Annexure -1

Synopsis for Learning Project-II

Roll No	Regd No	Name of the Student
23CSEAIML057	23UG010842	BIBHUKALYAN NAYAK
23CSEAIML047	23UG010832	SAMBIT SUBHANKAR DAS
23CSEAIML054	23UG010839	ATAL MAHAPRASAD

Title Of Project: Real-Time Face Detection Using PyTorch and Deep Learning Models

Synopsis:

1. Introduction

Face detection is a fundamental computer vision task with applications ranging from security systems to social media platforms. This project leverages **PyTorch** and deep learning models to design and implement a robust, real-time face detection system. The system will accurately detect faces in images and videos and provide visual feedback using bounding boxes.

2. Objectives

- Develop a face detection system using PyTorch and deep learning techniques.
- Ensure real-time performance for face detection in video streams.
- Highlight detected faces with bounding boxes.
- Integrate pre-trained models like **Faster R-CNN**, **MTCNN**, or **YOLO** for accuracy.
- Compare the performance of Haar Cascade and deep learning models.
- Implement a user-friendly interface for seamless system interaction.

3. Scope of the Project

The project aims to:

- Utilize **grayscale conversion** for efficient image processing.
- Implement a face detection pipeline capable of handling both static images and real-time video streams.
- Compare traditional face detection techniques (e.g., Haar Cascade) with modern deep learning approaches.
- Demonstrate scalability and efficiency for real-world applications.

4. Methodology

Step 1: Data Preparation

- Load datasets (e.g., WIDER FACE) or use live video streams via OpenCV.
- Preprocess images (grayscale conversion, normalization, resizing).

Step 2: Model Selection

- Use pre-trained deep learning models for face detection, such as:
    - **MTCNN** (Multi-task Cascaded Convolutional Networks)

    - **Faster R-CNN**
    - **YOLO (You Only Look Once)**
- Implement PyTorch for model training and inference.

Step 3: Real-Time Face Detection

- Use OpenCV to read video frames and pass them through the detection model.
- Apply bounding boxes to detected faces.
- Optimize performance for real-time processing.

Step 4: Performance Comparison

- Compare Haar Cascade-based detection with deep learning-based models in terms of:
    - Accuracy
    - Speed (FPS)
    - Scalability

Step 5: User Interface

- Design a graphical interface to:
    - Upload images or videos.
    - Toggle between detection methods.
    - View live detection results.

5. Tools and Technologies

- **Programming Language:** Python
- **Frameworks/Libraries:**
    - PyTorch (for deep learning models)
    - OpenCV (image/video processing)
    - NumPy, Matplotlib (data processing and visualization)
- **Hardware/Software:**
    - GPU/CPU for model execution
    - Camera for real-time video input

6. Expected Outcomes

- A fully functional, real-time face detection system.
- Performance analysis of traditional (Haar Cascade) vs. modern deep learning models.
- A user-friendly interface for demonstrating face detection in images and videos.
- Insights into optimizing detection models for real-world deployment.

7. Applications

- Surveillance and Security Systems
- Real-Time Attendance Monitoring
- Human-Computer Interaction
- Social Media Platforms
- Augmented Reality

8. Conclusion

This project will demonstrate the efficiency and accuracy of real-time face detection using deep learning models in PyTorch. It aims to bridge the gap between traditional methods and modern advancements in computer vision, offering scalable and real-world applications.

Group No:

Name of the Class Teacher: Ms. Sucheta Krupalini Maharana

Ms. Sucheta Krupalini Maharana
Class Teacher

Mr.Bhavani Sankar Panda
Project Coordinator