VEHICLE DETECTION, CLASSIFICATION & USING OPENCV & PYTHON

Author: Yawalkar Ajinkya Ganpati
Department: Electronics and Communication Engineering
Affiliation: Indian Institute of Technology, Roorkee
email: y_aganpati@ece.iitr.ac.in

***Abstract*— This project presents an advanced system for vehicle detection, classification, and counting utilizing computer vision techniques and deep learning. Leveraging the YOLOv3 model and OpenCV, the system efficiently identifies vehicles in real-time video streams or static images. The tracking algorithm employs the Euclidean distance concept, ensuring robust object tracking across frames. The project provides a comprehensive overview of the entire process, from initializing the network and preprocessing frames to post-processing detection outputs and tracking vehicle counts. The system demonstrates its effectiveness in monitoring traffic, offering a valuable tool for traffic management and analysis.**

***Keywords*— vehicle detection, classification, counting, YOLOv3, OpenCV, computer vision, deep learning, Euclidean distance, object tracking, traffic analysis.**

I. MOTIVATION AND RELEVANCE

Modern traffic management systems rely heavily on advanced computer vision and deep learning techniques for efficient monitoring and analysis. This project introduces a robust solution for vehicle detection, classification, and counting, leveraging the power of the YOLOv3 model and OpenCV. With a focus on real-time video streams and static images, the system employs innovative tracking algorithms based on Euclidean distance, ensuring accurate and persistent object tracking. The following sections present a comprehensive overview of the project, detailing the key methodologies, algorithms, and steps involved in creating an effective tool for traffic analysis and management. The integration of these technologies aims to contribute to the evolution of intelligent transportation systems and enhance our ability to understand and optimize traffic flow.

II. EASE OF USE

A. *Selection of Technology Stack*

Choosing an appropriate technology stack is fundamental to the success of any computer vision project. In parallel to selecting the right template for document formatting, we meticulously chose OpenCV, YOLOv3, and tracking algorithms to build a robust vehicle detection and classification system. The integration of these tools serves as the backbone of our project, ensuring seamless and effective processing.

B. *Adherence to System Specifications*

Our project strictly adheres to predefined algorithms and methodologies. The YOLOv3 model, OpenCV configurations, and tracking algorithms act as the specifications for our system. Any deviation from these standards could compromise the integrity of the entire vehicle detection and classification system. This approach ensures a standardized and reliable solution for analyzing and monitoring traffic scenarios.

III. ABOUT YOLOV3

What is YOLO?

1. YOLO stands for You Only Look Once. It is a real-time object recognition algorithm. It can classify and localize multiple objects in a single frame.
2. YOLO is a very fast and accurate algorithm for its simpler network architecture.

How does YOLO work?

YOLO works mainly using these techniques.
1. Residual Blocks – Basically, it divides an image into NXN grids.

2. Bounding Box regression – Each grid cell is sent to the model. Then YOLO determines the probability of the cell containing a certain class and the class with the maximum probability is chosen.

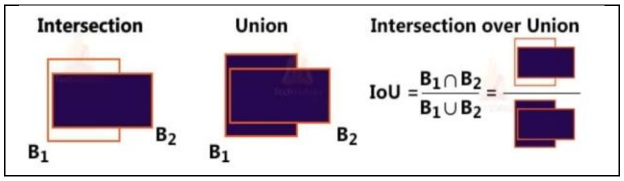


Fig. IOU using set theory explanation (source)

3. Intersection Over Union (IOU) – IOU is a metric that evaluates the intersection between the predicted bounding box and the ground truth bounding box. A Non-max suppression technique is applied to eliminate the bounding boxes that are very close by performing the IoU with the one having the highest class probability among them.

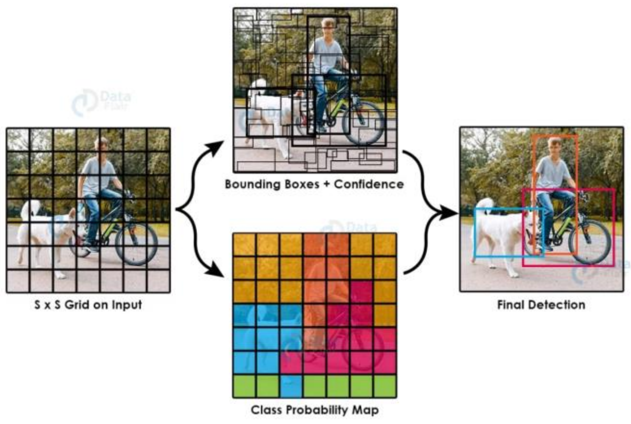


Fig. Intersection over Union (source)

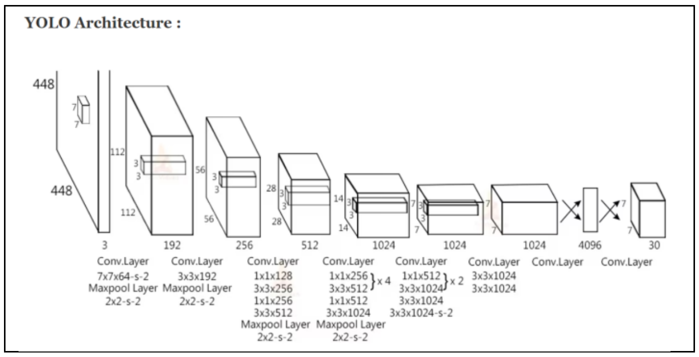


Fig. YOLO Architecture (source)

1. The YOLO network has 24 convolutional layers followed by 2 fully connected layers. The convolutional layers are pre-trained on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.
2. The layers Alternating 1 × 1 reduction layer and 3×3 convolutional layers to reduce the feature space from preceding layers.
3. The last 4 layers are added to train the network for object detection.
4. The last layer predicts the object class probability and the bounding box probability.
5. In this project, we have used OpenCV's DNN module to work with YOLO directly. DNN means Deep Neural Network. OpenCV has a built-in function to perform DNN algorithms.

IV. About Project Work

Work is divided into the following sections:
1. Web Scraping for Dataset Preparation
2. Vehicle Detection & Tracking
3. Vehicle Classification & Counting
4. Dependency Installation
5. Data frame creation

In this project, I have detected and classified cars, buses, trucks, and motorbikes on the road, and counted the number of vehicles traveling through a road. And the data will be stored to analyze different vehicles that travel through the road. We'll create different programs for different functions to do this project.

1. **Tracker** –
The tracker basically uses the Euclidean distance concept to keep track of an object. It calculates the difference between two center points of an object in the current frame vs the previous frame, and if the distance is less than the threshold distance then it confirms that the object is the same object of the previous frame.
Also has a center calculation function to calculate the center of the frame based on coordinates.

2. **Count Vehicle –**
Counts vehicles based on their movement across predefined lines in the frame.
Integrates object positions with designated lines. When a vehicle crosses these lines, the count is updated.
Leverages the tracked object data from the Tracker for accurate counting.

3. **Data Preparation –**
Acquires traffic images for training or testing vehicle detection and classification models.
Utilizes the Beautiful Soup library to parse HTML content and requests to retrieve images from Google search results. The script navigates through image elements, extracts image URLs, and downloads them. This process ensures a diverse set of images for robust model training and testing, enhancing the project's effectiveness in real-world scenarios.

4. **Image Processing –**
Enhances images to improve object detection accuracy by the trained model.
The OpenCV library is employed to preprocess images. This involves resizing, blob creation, and setting input data for the neural network.
The processed image is then fed into the pre-trained YOLOv3 model. Objects are detected, and their bounding boxes, confidence scores, and class IDs are extracted.
Non-maximum suppression (NMS) is applied to filter out redundant bounding boxes.
The results are then post-processed to count and track vehicles, with relevant information stored for further analysis. This ensures accurate vehicle detection and tracking in images.

5. **Video Processing –**
Applies real-time vehicle detection and classification to video streams incorporates functions for a single image and extends it to continuous frames.
The video processing function reads frames from the input video. Each frame is resized and preprocessed for object detection using the YOLOv3 model.

The detected objects are then post-processed to count and track vehicles. Real-time results are displayed on the video frames, and the processed data is saved in a CSV file.
This enables dynamic tracking and classification of vehicles in a video, making it a crucial component for real-world applications like traffic monitoring and analysis.

6. **Dependency –**
Manages necessary libraries, configurations, and auxiliary functions needed for the project.
The dependency file initializes key components, such as the Distance Calculator for object tracking, and configures the YOLOv3 model for vehicle detection.
It sets up color coding, class names, and necessary constants.
By encapsulating these elements, the dependency file ensures a modular and organized structure, facilitating seamless integration of external components into the project.
It acts as a foundational module, ensuring that required dependencies are readily available for other project files, promoting clarity and maintainability.

7. **DataFrame -**
Manages the creation of structured data frames from raw results for better analysis and presentation.
The DataFrame file takes raw results from image or video processing and refines them into organized data frames.
It first reads the cumulative results from CSV files, drops empty rows, and converts columns to numeric types.
It then calculates the difference between consecutive rows and generates a final data frame.
By converting results into a structured format, the file enhances data integrity, facilitates analysis, and simplifies downstream processing.

Prerequisites:
1. Python – 3.9.x
2. OpenCV – 4.4.0
It is strongly recommended to run DNN models on GPU.
You can install OpenCV via “pip install opencv-python opencv_contrib-python”.
3. Numpy – 1.20.3
4. YOLOv3 Pre-trained model weights and Config Files.

Web Scraping for Dataset Preparation

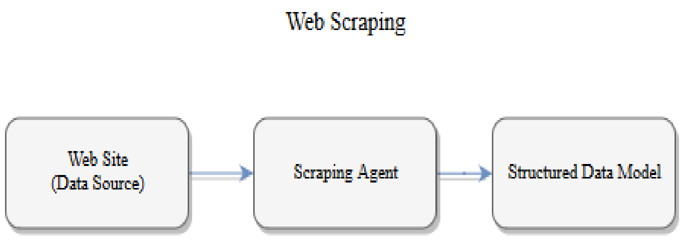
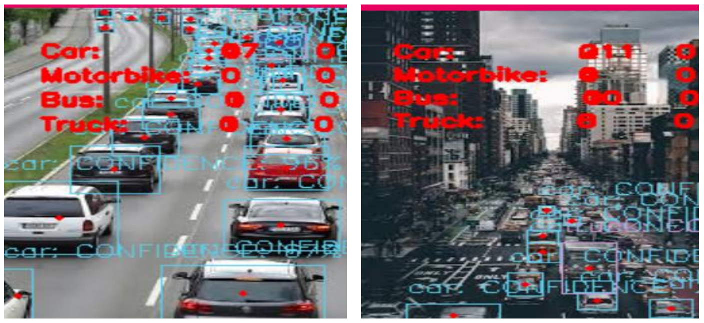


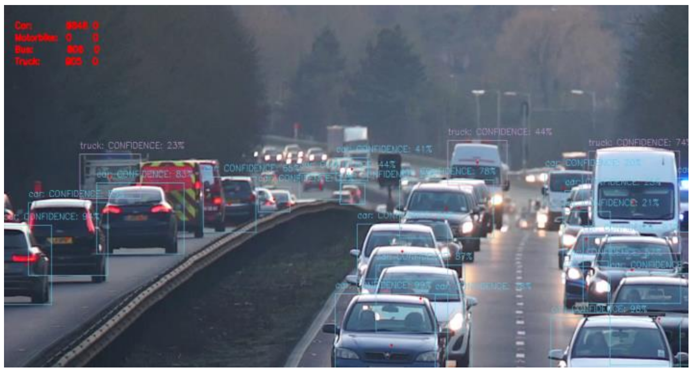
Fig. Web Scraping Flow Diagram (source)

V. Results & Analysis

1. **Image Processing Results:**
Conducted evaluations using diverse datasets to analyze the system's performance.
Utilized a variety of traffic situations and environmental conditions.



2. **Video Processing Results:**



VI. Summary

In this project, we’ve built an advanced vehicle detection and classification system using OpenCV. We’ve used the YOLOv3 algorithm along with OpenCV to detect and classify objects incorporating deep neural networks, file handling techniques, and some advanced computer vision techniques.

VII. Advantages and Limitations

Advantages:
1. Real-time Processing: The project allows for real-time processing of video streams, enabling the detection and classification of vehicles as they move through the frame.
2. Flexibility: The use of Python provides a flexible and extensible platform for further development and integration with other technologies or modules.
3. High Accuracy: Leveraging advanced computer vision techniques, the project can achieve high accuracy in vehicle detection and classification, even in challenging conditions.
4. Customization: The project can be easily customized to cater to specific requirements, such as different types of vehicles or specific environmental conditions.
5. Scalability: The project can be scaled to handle various camera setups, making it suitable for deployment in different

scenarios, such as urban traffic monitoring or highway surveillance.

Limitations:
1. Complex Backgrounds: Busy or cluttered backgrounds might lead to false positives or difficulties in accurately distinguishing vehicles from other objects.
2. Limited Vehicle Types: The accuracy of classification may be limited to a predefined set of vehicle types, and the system may not generalize well to uncommon or new vehicle models.
3. Computational Resources: Real-time processing might demand substantial computational resources, limiting the project's deployment of resource-constrained devices.

VIII. Future Scope

The future scopes of the project align with both the critical needs of the industry, especially in the context of autonomous vehicles and smart city development, and the ongoing trends in technology, such as the integration of deep learning and edge computing.
1. Integration with Autonomous Vehicles
2. Deep learning models, such as Convolutional Neural Networks (CNNs), can provide more accurate and robust vehicle detection.
3. Real-time processing on edge computing devices reduces latency and dependency on centralized servers.
4. Contribute to smart city infrastructure by aiding in traffic management and urban planning

Acknowledgment

I extend my sincere gratitude to all those who have contributed to the successful realization of this project. I would like to express my appreciation to the developers and contributors of OpenCV and YOLOv3, whose frameworks formed the cornerstone of our vehicle detection, classification, and counting system.

References

[1] Tech Vidvan. (n.d.). OpenCV Vehicle Detection, Classification & Counting. Tech Vidvan.(URL)

[2] Mali, P., Pallavi, L., Kondakalla, B., Ch, M. B., YCA, P. R., & Kondaveeti, B. (2023, April). Vehicle Detection, Classification and Counting. In 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN) (pp. 631-636). IEEE.

[3] Fachrie, M. (2020). A simple vehicle counting system using deep learning with YOLOv3 model. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(3), 462-468.

[4] Rashmi, C. R., & Shantala, C. P. (2020, November). Vehicle density analysis and classification using YOLOv3 for smart cities. In *2020 4th international conference on electronics, communication and aerospace technology (ICECA)* (pp. 980-986). IEEE.

[5] Bose, S., Ramesh, C. D., & Kolekar, M. H. (2022, August). Vehicle Classification and Counting for Traffic Video Monitoring Using YOLO-v3. In *2022 International Conference on Connected Systems & Intelligence (CSI)* (pp. 1-8). IEEE.