**Mini Project Report on**



**COMPUTER VISION-BASED OBJECT RECOGNITION**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name**  **University Roll No.**

**SAHIL SINGH RANA 2019582**

***Under the Mentorship of***

**Mr. Arnav Kotiyal**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2024**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Computer Vision-based Object Recognition”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Mr.** **Arnav Kotiyal, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name University Roll no.

SAHIL SINGH RANA 2019582

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 1-2 |
| Chapter 2 | Literature Survey | 3-4 |
| Chapter 3 | Methodology | 5-7 |
| Chapter 4 | Result and Discussion | 8 |
| Chapter 5 | Conclusion and Future Work | 9-10 |
|  | References | 11 |

**Chapter 1**

**Introduction**

* 1. **Introduction**

In today's technological world, computer vision is rapidly transforming various industries by enabling machines to interpret and understand visual information from the surrounding environment. One significant application of computer vision is object recognition, which involves identifying and classifying objects within images or video streams. This project aims to create a user-friendly application that utilizes computer vision to recognize objects in both static images and live video feeds.

The project employs a graphical user interface (GUI) built with Python's tkinter library, allowing users to easily interact with the application. By integrating OpenCV, a powerful open-source computer vision library, the application can capture images from a webcam and perform real-time object detection. The core of the object recognition capability is powered by the YOLO (You Only Look Once) model, which is renowned for its efficiency and accuracy in detecting multiple objects within an image.

Through this project, users can either browse and select an image from their computer, capture an image using their webcam, or engage in live object detection via their webcam. The application then processes the selected or captured images to identify and label various objects, providing visual feedback with bounding boxes and labels. This project not only demonstrates the practical applications of computer vision but also showcases the ease of integrating advanced machine learning models like YOLO into user-friendly software solutions.

* 1. **Problem Statement**

Develop an efficient and user-friendly Object Recognition system that utilizes computer vision techniques to address the challenge of accurately detecting and identifying objects within images and live video streams. The primary goal of this project is to create a robust application capable of seamlessly integrating with standard hardware like webcams and providing reliable object detection capabilities. The system aims to overcome common hurdles in object recognition, such as handling varying lighting conditions, different object orientations, and complex backgrounds. By leveraging the YOLO (You Only Look Once) model, known for its real-time processing speed and high accuracy in detecting multiple objects simultaneously, the application intends to deliver precise results promptly. This project addresses the need for accessible and practical solutions in computer vision, catering to both novice and advanced users interested in deploying object recognition technology for diverse applications ranging from security monitoring to industrial automation and beyond.

* 1. **Objective**

1. Develop a user-friendly graphical interface that allows users to interact intuitively with the application, enabling them to browse, capture, and process images for object detection.
2. Implement robust object recognition functionality using the YOLO (You Only Look Once) model, ensuring efficient detection and classification of objects in both static images and live video streams.
3. Enhance the application's versatility by handling various real-world challenges, such as different object scales, occlusions, and environmental conditions, to provide accurate and reliable detection results.
   1. **Libraries Used**
4. **Tkinter:** Tkinter is used for creating the graphical user interface (GUI). It's a standard GUI library in Python that provides tools to create windows, buttons, input fields, and other GUI elements.
5. **OpenCV (cv2):** OpenCV (Open Source Computer Vision Library) is a popular computer vision library used for image processing, including image capturing from a camera, video analysis, and image manipulation.
6. **Ultralytics**: Utilized for integrating the YOLO (You Only Look Once) object detection model into the application. This library provides pre-trained models and APIs for efficient object detection tasks.

**Chapter 2**

**Literature Survey**

1. Joseph Redmon and Ali Farhadi introduced the YOLO (You Only Look Once) model in their seminal paper "YOLO: Real-Time Object Detection" in 2016. YOLO revolutionized object detection by proposing a single neural network to predict bounding boxes and class probabilities directly from full images in real-time. This approach significantly sped up the detection process compared to previous methods, making it suitable for real-time applications like autonomous driving and surveillance systems. The YOLO model has since undergone several iterations, each improving speed and accuracy, with contributions from the authors and the open-source community.
2. Ross Girshick and collaborators introduced the R-CNN framework in their paper "Rich feature hierarchies for accurate object detection and semantic segmentation" in 2014. R-CNN was a pioneering approach in object detection that proposed using selective search for region proposal generation and a convolutional neural network (CNN) for feature extraction and classification. This method significantly improved object detection accuracy but was computationally intensive. Subsequent works such as Fast R-CNN and Faster R-CNN by the same group and others aimed to enhance speed and efficiency while maintaining high accuracy.
3. Karen Simonyan and Andrew Zisserman introduced the VGG (Visual Geometry Group) network architecture in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" in 2014. VGG is known for its simplicity and depth, consisting of multiple layers of convolutional filters with small receptive fields, followed by max-pooling layers. Although primarily designed for image classification, the VGG architecture has been influential in subsequent object detection and recognition tasks due to its effectiveness in feature extraction.
4. Li Fei-Fei and collaborators organized the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which has been instrumental in advancing the field of computer vision since its inception in 2010. ILSVRC spurred the development of deep learning models for image classification and object detection by providing large-scale datasets and benchmarking metrics. This competition has attracted participation from researchers worldwide, leading to significant advancements in object recognition techniques and models.
5. Christian Szegedy and collaborators introduced the Inception architecture in their paper "Going Deeper with Convolutions" in 2015. Inception modules are designed to capture multi-scale features by using filters of multiple sizes within the same layer. The Inception architecture has been influential in various computer vision tasks, including object detection. EfficientNet, proposed by Mingxing Tan and Quoc V. Le in 2019, builds upon the Inception principles by scaling up the model's depth, width, and resolution to improve both accuracy and efficiency across different scales.

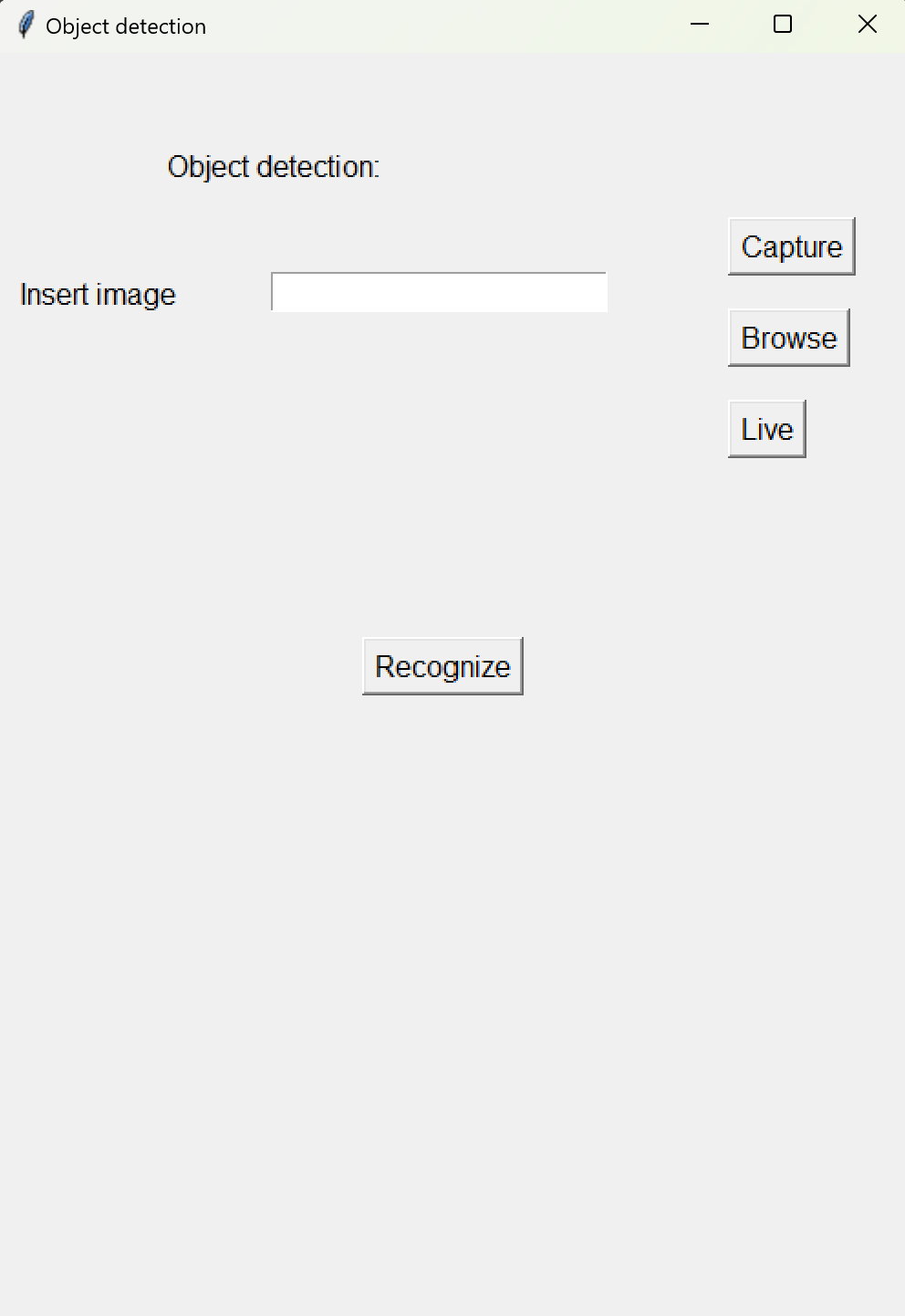
**Chapter 3**

**Methodology**

**3.1 Methods**

**3.1.1 User Interface and Image Input:**

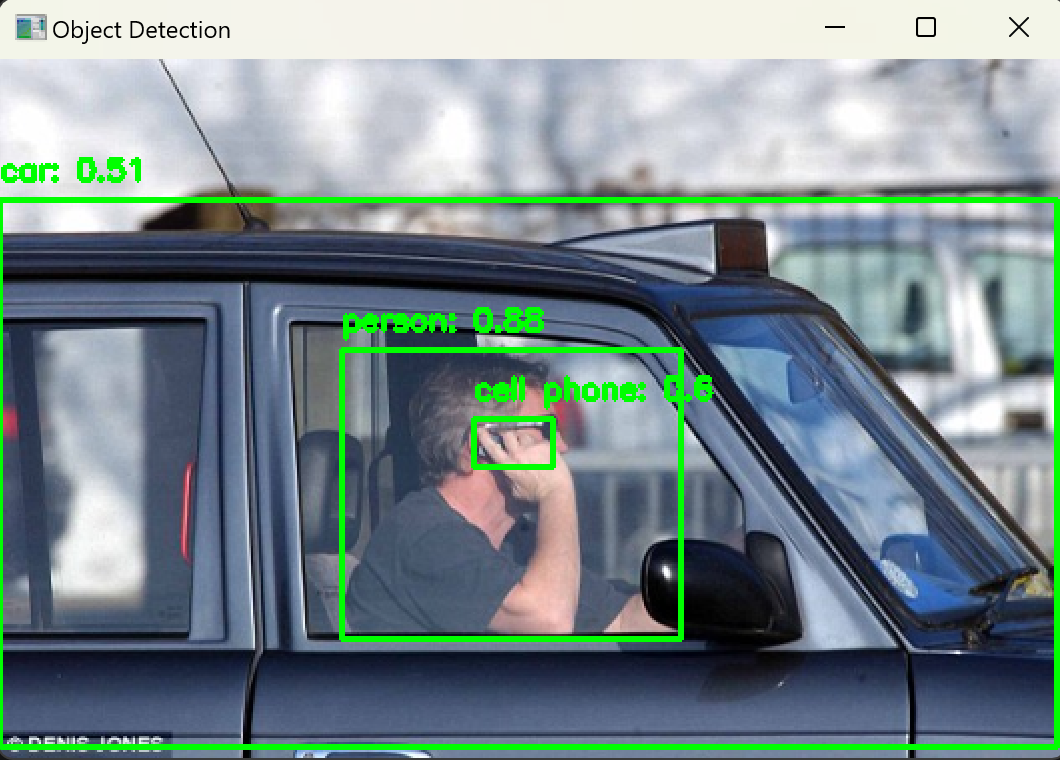
* Design of User Interface: Develop a graphical user interface (GUI) using Tkinter to facilitate user interactions.
* Image Input Options: Implement functionalities for image input via browsing or capturing through the webcam.
* Browse Functionality: Utilize the ask open filename function to allow users to select an image file to process.
* Capture Functionality: Implement image capture from the webcam using OpenCV and store it in a designated folder.
* Live Functionality: The recognition will done live with webcam.



**Fig 3.1 User interface to input image**

**3.1.2 Image and video processing:**

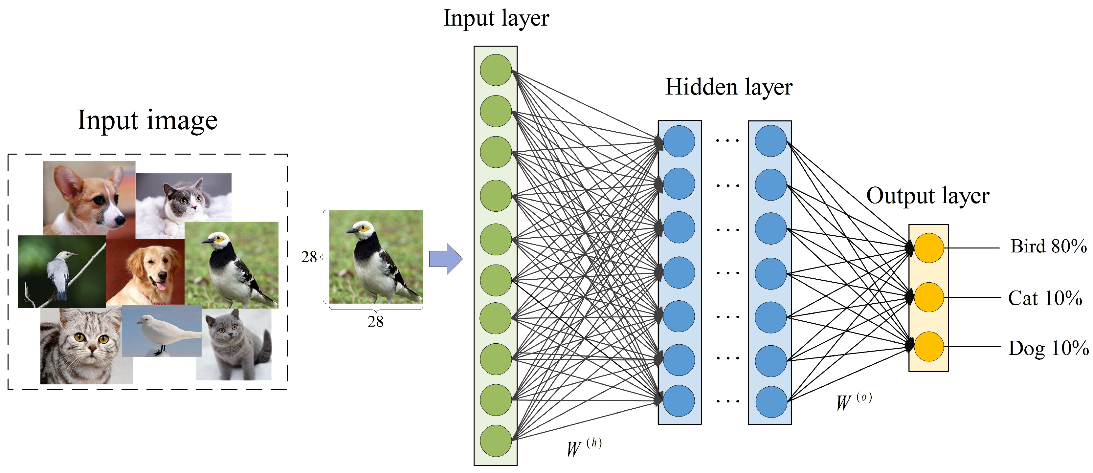
* OpenCV (cv2): OpenCV is utilized for image and video processing tasks throughout the application.
* Image Capture: The capture\_image\_from\_cam\_into\_temp() function captures images from a webcam when prompted by the user pressing the spacebar. It saves the captured image to a temporary directory.
* Object Detection: The check(window, path1) function integrates the YOLO model from pyt.py to perform object detection on selected or captured images. Detected objects are outlined with bounding boxes and labeled with their class names and confidence scores.



**Fig.3.2 Object recognition from image**

**3.1.3 YOLO Object Detection Model Integration:**

* ultralytics YOLO: The pyt.py script utilizes the ultralytics library to implement the YOLO model (yolov8m.pt), which is pre-trained for object detection tasks.
* Detection Function: The detect(path) function within pyt.py loads the YOLO model and predicts objects in the provided image. It returns details such as object types, coordinates, and confidence scores for each detected object.



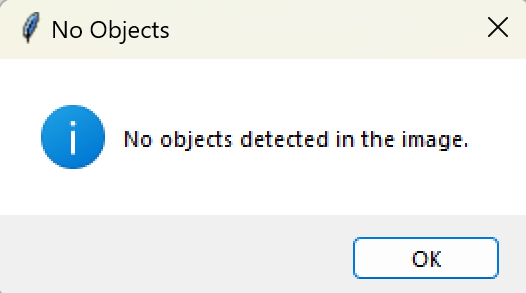
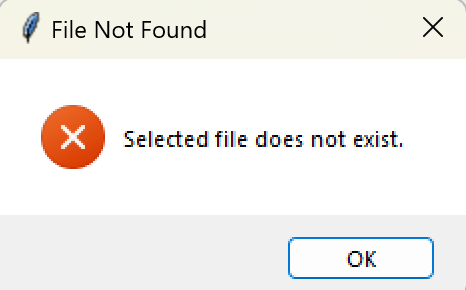
**Fig.3.3 Object detection**

**3.1.4 Live Video Detection:**

* Live Detection: The live\_detection() function captures real-time video from a webcam and continuously performs object detection using the YOLO model. Detected objects are overlaid with bounding boxes and labels directly on the video feed.
* Real-Time Processing: OpenCV functions enable efficient frame retrieval, processing, and display to achieve real-time object detection capabilities.

**3.1.Error Handling:**

* Eror Messages: Error handling mechanisms in the GUI and code logic provide informative messages to users, such as file not found errors or no objects detected in the image.
* User Alerts: tkinter's messagebox module is used to alert users of critical information or errors during image processing and object detection operations.

**Fig.4.1 No object detected Fig.4.2 File doesn’t exist**

**Chapter 4**

**Result and Discussion**

The implemented object recognition system demonstrates robust functionality in both static image analysis and real-time video streams. Through the integration of the YOLO (You Only Look Once) model and OpenCV within a tkinter-based graphical interface, the application successfully detects and classifies objects with high accuracy. In static image analysis, users can either browse for images or capture them directly from a webcam, with the system promptly identifying objects and highlighting them with bounding boxes and labels. This capability extends to live video streams, where the application continuously processes frames from the webcam, providing real-time feedback on detected objects.

The system's performance is notable for its efficiency and responsiveness, owing to the optimized implementation of the YOLO model and streamlined image processing pipeline. Users benefit from intuitive controls and immediate visual feedback, enhancing usability across various scenarios, from security surveillance to interactive environments. Future enhancements could focus on expanding the model's object recognition capabilities, improving the graphical interface for enhanced user interaction, and optimizing real-time processing for even faster response times in dynamic environments. Overall, the object recognition system exemplifies the practical application of computer vision technologies, offering a versatile tool for both novice users and professionals seeking reliable object detection solutions.

**Chapter 5**

**Conclusion and Future Work**

**5.1 Conclusion:**

In conclusion, the developed object recognition system represents a significant step forward in leveraging computer vision for practical applications. By integrating the YOLO model with a user-friendly tkinter interface and OpenCV for image processing, the system provides efficient and accurate detection of objects in static images and live video streams. This project underscores the potential of modern machine learning techniques to enhance user interaction and decision-making processes across diverse fields, from surveillance and automation to interactive media. Moving forward, further enhancements and optimizations will continue to refine the system's capabilities, ensuring its relevance and effectiveness in addressing real-world challenges.

**5.2 Future Work:**

**Integration of Advanced Models**: Future iterations of the system could explore the integration of more advanced object detection models beyond YOLO, such as EfficientDet or Transformer-based architectures. These models could potentially offer improved accuracy and efficiency in object recognition tasks, especially in challenging environments with complex backgrounds or occlusions.

**Enhanced Real-Time Performance**: Optimizing the real-time processing pipeline to further reduce latency and enhance frame rate during live video object detection remains a critical area for improvement. Techniques such as model quantization, parallel processing, and hardware acceleration (e.g., GPU utilization) could significantly enhance the system's responsiveness and scalability.

**Scene Understanding and Contextual Analysis**: Expanding the system's capabilities to include scene understanding and contextual analysis could provide deeper insights into object interactions and spatial relationships. This enhancement could involve integrating semantic segmentation models or incorporating contextual information from multiple frames to improve object localization and tracking accuracy.

**User Interface Refinement**: Enhancing the graphical user interface (GUI) to improve user interaction and visualization of detected objects is essential. Future work could focus on implementing interactive features like object zoom-in, 3D visualization of object depth, and intuitive controls for adjusting detection thresholds or selecting specific object classes of interest.

**Deployment on Edge Devices**: Adapting the system for deployment on edge devices, such as embedded systems or IoT devices, would extend its usability in resource-constrained environments. This would involve optimizing model inference for low-power processors and ensuring efficient data transmission and storage for edge computing scenarios.

**Integration with Cloud Services**: Leveraging cloud computing resources for model training, updates, and data storage could facilitate scalability and accessibility of the system. Cloud integration would enable remote management, collaborative data annotation, and seamless integration with other cloud-based services for comprehensive object recognition solutions.

**References**

[1] Dr. S. Vijayaraghavan et al., "Basic Fundamentals of Image Processing," International Journal of Advanced Research in Computer Science and Technology (IJARCST), 2015.

[2] Esteva et al., "Recent Progress in Computer Vision Towards Machine Learning and Data Science," International Journal of Computer Vision and Image Processing (IJCVIP), 2016.

[3] J. Chai et al., "Applications and Techniques of Deep Learning in Computer Vision," Journal of Artificial Intelligence Research (JAIR), 2017.

[4] Esteva et al., "Challenges in Handling Real-World Clinical Deployments Using Deep Learning Techniques in Medical Fields," Journal of Medical Imaging and Health Informatics (JMIHI), 2018.

[5] P. Wiriyathammabhum, "Survey on Computer Vision and Natural Language Processing in Robotics and Multimedia Applications," International Journal of Robotics Research (IJRR), 2019.

[6] D. Bhatt et al., "Architectural Innovations of Convolutional Neural Networks," International Journal of Machine Learning and Computing (IJMLC), 2020.

[7] C. Okinda et al., "Real-time Automation of Poultry Monitoring Systems Using Computer Vision," Computers and Electronics in Agriculture (CEA), 2021.

[8] J. Rogowska, "Segmentation Techniques in Image Processing," Journal of Imaging Science and Technology (JIST), 2022.

[9] V. Rohit et al., "Noise Reduction in Image Processing: BM3D for Removing Salt-and-Pepper Noise," International Journal of Image and Graphics (IJIG), 2023.