***A PROJECT ON***

# “Real-Time Object Detection”

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE COURSE OF

DIPLOMA IN BIG DATA ANALYSIS



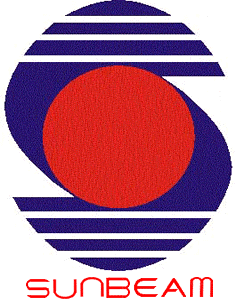
**SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

Submitted By:

Aniket Kangude (89495)

Omkar Mhetre (89441)

**Mr. Nitin Kudale Mrs. Manisha Hingne** Centre Coordinator Course Coordinator



**CERTIFICATE**

This is to certify that the project work under the title ‘Real-Time Object Detection’ is done by Aniket Kangude & Omkar Mhetre in partial fulfillment of the requirement for award of Diploma in Big Data Analysis Course.

Mr. Aniket P Mrs. Manisha Hingne

**Project Guide** **Course Coordinator**

Date:

# ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mrs. Manisha Hingne (Course Coordinator, SIIT ,Pune) and Project Guide Mr. Aniket P.

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Aniket Kangude

DBDA Feb 2025 Batch, SIIT Pune

Omkar Mhetre

DBDA Feb 2025 Batch, SIIT Pune

**TABLE OF CONTENTS**

1. **Introduction**
   1. Introduction And Objectives
   2. Why this problem needs To be Solved?
   3. Dataset Information

## Problem Definition and Algorithm

* 1. Problem Definition
  2. Algorithm Definition

## Experimental Evaluation

* 1. Methodology/Model
  2. Exploratory Data Analysis

## Results And Discussion

1. **GUI**
2. **GitHub link**

## Future Work And Conclusion

* 1. Future Work

7.2 Conclusion

* + 1. **Introduction**
       1. **Introduction And Objectives:**

The Object Detection project is designed to identify and locate specific objects within images or videos in real time using deep learning techniques. This system leverages the **YOLO (You Only Look Once)** pre-trained model, fine-tuned with a custom dataset, to achieve high accuracy and fast detection speeds. YOLO’s single neural network architecture allows simultaneous prediction of bounding boxes and class probabilities, making it suitable for applications where both speed and precision are critical. The model was trained and tested on real-world scenarios relevant to the chosen dataset, enabling it to adapt to specific object categories beyond generic datasets like COCO or Pascal V.

**Objectives:**

1. To implement an efficient object detection system using the YOLO architecture for real-time and accurate predictions.
2. To fine-tune a pre-trained YOLO model with a custom dataset to improve detection performance for domain-specific objects.
3. To process input images or live video streams and draw bounding boxes with confidence scores around detected objects.

## Why this problem needs To be Solved ?

Manual object identification is slow, error-prone, and inefficient for large-scale or real-time applications. With growing visual data from cameras and sensors, an automated, accurate, and fast detection system like YOLO is essential to improve efficiency, reduce human error, and enable real-time decision-making in areas such as security, traffic monitoring, and industrial inspection.

## Dataset Information.

The dataset used for this project is a **custom dataset** created to train the YOLO pre-trained model for domain-specific object detection. It contains images of the target objects collected from various sources, ensuring diversity in backgrounds, lighting conditions, and object orientations. Each image is labelled with bounding boxes and class names using annotation tools like **Labelling**. The dataset is split into:

For this project, a custom dataset was created by scraping images from **Google Images** using the Python library **iCrawler**. Specifically, the GoogleImageCrawler module from icrawler.builtin was used to automate the image collection process. A total of **15,000 images** were downloaded, covering multiple object categories relevant to the detection task.

The collected images were then manually filtered to remove duplicates and low-quality data. Each image was annotated in **YOLO format** using **LabelImg**, where bounding boxes were drawn around the target objects and class labels were assigned. The dataset was split into **training, validation, and test sets** to train and evaluate the YOLO model effectively.

This large and diverse dataset helped improve the model’s robustness, enabling accurate detection across different object shapes, sizes, backgrounds, and lighting conditions.

* **Training Set:** 80% of the images for model learning.
* **Validation Set:** 10% of the images for tuning and evaluation.
* **Test Set:** 10% of the images for final performance assessment.

## Problem Definition and Algorithm:

* + - 1. **Problem Definition**

The project aims to develop an automated system capable of detecting and locating specific objects within images or video streams in real time. Traditional methods of object identification are slow, inconsistent, and unable to handle complex variations in object appearance, lighting, and background.

By fine-tuning a pre-trained **YOLO** model with a custom dataset, the system will be able to accurately predict object classes and draw bounding boxes around them. The solution must achieve **high accuracy, low latency, and adaptability** for real-world applications such as surveillance, traffic monitoring, and industrial automation.

* + - 1. **Algorithm Definition**

The project uses the YOLO (You Only Look Once) object detection algorithm, which treats detection as a single regression problem. Instead of applying a classifier to different regions of an image, YOLO processes the entire image in one pass, predicting bounding boxes and class probabilities simultaneously.

**Key Steps:**

1. Input Processing – Resize and normalize the image to YOLO’s required dimensions.
2. Feature Extraction – Pass the image through a convolutional neural network (CNN) to extract spatial features.
3. Bounding Box Prediction – Divide the image into an S×S grid, where each grid cell predicts bounding box coordinates, objectless score, and class probabilities.
4. Non-Maximum Suppression (NMS) – Remove overlapping boxes to keep only the most accurate predictions.
5. Output – Display detected objects with bounding boxes and confidence scores in real time.

YOLO’s single-shot architecture allows high-speed detection while maintaining good accuracy, making it ideal for time-sensitive applications.

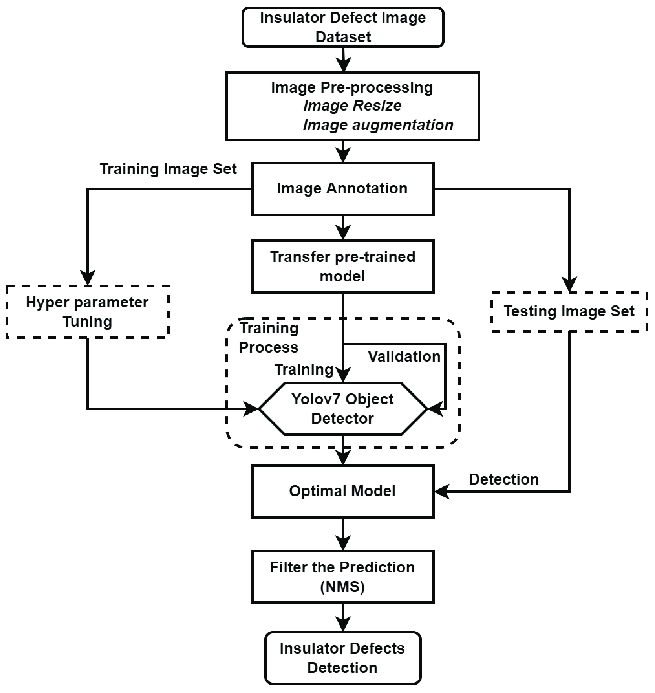
## Experimental Evaluation:

* + - 1. **Methodology:**

T The development of the Object Detection system using the YOLO model follows these steps:

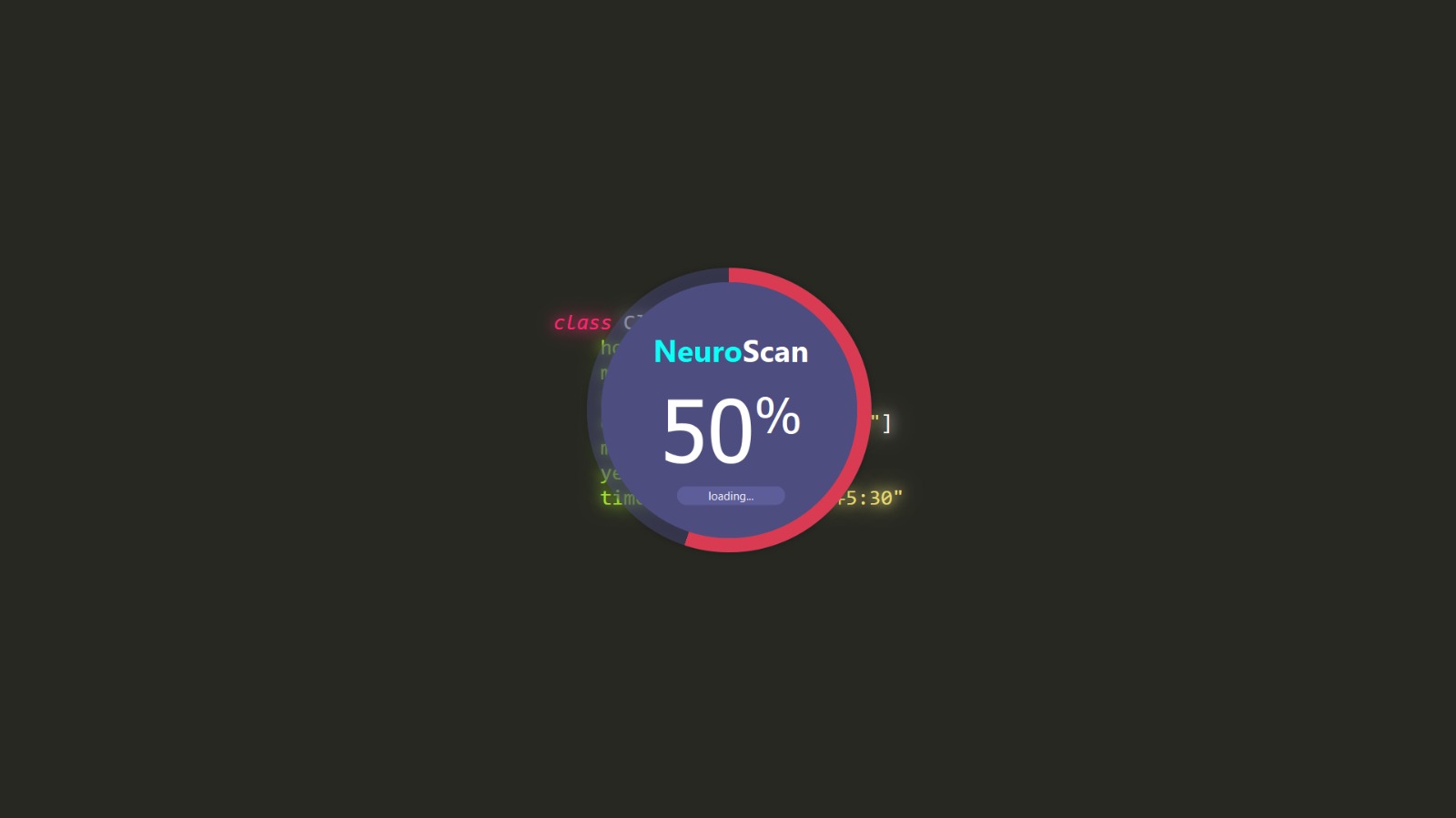
1. **Data Collection and Preparation**
   * Gather a **custom dataset** containing images of the target objects.
   * Annotate images using **Labelling** to generate bounding box coordinates and class labels.
   * Split the dataset into **training (80%)**, **validation (10%)**, and **testing (10%)** sets.
2. **Data Preprocessing**
   * Resize all images to match YOLO input size (e.g., 416×416 pixels).
   * Normalize pixel values for faster convergence.
   * Ensure annotation files are in YOLO format.
3. **Model Selection and Training**
   * Use a **pre-trained YOLO model** (e.g., YOLOv5) as the base.
   * Fine-tune the model on the custom dataset to improve detection accuracy.
   * Monitor performance using metrics like **Precision, Recall, and mAP**.
4. **Model Evaluation**
   * Test the model on unseen images to evaluate accuracy and detection speed.
   * Use confusion matrix and loss curves to assess model performance.
5. **Deployment and Testing**
   * Implement the trained model for **real-time detection** using webcam or video feed.
   * Optimize for faster inference using GPU acceleration.

**Flow Diagram :**

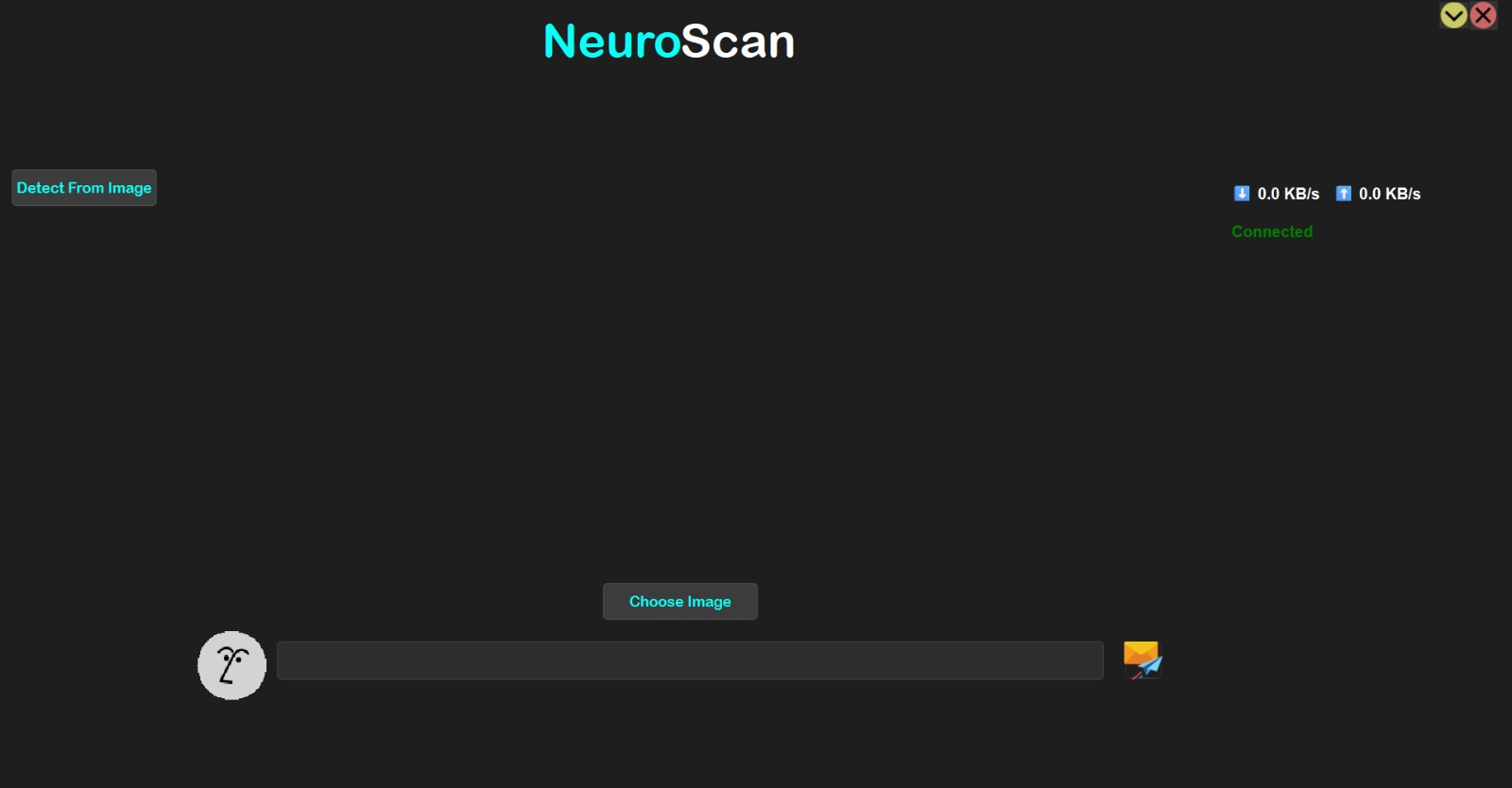


**4. Results and discussion:**

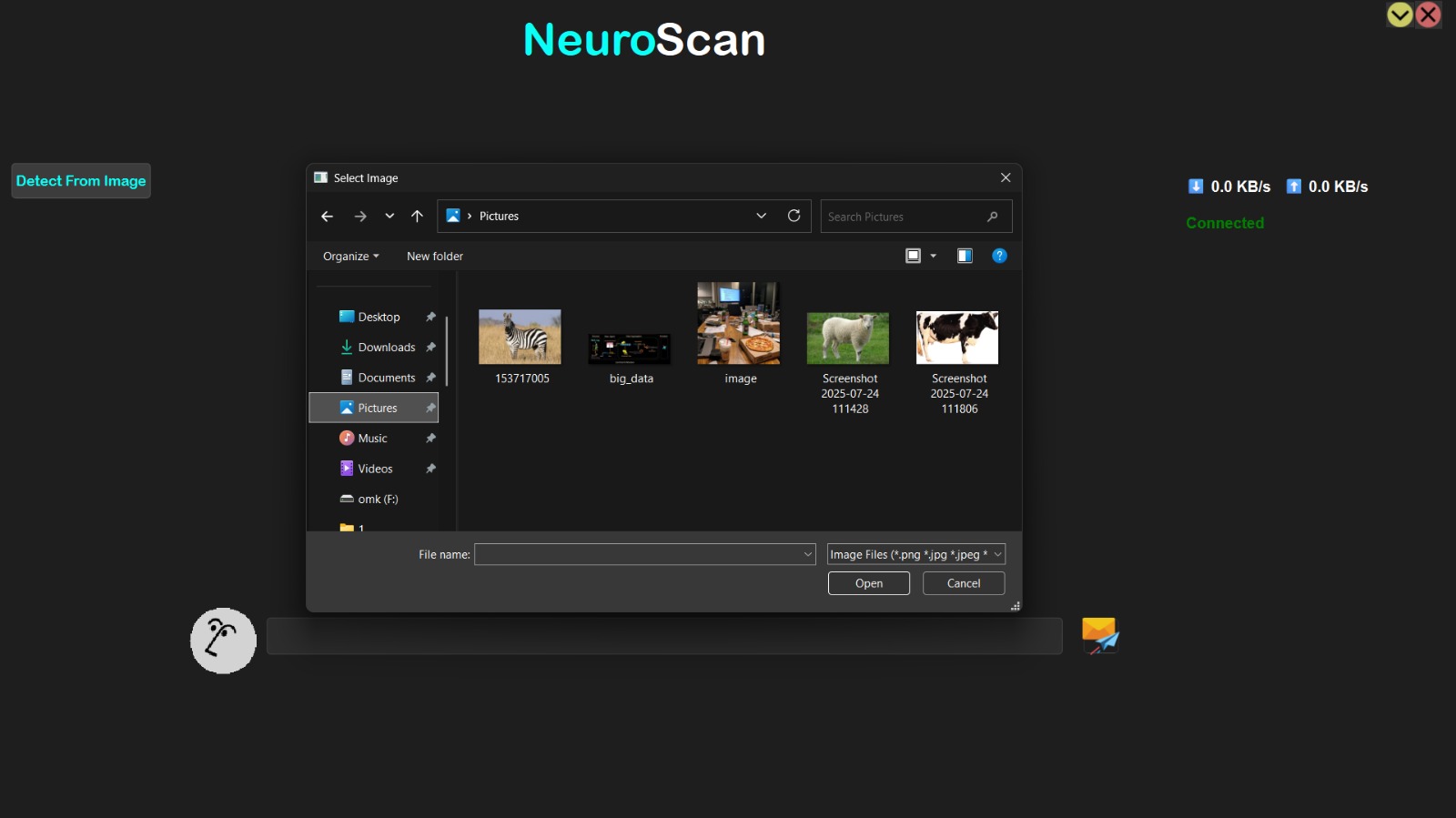
**4.1 Results**



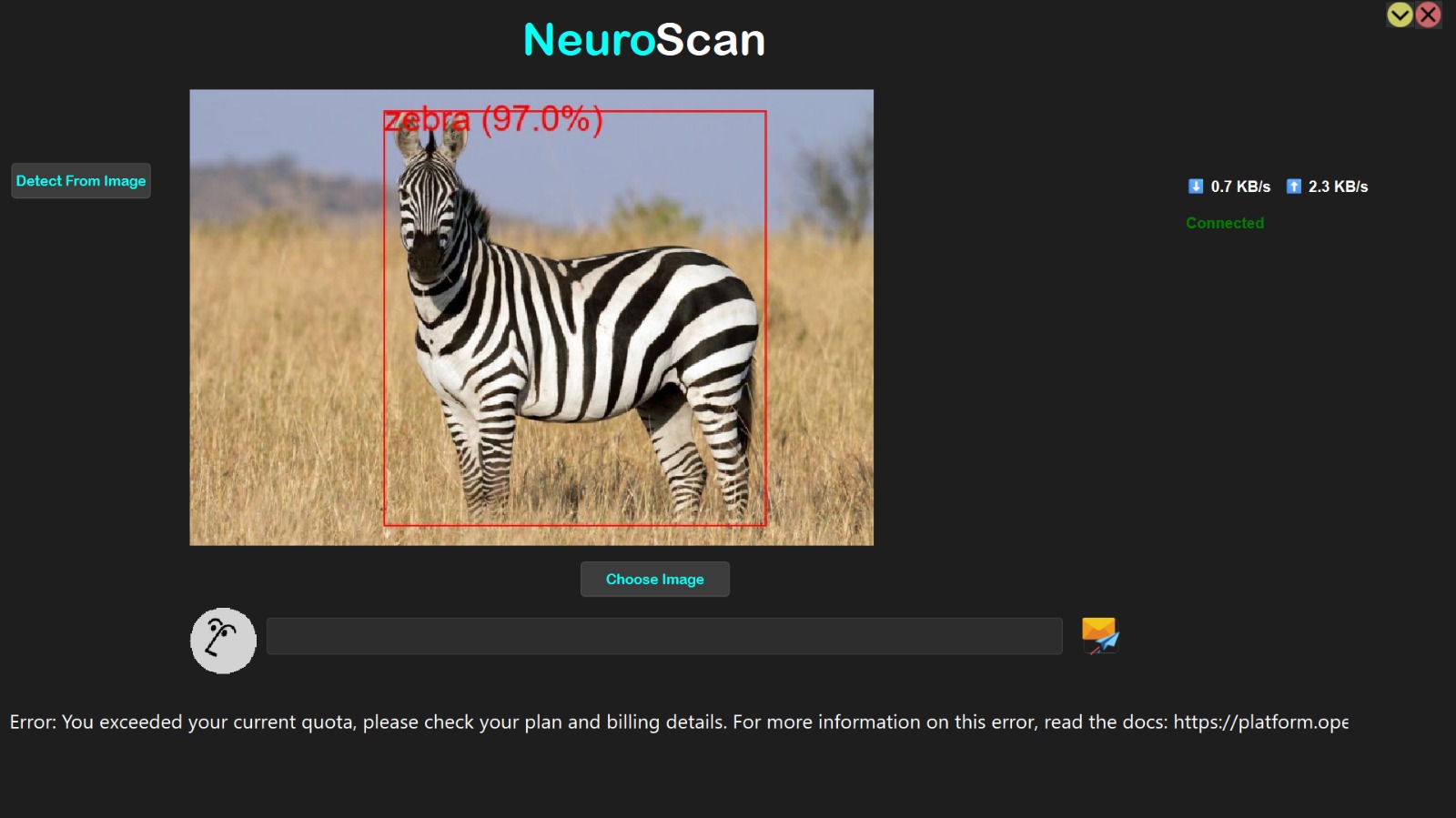
**Fig. Output 01**



**Fig. Output 02**



**Fig. Output 03**



**Fig. Output 04**

**4.2 Discussion on Output**

The object detection system accurately identified and localized objects in images and real-time video streams using the fine-tuned YOLO model. The results were displayed with bounding boxes, class labels, and confidence scores for clarity. The GUI built with PySide6 and Qt Designer allowed easy selection of files and direct visualization of detection results. The system performed well in terms of speed and accuracy, though slight performance drops were noticed in low-light or partially occluded scenarios. Overall, the output met the project’s goals of accuracy, speed, and user-friendliness.

**5. GUI:**

In this object detection project, PySide6 and Qt Designer were used to provide a graphical user interface (GUI) that makes the system more interactive and user-friendly. Instead of running the detection process through command-line instructions, the GUI allows users to perform tasks such as selecting images or videos, starting real-time detection, and viewing results with ease.

* PySide6 was chosen because it offers a cross-platform, modern, and responsive interface with a wide range of built-in widgets suitable for complex applications.
* Qt Designer simplified the design process by allowing the interface to be built visually through drag-and-drop, reducing development time and keeping the layout consistent.
* The separation of frontend (GUI) and backend (YOLO object detection logic) ensures better maintainability and scalability of the project.

By integrating PySide6 and Qt Designer, the project not only delivers accurate object detection results but also ensures that non-technical users can operate it without needing to understand the underlying code.

**6. GitHubLink:**

<https://github.com/Aniket0727/PG-DBDA-ML-Project.git>

**7. Future work And Conclusion**

## 7.1 Future Work:

* Expand the dataset with more diverse and complex images to improve model robustness.
* Integrate **object tracking** for continuous monitoring in video streams.
* Optimize the model for **edge devices** like Raspberry Pi or Jetson Nano.
* Deploy as a **web or mobile application** for remote access and usage.
* Implement **multi-class detection** for broader object categories.
* Use **transfer learning** with the latest YOLO versions (YOLOv8/YOLO-NAS) for better accuracy and speed.

## 7.2 Conclusion:

This project successfully demonstrates the implementation of a **real-time object detection system** using the YOLO algorithm, fine-tuned on a custom dataset. The model achieved accurate and fast detection, making it suitable for practical applications such as surveillance, automation, and quality inspection. By leveraging YOLO’s single-shot detection approach, the system efficiently identifies and locates objects in both images and live video streams.

The results highlight the potential of deep learning–based object detection in solving real-world problems where **speed, accuracy, and adaptability** are essential. With further improvements in dataset diversity, model optimization, and deployment, this project can be extended into a fully scalable solution for various industries.