iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.71908

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

AI- Powered Smart Object Counting System Using Machine Learning and Real-Time Image Analysis

K Tulasi Krishna Kumar[1], Yanamareddy Gopal Krishna[2]
[1]*Assistant Professor, Training and Placement officer,* [2]*MCA Final Semester, Master of Computer Applications, Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India*

Abstract: ***The rapid advancement of artificial intelligence (AI) and computer vision has enabled the automation of tasks that traditionally required manual supervision. This paper presents the development of a smart object counting system powered by machine learning and real-time image analysis. The proposed system uses a pre-trained YOLO (You Only Look Once) model for efficient object detection from a live video stream, enabling accurate and real-time counting of targeted objects. Implemented using Python, OpenCV, and PyTorch, the system offers high-speed performance and can be customized for various object classes such as people, vehicles, or products. The real-time capability and flexibility make it suitable for applications in surveillance, traffic management, manufacturing, and retail analytics. The system is lightweight and can run on standard hardware, making it practical for deployment in resource-constrained environments. This paper details the architecture, implementation methodology, and performance evaluation of the system, and discusses possible enhancements for future development.***
Index Terms: Object Detection, Real-Time Image Processing, YOLO, Smart Counting System, Machine Learning, Computer Vision, OpenCV, Surveillance, Deep Learning, Video Analytics.

I. INTRODUCTION

In recent years, the integration of artificial intelligence (AI) with computer vision has led to significant advancements in automation across various domains. Object counting, a fundamental task in applications such as crowd monitoring, inventory tracking, vehicle detection, and manufacturing, has traditionally relied on manual efforts or simple sensor-based systems, which often lack accuracy and scalability.This project aims to design and implement a smart object counting system using AI-driven techniques and real-time image analysis. By leveraging a pre-trained YOLOv5 model and integrating it with OpenCV, the system captures video frames from a live camera feed, detects specified objects, and counts them in real-time. The primary goal is to develop a lightweight and efficient system that can operate on general-purpose laptops without requiring high-end hardware.

A. Existing System

In the current landscape, several traditional methods are used for object counting across different domains. These existing systems are based on sensor technologies, manual monitoring, and classical image processing algorithms.One of the most widely used methods is the sensor-based counting system, which relies on devices such as infrared (IR) sensors, pressure pads, or ultrasonic sensors. These are often deployed at entry and exit points to detect and count the number of objects or people passing through. While cost-effective, these systems are limited in scope as they cannot identify object types, detect overlapping objects, or function accurately in high-traffic or dynamic environments

Classical computer vision techniques, such as motion detection, background subtraction, and contour detection, have been applied for object counting in video streams. However, these methods are heavily affected by environmental changes, such as varying light conditions, shadows, occlusions, and background clutter. Their ability to detect and distinguish between multiple object classes is very limited.

In traffic monitoring, loop detectors and magnetic sensors are embedded in roads to count vehicles. These are accurate but expensive to install and maintain, and are not suitable for Due to these drawbacks, traditional object counting systems fail to meet the needs of modern applications such as smart surveillance, automated inventory management, and AI-based analytics.This highlights the need for a more robust, scalable, and intelligent system that can operate effectively in real time while adapting to various environments and object types. The proposed AI-powered system addresses these issues by leveraging machine learning and deep learning techniques for high-accuracy, real-time object detection and counting.

©IJRASET: All Rights are Reserved

1) Challenges:

- Real-Time Performance ;*Achieving accurate object detection and counting in real time requires high-speed processing. Without a dedicated GPU, real-time inference on live video streams can become slow, affecting the overall system responsiveness.*
- Accuracy in Complex Environments :Environments with poor lighting, shadows, cluttered backgrounds, or camera motion can lead to false detections or missed objects. The model must be robust enough to perform reliably across different conditions.
- Overlapping and Occluded Objects :When multiple objects are close together, partially hidden, or overlapping, it becomes difficult for the system to correctly detect and count them without duplication or omission.
- Model Generalization :A pre-trained object detection model may not always detect specific objects with high accuracy unless it is trained or fine-tuned on a dataset closely related to the target environment or object types.
- Limited Computing Resources :Many real-world deployments require the system to run on laptops or edge devices with limited computing power and memory. Optimizing model size and efficiency without compromising accuracy is a significant challenge.
- Dataset Requirements ;Training or fine-tuning a model requires a large and well-annotated dataset. Collecting and labeling such data can be time-consuming and resource-intensive.
- Multi-Class Counting Complexity ;Counting multiple types of objects simultaneously increases the complexity of the system. Ensuring accurate class-wise detection and counting is more challenging than single-object detection.
- Integration and Deployment :Deploying the system as a complete solution may require integration with camera hardware, cloud platforms, or local databases, which can introduce compatibility and maintenance issues.

B. Proposed system:

To overcome the limitations of existing object counting methods, the proposed system introduces an AI-powered solution that combines deep learning-based object detection with real-time image processing techniques. The system is designed to detect and count objects from a live video feed or camera input using a pre-trained YOLO (You Only Look Once) model, ensuring both high accuracy and real-time performance.The proposed system captures frames from a live camera or video stream and processes them using a deep learning model to detect and classify objects. Once detected, the system applies logic to count the number of instances of a target object class in each frame and maintain a running count. The results are displayed visually on the video stream with bounding boxes and object labels, and the count is updated dynamically.

1) Advantages:

- Real-Time Object Counting

The system processes live video streams and provides instant object detection and counting, making it suitable for dynamic and time-sensitive applications

- High Accuracy

By using state-of-the-art deep learning models like YOLOv5, the system achieves high detection and counting accuracy, even in complex scenes.

- Automation and Efficiency

Eliminates the need for manual counting or supervision, reducing human errors and saving time and labor costs.

- Multi-Object Detection

Capable of identifying and counting multiple object types (e.g., people, vehicles, products) simultaneously in a single frame.

- Scalable and Flexible

Can be adapted for various use cases such as traffic monitoring, retail analytics, warehouse inventory, classroom attendance, etc.

- Hardware-Friendly

Designed to run on standard laptops without requiring expensive GPUs, making it accessible for educational, research, or small business purposes.

- Customizable and Extensible

The system can be trained or fine-tuned on custom datasets to support new object classes or specific environments.

- Visual Feedback

Provides real-time visual output with bounding boxes, labels, and count overlays, which helps in easy monitoring and verification.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

- ·Portable and Lightweight

Can be deployed easily on different platforms like laptops, Raspberry Pi, or edge devices with minimal setup.

- Open-Source Friendly

Leverages open-source libraries such as YOLOv5, OpenCV, and PyTorch, enabling easy customization and development

II. ARCHITECTURE

The architecture of the AI-powered smart object counting system is designed in a modular way to ensure efficient real-time performance and scalability. It begins with the hardware layer, where a webcam or IP camera is used to capture live video feeds. These frames are immediately passed to the video processing unit. The captured video frames undergo pre-processing which includes resizing, normalization, and format conversion to ensure compatibility with the deep learning model. Once pre-processed, each frame is fed into a pre-trained YOLOv5 model, which serves as the object detection engine. The YOLO model analyzes each frame and returns bounding boxes, class labels, and confidence scores for every detected object. These detections are passed to a filtering module, where only the relevant object classes (e.g., people, vehicles, or custom targets) are selected for counting. The filtered results are then processed by the counting logic, which keeps track of how many target objects appear in each frame. Optional features like region of interest (ROI) detection or object tracking may also be integrated here to prevent double-counting. Once counted, the output is passed to the display module, which overlays bounding boxes, labels, and the live count directly on the video stream. The system uses OpenCV for visual display and real-time interaction. It is optimized to run on standard laptop hardware using only CPU resources, though it can also leverage GPU for faster processing if available. The architecture ensures that each module operates independently, allowing easy updates or replacements in the future. This flow from camera to display forms a complete pipeline from input to output. The real-time feedback loop makes the system highly suitable for live monitoring tasks. It is also lightweight enough to be deployed in embedded systems or edge devices. By using open-source tools and modular design, the system remains flexible, portable, and easily extendable.

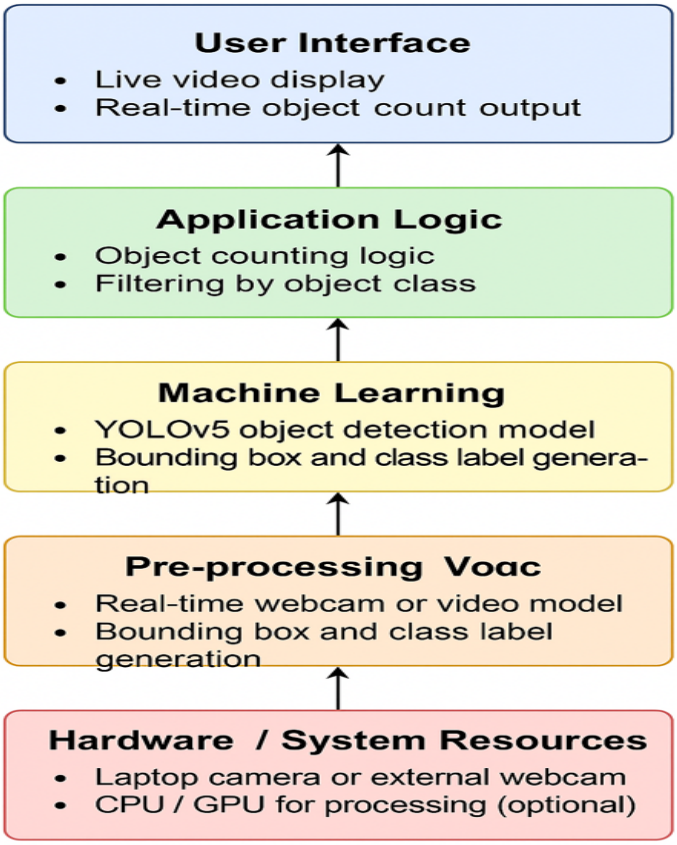


Fig:2 Architecture

A. Algorithm:

The algorithm for the AI-powered smart object counting system begins with the initialization of the YOLOv5 object detection model and the video capture device, such as a webcam or external camera. Once initialized, the system continuously reads video frames in real time. Each frame is first pre-processed by resizing it to match the input dimensions required by the model, followed by normalization of pixel values and, if necessary, conversion of the frame's color format (e.g., BGR to RGB). The pre-processed frame is then passed to the YOLOv5 model, which performs object detection and returns bounding boxes, class labels, and confidence scores for each detected object. The system filters these detections based on a defined confidence threshold and the specific object classes of interest, such as persons or vehicles. After filtering, the algorithm counts the number of valid objects in the frame. This count is then overlaid on the video feed along with the bounding boxes and class labels to provide a real-time visual representation of the detection and counting results. The process repeats for each incoming frame until the user terminates the video stream. Finally, the system releases all resources, including the video feed and display windows, ensuring a clean and efficient shutdown

B. Techniques:

The AI-powered smart object counting system leverages a combination of computer vision, deep learning, and real-time image processing techniques to achieve accurate and efficient object detection and counting. The primary technique employed is **object detection using the YOLOv5 (You Only Look Once)** algorithm, a state-of-the-art deep learning model that can detect multiple objects in a single frame with high speed and accuracy. YOLOv5 operates by dividing each input image into a grid and predicting bounding boxes and class probabilities for each grid cell. This one-stage detection approach allows for real-time performance even on modest hardware like standard laptops. Alongside this, the project utilizes **image pre-processing techniques** such as resizing, normalization, and color space conversion to ensure the input frames are compatible with the model and enhance detection quality. Additionally, **OpenCV** is used for frame handling, drawing bounding boxes, displaying labels, and visualizing live output. The system may also incorporate **thresholding techniques** to filter out low-confidence predictions and focus only on reliable detections. Optionally, **basic object tracking** techniques like centroid tracking or frame-to-frame ID mapping can be used to avoid double-counting objects across sequential frames. Together, these techniques form a robust pipeline for detecting, filtering, and counting objects in real-time video streams.

C. Tools:

The development of the AI-powered smart object counting system involves a variety of software tools and libraries that collectively support real-time image analysis and machine learning functionalities. The core detection model is built using **YOLOv5**, an open-source object detection algorithm implemented in **Python** and based on the **PyTorch** deep learning framework. PyTorch is preferred for its dynamic computation graph and ease of customization. **OpenCV** (Open Source Computer Vision Library) plays a crucial role in video capture, image pre-processing, and displaying annotated video output with bounding boxes and labels. The entire system is typically developed and executed in an Integrated Development Environment (IDE) such as **Visual Studio Code (VS Code)** or **Jupyter Notebook**, providing a convenient interface for writing and testing Python code. For handling dependencies and packages, **pip** or **Anaconda** is used to manage libraries like numpy, matplotlib, and other required modules. In some cases, **Google Colab** can be used to test the model with GPU support, especially when the local system lacks GPU resources. Additional tools like **LabelImg** may be employed if custom object datasets are used, allowing for manual annotation and preparation of training data. Together, these tools create a comprehensive and flexible environment for building and deploying the real-time smart object counting system.

D. *Methods*:

The AI-powered smart object counting system utilizes a sequence of well-defined methods to achieve accurate and efficient object detection and counting in real time. The process begins with **video acquisition**, where a continuous stream of frames is captured from a webcam or external camera. Each frame is then passed through a **pre-processing method** that includes resizing the image to match the input dimensions expected by the detection model, normalizing pixel values, and converting color formats (e.g., from BGR to RGB). Once pre-processed, the frame is fed into the **YOLOv5 object detection model**, a deep learning-based method known for its speed and accuracy. YOLOv5 analyzes the image and applies its convolutional neural network architecture to detect and classify objects, outputting bounding boxes and confidence scores. These detections are then passed to a **filtering method**, where only relevant object classes are retained based on predefined criteria, and low-confidence detections are discarded. After filtering, a **counting method** is applied, which tallies the number of valid objects per frame.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Optionally, a **tracking method** can be integrated to assign unique IDs to objects and avoid double-counting across frames. Finally, a **visualization method** is used to draw bounding boxes and display the object count on the video stream, offering a complete and interactive real-time monitoring system. Each of these methods works in coordination to deliver a robust and responsive object counting solution.III.

III. METHODOLOGY

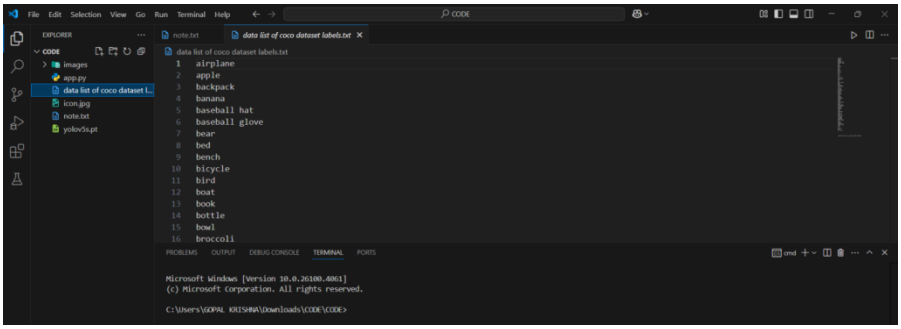
A. Input:

The primary input to the AI-powered smart object counting system is a **real-time video stream** captured through a **webcam, IP camera, or CCTV** device. This video feed provides continuous frames that are used for object detection and counting. Each frame acts as an input image containing various objects, such as people, vehicles, or other custom-defined targets. These frames are dynamically captured in real time and serve as raw input data for further processing. The system does not require any static or pre-recorded input images, making it suitable for live monitoring environments. In addition to the video feed, **configuration parameters** such as the object classes to detect (e.g., "person", "car"), detection confidence thresholds, and frame size may also be considered as part of the input settings. These parameters allow the system to be customized for specific use cases. Optionally, if a custom dataset is used for training or fine-tuning the model, **labeled image data** may also be considered as offline input during the development or training phase. Overall, the input to this system is dynamic, real-time visual data along with configurable parameters that guide the behavior of the object detection and counting logic.

- App.py



- Coco data set



B. Method of Process:

The method of process in the AI-powered smart object counting system begins with the **real-time acquisition of video frames** from a live camera feed. Each captured frame is immediately passed through a **pre-processing stage**, where the image is resized, normalized, and converted to the appropriate color format suitable for the detection model. Once pre-processed, the frame is fed into the **YOLOv5 object detection model**, which uses a convolutional neural network to detect and classify multiple objects within the image. The model outputs bounding boxes, object class labels, and associated confidence scores. In the next step, a **filtering mechanism** removes low-confidence detections and selects only the target object classes (such as "person", "car", etc.) as defined by the user. The filtered detections are then passed to the **counting logic**, which computes the number of valid objects present in each frame. Optionally, **tracking algorithms** can be integrated to maintain object identity across frames and avoid duplicate counts.

The final results, including bounding boxes and object counts, are then **rendered visually on the video stream** using OpenCV, providing the user with real-time feedback. This process continues in a loop until the user terminates the system, ensuring continuous detection, counting, and visualization of objects in a live environment.

C. Output:

The output of the AI-powered smart object counting system is a **real-time annotated video stream** displaying the detection and count of specific objects within each frame. The system overlays **bounding boxes** around the detected objects and labels them with their respective class names (e.g., "person", "car") and confidence scores. In addition to the visual annotations, a **live object count** is prominently displayed on the screen, continuously updating as new frames are processed. This allows users to monitor the number and type of objects present in the camera's field of view at any given moment. The system provides immediate feedback, making it suitable for real-time monitoring in applications like surveillance, crowd analysis, traffic management, and retail analytics. Optionally, the output can also be saved in the form of **detection logs, screenshots, or video recordings** for future analysis. The output is designed to be clear, accurate, and responsive, ensuring that users receive meaningful information without delay







IV. RESULTS

The system successfully detects and counts objects in real time using a live video feed.
It displays bounding boxes, object labels, and an updated count on the screen continuously.
The results demonstrate high accuracy and responsiveness on standard laptop hardware.

V. DISCUSSIONS

The AI-powered smart object counting system demonstrates the effective integration of deep learning and computer vision techniques to address real-world monitoring and analytics challenges. The use of the YOLOv5 model provides a balance between speed and accuracy, enabling real-time detection and counting on standard computing hardware without the need for expensive GPUs. Throughout the implementation, the system effectively handles different object sizes, overlapping objects, and varying lighting conditions in the input video stream. However, certain limitations were observed, such as decreased accuracy in low-light or noisy environments, and occasional false positives in cluttered scenes. These challenges highlight the importance of proper camera placement and model tuning based on the application scenario. The modular design of the system makes it adaptable, allowing enhancements such as object tracking, edge device deployment, or integration with cloud-based analytics platforms. Overall, the results support the viability of using AI-based solutions for automated object counting, offering a scalable and efficient alternative to manual monitoring systems.

VI. CONCLUSION

In conclusion, the AI-powered smart object counting system effectively demonstrates how machine learning and real-time image processing can be combined to automate the task of object detection and counting. By leveraging the YOLOv5 model and OpenCV for video stream handling and visualization, the system achieves accurate and real-time results suitable for a variety of applications such as surveillance, traffic analysis, and retail management. The implementation on standard hardware, such as a laptop, confirms the system's efficiency and practical usability without requiring high-end computational resources. Despite minor limitations in complex or low-light environments, the system proves to be a reliable and scalable solution for real-time monitoring. This project not only showcases the power of AI in visual data analysis but also opens up pathways for future enhancements like object tracking, multi-camera integration, and deployment on embedded systems for edge computing.

VII.FUTURE SCOPE

The AI-powered smart object counting system holds significant potential for future development and expansion across various domains. One promising direction is the integration of advanced tracking algorithms, which can help maintain consistent object identities across frames, reducing the chances of double-counting. Additionally, the system can be enhanced with multi-class detection capabilities, allowing it to simultaneously count and analyze multiple object types with greater precision. Deployment on edge devices like Raspberry Pi, Jetson Nano, or mobile platforms can make the system more portable and accessible for field operations. Incorporating cloud-based analytics can enable centralized data storage, remote monitoring, and long-term trend analysis. Further improvements in the training dataset, including more diverse environments and lighting conditions, can enhance the model’s robustness in real-world scenarios. Lastly, integrating the system with IoT sensors and alert mechanisms can transform it into a fully automated smart surveillance solution with applications in security, industrial automation, and smart city infrastructure.

VIII. ACKNOWLEDGEMENT

Kandhati Tulasi Krishna Kumar Nainar: Training & Placement Officer with 16 years’ experience in training & placing the students



into IT, ITES & Core profiles & trained more than 9,700 UG, PG candidates & trained more than 450 faculty through FDPs. Authored various books for the benefit of the diploma, pharmacy, engineering & pure science graduating students. He is a Certified Campus Recruitment Trainer from JNTUA, did his Master of Technology degree in CSE from VTA and in process of his Doctoral research. He is a professional in Pro-E, CNC certified by CITD He is recognized as an editorial member of IJIT (International Journal for Information Technology & member in IAAC, IEEE, MISTE, IAENG, ISOC, ISQEM, and SDIWC. He published 6 books, 55 articles in various international journals on Databases, Software Engineering, Human Resource Management and Campus Recruitment & Training

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

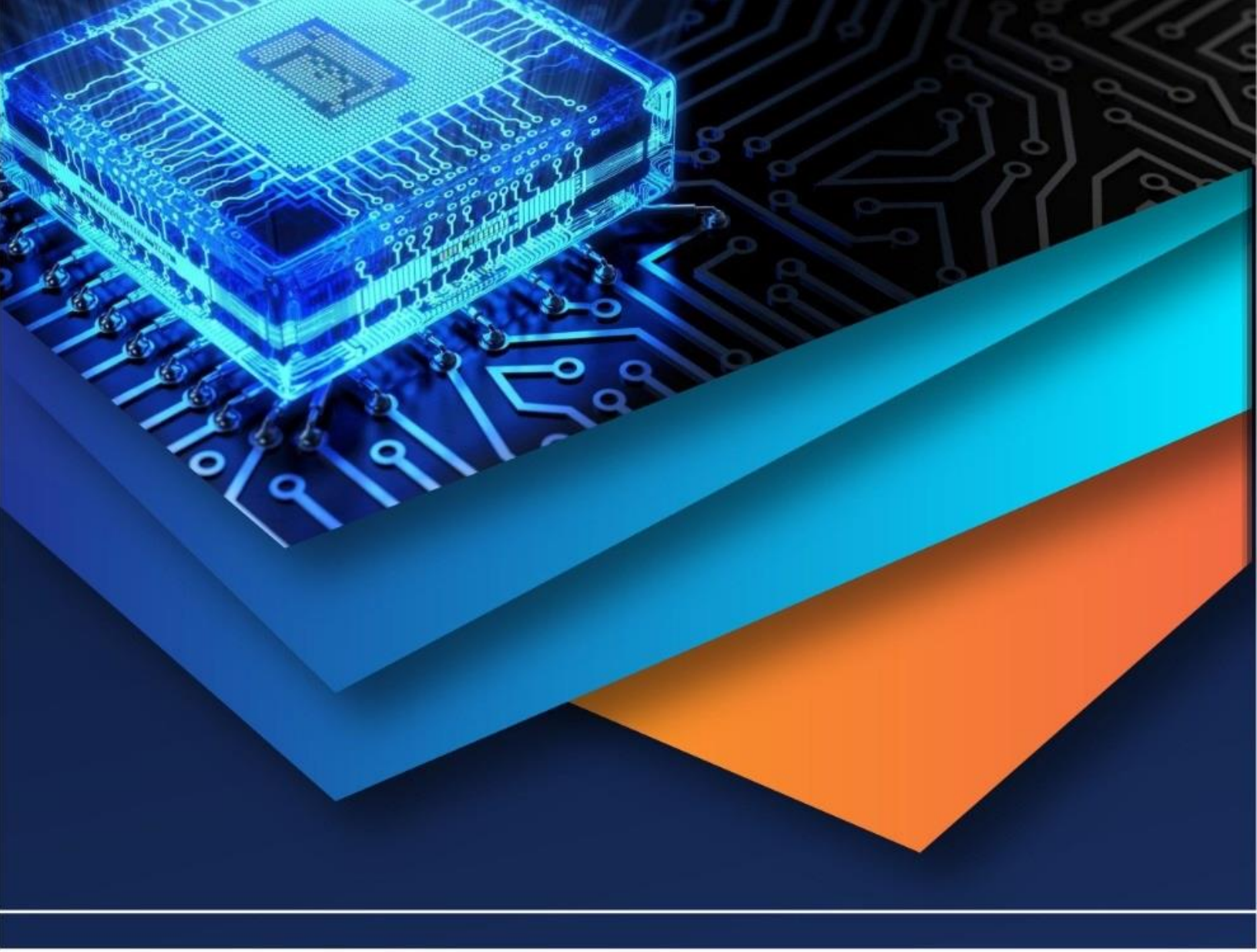


Yanamareddy Gopal Krishna is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine Learning Y GOPAL KRISHNA has taken up his PG project on AI Powered Smart Object Counting system Using Machine Learning and Real Time Image Analysis and published the paper in connection to the project under the guidance of K. T. Krishna Kumar Nainar, Training & Placement Officer, SVPEC.

REFRENCES

[1] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and* Accuracy of Object Detection https://arxiv.org/abs/2004.10934
[2] Ultralytics. (2023). YOLOv5 Official GitHub Repository. https://github.com/ultralytics/yolov5
[3] OpenCV. (2023). Open Source Computer Vision Library https://opencv.org
[4] PyTorch. (2023). An Open Source Machine Learning Framework https://pytorch.org
[5] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement https://arxiv.org/abs/1804.02767
[6] LabelImg. (2023). A Labeling Tool for Image Annotation. https://github.com/tzutalin/labelImg
[7] Google Colab. (2023). Free Cloud GPU for Machine Learning https://colab.research.google.com
[8] Roboflow. (2023). Create and Annotate Object Detection Datasets https://roboflow.com
[9] Kaggle. (2023). Free Datasets and Notebooks for Object Detection https://www.kaggle.com
[10] Machine Learning Mastery. (2021). *Introduction to Object Detection* https://machinelearningmastery.com
[11] Papers with Code. (2023). State-of-the-Art YOLO Models Benchmarks https://paperswithcode.com/task/object-detection
[12] TensorFlow Object Detection API. (2023). https://github.com/tensorflow/models/tree/master/research/object_detection
[13] Scikit-learn. (2023). Machine Learning in Python. https://scikit-learn.org
[14] NumPy. (2023). Scientific Computing with Python https://numpy.org
[15] Matplotlib. (2023). Plotting and Visualization Library https://matplotlib.org
[16] Towards Data Science. (2022). Guide to YOLO Object Detection. https://towardsdatascience.com
[17] CVPR. (2022). Top Papers on Real-Time Object Detection https://cvpr2022.thecvf.com
[18] Visual Studio Code. (2023). Free Source Code Editor. https://code.visualstudio.com
[19] Jetson Nano by NVIDIA. (2023). Edge AI Platform for YOLO. https://developer.nvidia.com/embedded/jetson-nano
[20] Edge Impulse. (2023). Deploy AI Models on Embedded Device https://www.edgeimpulse.com
[21] ArXiv.org. (2023). Research Papers on Real-Time Vision AI. https://arxiv.org
[22] AIHub. (2023). Object Detection Research Resources. https://aihub.cloud
[23] GitHub – Object Counting Systems. https://github.com/topics/object-counting
[24] LearnOpenCV.com. (2023). YOLO and OpenCV Tutorials https://learnopencv.com
[25] Analytics Vidhya. (2023). YOLO Explained – A Beginner's Guide https://www.analyticsvidhya.com

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)