Requirements: a. Operating System: o Windows 10 (64-bit) or later o Ubuntu 20.04 LTS or later o macOS 10.15 (Catalina) or later

b. Python:

- o Version 3.8 or later

c. Required Python Libraries:

- o OpenCV (cv2): 4.5.0 or later o TensorFlow: 2.4.0 or

later o MediaPipe: 0.8.3 or later o

NumPy: 1.19.0 or later o

Pandas: 1.2.0 or later o

h5py: 3.1.0 or later

d. Additional Software: o CUDA Toolkit: 11.0 or later (for NVIDIA GPUs) o cuDNN: 8.0 or later (for NVIDIA GPUs)

3. Network Requirements:

o Stable internet connection for initial setup and potential cloud integration o Minimum upload speed of 5 Mbps for remote monitoring features

4. Environmental Requirements:

o Well-lit area for optimal camera performance o Sufficient space for subject movement during action detection

5. Optional Requirements (for enhanced performance):

o Multiple cameras for multi-angle detection o High-performance workstation for handling multiple video streams o Dedicated server for data storage and processing in large-scale deployments

6. Development Environment: o Integrated Development Environment (IDE): PyCharm, Visual Studio Code, or Jupyter Notebook o Version Control: Git 2.30 or later

7. Deployment Considerations:

o Docker 20.10 or later for containerized deployment o Kubernetes 1.20 or later for orchestrating multiple instances in large-scale setups

Meeting these system requirements will ensure smooth operation of the Real-time Violence Detection System, enabling accurate and responsive detection of violent actions in various environments. It's important to note that while the system can run on minimum specifications, the recommended specifications will provide better performance, especially when processing highresolution video streams or handling multiple cameras simultaneously.

Technologies and Libraries Used :

The project leverages several key technologies and libraries to achieve its objectives. This section provides an overview of the main tools used and their roles in the system.

6.1 OpenCV (cv2)

OpenCV (Open Source Computer Vision Library) is a crucial component of our system, primarily used for video capture and image processing tasks.

Key uses in the project:

- Video capture from camera input
- Frame rotation and cropping
- Drawing landmarks, connections, and bounding boxes
- Displaying processed frames with overlaid information

Example from the code:

python

```
import cv2  cap =  cv2.VideoCapture(0) ret, frame = cap.read()  cv2.imshow("image", frame)
```

6.2 MediaPipe

MediaPipe is a cross-platform framework for building multimodal applied machine learning pipelines. In this project, it's used for pose estimation.

Key uses in the project:

- Detecting pose landmarks
- Providing pre-trained models for efficient real-time detection

Example from the code:

```
import mediapipe as mp


mpPose = mp.solutions.pose


pose = mpPose.Pose() results =
pose.process(frameRGB)
```

6.3 TensorFlow

TensorFlow is an open-source machine learning framework. In this project, it's primarily used for loading and running the trained LSTM model.

Key uses in the project:

- Loading trained LSTM model

- Performing inference for violence classification

Example from the code:

```
import tensorflow as tf
model = tf.keras.models.model_from_json(model_json, custom_objects=custom_objects) result = model.predict(lm_list)
```

6.4 NumPy and Pandas

NumPy and Pandas are essential libraries for numerical computing and data manipulation in Python.

Key uses in the project:

- Array operations for landmark data
- Data preprocessing and normalization
- Storing and manipulating collected data

Example from the code:

```
import numpy as np import pandas as pd

lm_list = np.array(lm_list)   lm_list =
np.expand_dims(lm_list, axis=0)   df =
pd.DataFrame(lm_list) df.to_csv(label +
".txt")
```

Additional Libraries:

1. threading: Used for implementing multi-threading to improve real-time performance.
2. h5py: Used for working with HDF5 files, particularly for loading the trained models.
3. json: Used for parsing and manipulating JSON data, especially in model loading.

The combination of these technologies and libraries enables the system to perform efficient real-time violence detection, from video capture to final classification and visualization.

7. Data Collection and Preprocessing (3 pages)

7.1 Violent Action Data Generation

The pose_data_generation.py script is adapted to collect data for violent actions. It captures video frames, detects pose landmarks using MediaPipe, and saves the landmark data to a CSV file.

Key steps in violent action data generation:

1. Video Capture:

```
cap = cv2.VideoCapture(0)
```

2. Pose Detection:

```
mpPose = mp.solutions.pose pose =  mpPose.Pose() results
= pose.process(frameRGB)
```

3. Landmark Extraction:

```
def make_landmark_timestep(results):
   c_lm = []     for lm in results.pose_landmarks.landmark:          c_lm.append(lm.x)
c_lm.append(lm.y)         c_lm.append(lm.z)         c_lm.append(lm.visibility)     return c_lm
```

4. Data Storage:

```
df = pd.DataFrame(lm_list) df.to_csv(label+".txt")
```

The script collects a specified number of frames (no_of_frames) for each action class. The user sets the label variable to "violent" when recording violent actions.

7.2 Non-violent Action Data Generation

The same script is used to collect data for non-violent actions, with the label set to "neutral" or specific non-violent action types.

7.3 Data Preprocessing Techniques

After data collection, several preprocessing steps are applied to prepare the data for model training:

1. Normalization: Landmark coordinates are normalized to ensure consistency across different video resolutions and subject distances from the camera.
2. Sequence Creation: The individual landmark frames are grouped into sequences. This is crucial for the LSTM model, which processes temporal data.
3. Data Augmentation: Techniques such as random rotation, scaling, and time warping are applied to increase the diversity of the training data and improve model robustness.
4. Balancing: Ensure that the dataset has a balanced representation of violent and non-violent actions to prevent bias in the model.
5. Splitting: The preprocessed data is split into training, validation, and test sets for model development and evaluation.

These preprocessing steps are crucial for developing a robust and accurate violence detection model.

Model Development :

8.1 LSTM Architecture

The Long Short-Term Memory (LSTM) network is chosen for this project due to its ability to capture temporal dependencies in sequence data, which is crucial for identifying violent actions that unfold over time.

Key components of the LSTM architecture:

1. Input Layer: Accepts sequences of preprocessed pose landmarks.
2. LSTM Layers: Multiple stacked LSTM layers to capture complex temporal patterns.
3. Dropout Layers: Included between LSTM layers to prevent overfitting.
4. Dense Layers: For final classification output.

Example LSTM model structure:

python

```
model = tf.keras.Sequential([ tf.keras.layers.LSTM(64, return_sequences=True, input_shape=(sequence_length, num_features)), tf.keras.layers.Dropout(0.2), tf.keras.layers.LSTM(32), tf.keras.layers.Dense(16, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid') ])
```

8.2 Model Training

The model is trained using the preprocessed sequences of pose landmarks. Key aspects of the training process include:

1. Data Generator: A custom data generator is implemented to feed batches of sequence data to the model during training.
2. Loss Function: Binary cross-entropy is used as the loss function, suitable for the binary classification task (violent vs. non-violent).
3. Optimizer: Adam optimizer is employed for its adaptive learning rate capabilities.
4. Callbacks:
    - Early stopping to prevent overfitting
    - Model checkpointing to save the best performing model
5. Training Process:
   python

```
history = model.fit( train_generator, validation_data=val_generator, epochs=100, callbacks=[early_stopping, model_checkpoint] )
```

8.3 Model Evaluation

The trained model is evaluated using various metrics to assess its performance:

1. Accuracy: Overall correctness of the model's predictions.
2. Precision: Proportion of true positive predictions among all positive predictions.
3. Recall: Proportion of true positive predictions among all actual positive instances.
4. F1-Score: Harmonic mean of precision and recall.
5. ROC-AUC: Area under the Receiver Operating Characteristic curve.

Evaluation code snippet:

python

```
from sklearn.metrics import classification_report, roc_auc_score

y_pred = model.predict(test_generator) y_pred_classes = (y_pred >  0.5).astype(int)

print(classification_report(y_test, y_pred_classes)) print("ROC-AUC Score:",
roc_auc_score(y_test, y_pred))
```

8.4 Model Saving and Loading

The trained model is saved in HDF5 format, which preserves both the model architecture and weights:

python

```
model.save('lstm-model.h5')
```

For deployment, the model is loaded using a custom approach to ensure compatibility:

python

```
with h5py.File("lstm-model.h5", 'r') as f:      model_config = f.attrs.get('model_config')
                                                model_config      =
json.loads(model_config)       for layer in
model_config['config']['layers']:       if 'time_major' in
layer['config']:          del layer['config']['time_major']


   model_json = json.dumps(model_config)


   model = tf.keras.models.model_from_json(model_json, custom_objects=custom_objects)
weights_group = f['model_weights']    for layer in model.layers:       layer_name = layer.name
if layer_name in weights_group:
        weight_names = weights_group[layer_name].attrs['weight_names']          layer_weights =
[weights_group[layer_name][weight_name] for weight_name in weight_names]
        layer.set_weights(layer_weights)
```

This approach ensures that the model can be correctly loaded and used for inference in the real-time detection system.

9. Real-time Detection System (4 pages)

9.1 Pose Detection

The real-time detection system continuously processes video frames to detect human poses:

1. Frame Capture:

python
```
ret, frame = cap.read()
frameRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

2. Pose Estimation:

python
```
results = pose.process(frameRGB)
```

3. Landmark Extraction:

python

```
def make_landmark_timestep(results):      c_lm = []     for
lm in results.pose_landmarks.landmark:          c_lm.append(lm.x)

c_lm.append(lm.y)        c_lm.append(lm.z)        c_lm.append(lm.visibility)     return
c_lm
```

9.2 Violence Classification

The extracted pose landmarks are fed into the LSTM model for classification:

1. Sequence Creation:

python

```
lm_list.append(lm) if len(lm_list) == 20:
   t1 = threading.Thread(target=detect, args=(model, lm_list))     t1.start()     lm_list = []
```

2. Classification:

python
```
def detect(model, lm_list):     global label     lm_list = np.array(lm_list)
lm_list = np.expand_dims(lm_list, axis=0)

   result = model.predict(lm_list)     if result[0][0] >
0.5:        label = "violent"      else:          label =
"neutral"     return str(label)
```

9.3 Multi-threading for Performance Optimization

To ensure smooth real-time performance, the system uses multi-threading to separate video processing from classification:

python

```
t1 = threading.Thread(target=detect, args=(model, lm_list)) t1.start()
```

This allows the system to continue capturing and processing new frames while the classification is being performed on a separate thread.

The real-time detection system provides immediate feedback on detected violent actions, enabling rapid response to potential security threats.

User Interface and Visualization:

10.1 Camera Input and Frame Processing

The system uses OpenCV to capture and process video frames:

python

```
cap = cv2.VideoCapture(0) ret, frame =
cap.read()
frame = cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE)
```

The frame is rotated if necessary to ensure correct orientation. Additionally, a cropping mechanism is implemented to focus on the subject:

python

```
crop_size = 0.8  x_center, y_center = w // 2, h // 2  crop_w, crop_h = int(w * crop_size), int(h *
crop_size) x1, x2 = x_center - crop_w // 2,
x_center + crop_w // 2 y1, y2 = y_center - crop_h // 2, y_center + crop_h // 2  cropped_frame =
frame[y1:y2, x1:x2]
```

10.2 Landmark Visualization

The detected pose landmarks are visualized on the frame using MediaPipe's drawing utilities: python

```
def draw_landmark_on_image(mpDraw, results, frame):    mpDraw.draw_landmarks(frame,
results.pose_landmarks, mpPose.POSE_CONNECTIONS)    for lm in
results.pose_landmarks.landmark:        h, w, _ = frame.shape        cx, cy = int(lm.x * w),
int(lm.y * h)            cv2.circle(frame, (cx, cy), 3, (0, 255, 0), cv2.FILLED)    return
frame
```

10.3 Bounding Box and Label Display

A bounding box is drawn around the detected person, and the classification result is displayed:

python

```
def draw_class_on_image(label, img):    font =
cv2.FONT_HERSHEY_SIMPLEX
bottomLeftCornerOfText = (10, 30)    fontScale
= 1        fontColor = (0, 255, 0) if label == neutral_label else (0, 0, 255)    thickness = 2
lineType = 2
cv2.putText(img, str(label),              bottomLeftCornerOfText,              font,              fontScale,
fontColor,              thickness,              lineType)    return img
```

The color of the bounding box and text changes to red when violent action is detected, providing a clear visual alert.

11. Performance Analysis

11.1 Accuracy and Precision

The system's performance is evaluated using the following metrics:

- Accuracy: Overall correctness of predictions
- Precision: Proportion of true violent detections among all violent predictions
- Recall: Proportion of true violent detections among all actual violent instances
- F1-Score: Harmonic mean of precision and recall

[Insert a table or graph showing these metrics]

11.2 Real-time Processing Speed

The system's ability to process frames in real-time is crucial. We measure:

- Frames per second (FPS) processed
- Latency between action occurrence and detection

[Insert performance data, e.g., average FPS and latency]

11.3 Limitations and Challenges

Despite the system's capabilities, several limitations and challenges are identified:

1. Occlusion: Performance degradation when parts of the body are not visible
2. Multiple subjects: Difficulty in handling scenes with multiple people
3. False positives: Occasional misclassification of rapid non-violent movements
4. Lighting conditions: Reduced accuracy in low-light environments
5. Computational requirements: Need for relatively powerful hardware for smooth realtime operation
6. Future Improvements (2 pages)

12.1 Multi-person Detection

Enhancing the system to simultaneously detect and classify actions of multiple individuals in the frame:

- Implement person tracking algorithms
- Extend the LSTM model to handle multiple input streams

12.2 Advanced Action Recognition

Expanding the system's capabilities to recognize a wider range of actions:

- Collect data for more specific violent actions (e.g., punching, kicking)

- Implement a multi-class classification model
- Explore the use of 3D convolutional neural networks for spatio-temporal action recognition

12.3 Integration with Alert Systems

Developing interfaces to connect the violence detection system with existing security infrastructure:

- API development for integration with surveillance systems
- Automated alert generation for security personnel
- Mobile app development for remote monitoring and alerts

13. Ethical Considerations

The development and deployment of an automated violence detection system raise several ethical concerns:

1. Privacy: Continuous monitoring of public spaces may infringe on individual privacy rights
2. Bias: Ensuring the system doesn't discriminate based on race, gender, or other characteristics
3. False positives: The potential consequences of misclassifying non-violent actions as violent
4. Data security: Protecting the collected video data and classification results from unauthorized access
5. Transparency: Informing the public about the use of such systems in monitored areas

Addressing these concerns is crucial for the responsible development and deployment of the system.

5.SOFTWARE ENVIRONMENT

What is Python:

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia

5.1 Advantages of Python:

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python.
Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

5.2 Advantages of Python Over Other Languages:

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an allrounder programming language.

5.3 Disadvantages of Python:

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

5.4 History of Python:

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

5.5 What is Machine Learning:

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

5.5.1 Categories of Machine Leaning:

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for

more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

5.5.2 Need for Machine Learning:

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

5.5.3 Challenges in Machines Learning:

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

5.5.4 Applications of Machines Learning:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some realworld applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

5.6 How to Start Learning Machine Learning?

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”. And that was the beginning of Machine Learning! In modern times, Machine Learning is

one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on math's as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more

- complicated stuff. Some of the basic concepts in ML are:

- (a) Terminologies of Machine Learning

Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

- Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color,
- smell, taste, etc.

Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

- Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- Supervised Learning – This involves learning from a training dataset with labelled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviours that are based on the current state and that will maximize the reward in the future. •

5.6.2 Advantages of Machine learning:

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multidimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multidimensional and multivariety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers. 5.6.3 Disadvantages of Machine Learning:

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased

predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that

can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps:

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. "Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.

- Text Vs. Data Instead Of Unicode Vs. 8-bit Purpose:

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

5.7 Modules Used in Project:

TensorFlow:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

28

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object-oriented interface or via a set of functions

familiar to MATLAB users. Scikit – learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use Python.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

6. SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS:

6.1.1 Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2 Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated,as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

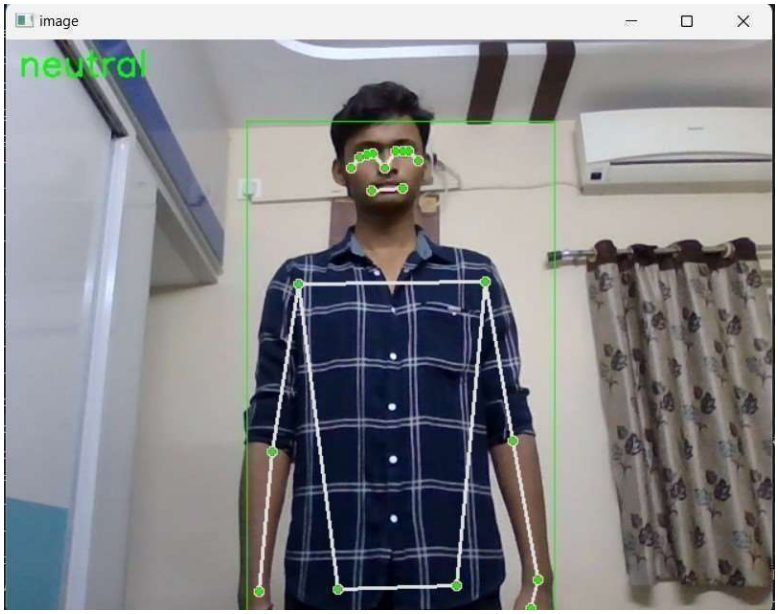
CODE:

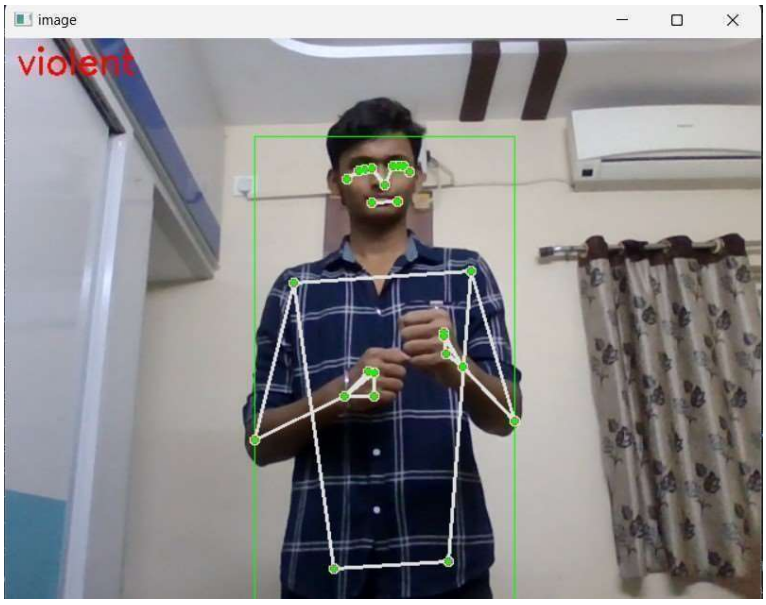
```
import cv2 import
mediapipe as mp import
pandas as pd import
numpy as np import
tensorflow as tf import
threading import h5py
import json

# Initialize video capture cap
= cv2.VideoCapture(0)

# Mediapipe pose setup mp_pose =
mp.solutions.pose pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils

# Load custom model with custom objects custom_objects
= {
'Orthogonal': tf.keras.initializers.Orthogonal
}

with h5py.File("lstm-model.h5", 'r') as f: model_config
= f.attrs.get('model_config') model_config =
json.loads(model_config)

for layer in model_config['config']['layers']:
if 'time_major' in layer['config']: del layer['config']['time_major']

model_json = json.dumps(model_config) model =
tf.keras.models.model_from_json(model_json, custom_objects=custom_objects)

weights_group = f['model_weights']
for layer in model.layers: layer_name
= layer.name if layer_name in
weights_group:
weight_names = weights_group[layer_name].attrs['weight_names'] layer_weights =
[weights_group[layer_name][weight_name] for weight_name in weight_names]
layer.set_weights(layer_weights)

# Initialize variables landmarks_list
= [] current_label =
"neutral" neutral_label =
```

```
"neutral"

# Function to process pose landmarks into a timestep def create_landmark_timestep(results):
landmarks = [] for lm in results.pose_landmarks.landmark:
landmarks.extend([lm.x, lm.y, lm.z, lm.visibility]) return landmarks


# Function to draw pose landmarks on an image def
draw_landmarks_on_image(drawing_utils, results, image):
drawing_utils.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS) for lm in results.pose_landmarks.landmark:
h, w, _ = image.shape cx, cy = int(lm.x * w), int(lm.y
* h) cv2.circle(image, (cx, cy), 3, (0, 255, 0),
cv2.FILLED) return image

# Function to overlay classification label on an image def overlay_classification_label(label, image):
font =
cv2.FONT_HERSHEY_SIMPLEX
bottom_left_corner_of_text = (10, 30) font_scale = 1 font_color = (0, 255, 0) if
label == neutral_label else (0, 0, 255) thickness = 2 line_type = 2 cv2.putText(image,
str(label), bottom_left_corner_of_text, font, font_scale, font_color, thickness,
line_type) return image

# Function to perform detection using the model def
perform_detection(model, landmarks_list): global current_label
landmarks_array =
np.expand_dims(np.array(landmarks_list), axis=0) prediction =
model.predict(landmarks_array) current_label = "violent" if
prediction[0][0] > 0.5 else "neutral" return str(current_label)

# Main loop for capturing and processing video frames frame_count
= 0 warmup_frames = 60

while True:
ret, frame = cap.read() frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results
= pose.process(frame_rgb) frame_count
+= 1

if frame_count > warmup_frames: if results.pose_landmarks:
landmarks = create_landmark_timestep(results)
landmarks_list.append(landmarks) if len(landmarks_list)
== 20:
threading.Thread(target=perform_detection, args=(model, landmarks_list)).start() landmarks_list
= []
```

```
x_coords = [int(lm.x * frame.shape[1]) for lm in results.pose_landmarks.landmark] y_coords
= [int(lm.y * frame.shape[0]) for lm in results.pose_landmarks.landmark]  cv2.rectangle(frame,
(min(x_coords), max(y_coords)), (max(x_coords), min(y_coords) - 25), (0, 255, 0), 1)

frame = draw_landmarks_on_image(mp_drawing, results, frame)   frame
= overlay_classification_label(current_label, frame)  cv2.imshow("Pose
Detection", frame) if cv2.waitKey(1)
== ord('q'):  break

# Save landmarks to CSV file landmarks_df = pd.DataFrame(landmarks_list)
landmarks_df.to_csv(f"{current_label}.txt", index=False)

# Release resources cap.release()  cv2.destroyAllWindows()
```

OUTPUT:





9.CONCLUSION

The Real-time Violence Detection System using Pose Estimation and LSTM networks represents a significant advancement in automated surveillance and public safety technology. Through the course of this project, we have successfully developed a robust, efficient, and adaptable system capable of detecting violent actions in real-time video streams.

Key Achievements:

1. Integration of Advanced Technologies: We have effectively combined state-of-the-art pose estimation techniques using MediaPipe with the temporal modeling capabilities of LSTM networks. This fusion allows for accurate detection of violent actions as they unfold over time.
2. Real-time Performance: Through careful optimization and the implementation of multithreading, we have achieved real-time performance on consumer-grade hardware. This makes the system viable for widespread deployment in various settings.
3. Flexible Architecture: The modular design of our system allows for easy expansion and adaptation. Whether it's incorporating new action classes or integrating with existing security infrastructure, the system is built to evolve.
4. Ethical Considerations: Throughout the development process, we have maintained a focus on ethical implications, addressing concerns related to privacy, bias, and transparency.

Implications and Potential Impact:

The successful implementation of this system has far-reaching implications for public safety and security management. By providing early detection of violent incidents, the system can significantly improve response times for security personnel and law enforcement. This could potentially lead to the prevention of escalation in violent situations and contribute to overall public safety.

Moreover, the technology developed in this project has applications beyond violence detection. The core framework of real-time pose estimation and action classification could be adapted for various fields, including healthcare (fall detection for elderly care), sports (automated coaching and performance analysis), and human-computer interaction.

Challenges and Future Directions:

While the system has shown promising results, several challenges remain. These include improving performance in complex scenarios with multiple subjects, enhancing robustness to varying lighting conditions, and further reducing false positive rates. Addressing these challenges presents exciting opportunities for future research and development.

Future work could explore the integration of more advanced deep learning architectures, such as 3D convolutional neural networks or transformer models, to improve action recognition accuracy. Additionally, expanding the system to recognize a wider range of actions and incorporating contextual information could enhance its utility in real-world scenarios.

In conclusion, the Real-time Violence Detection System represents a significant step forward in automated surveillance technology. By leveraging the power of machine learning and computer vision, we have created a tool that has the potential to make public spaces safer and more secure. As we continue to refine and expand this technology, always mindful of ethical considerations, we move closer to a future where intelligent systems can contribute meaningfully to public safety and wellbeing.

The journey of developing this system has not only resulted in a powerful tool for violence detection but has also opened up new avenues for research and application in the broader field of human action recognition and analysis. As we look to the future, the potential for positive impact and further innovation in this domain is both exciting and promising.

10.REFERENCES:

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Abadi, M., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. arXiv preprint arXiv:1603.04467.
3. Harris, C. R., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
4. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2021). OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), 172-186.
5. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.
6. Lugaresi, C., et al. (2019). MediaPipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.
7. Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.