OBJECT DETECTION USING YOLO

Sanjay Swaran Kumar Rohith Parsi N.Ankitha M.Ranga Yochana

CSE AI&ML CSE AI&ML CSE AI&ML CSE AI&ML  
 SR UNIVERSITY SR UNIVERSITY SRUNIVERSITY SR UNIVERSITY  
WARANGAL,INDIA WARANGAL,INDIA WARANGAL,INDIA WARANGAL,INDIA

**Abstract-**

This study investigates automatic vehicle counting in video footage using the You Only Look Once (YOLO) object recognition approach. The paper includes algorithms created with YOLO along with TensorFlow API and OpenCV for vehicle identification, tracking, and counting. By comparing the automated counting method's efficiency with hand counts, an accuracy of almost 90% is achieved. Missing is a common problem. Future research directions are suggested, including further techniques to reduce missing and improve accuracy.

**Keywords -** Automated Vehicle Counting ,YOLO ,[TensorFlow](https://www.mdpi.com/search?q=TensorFlow), [OpenCV](https://www.mdpi.com/search?q=OpenCV), prerecorded video.

**Introduction -**

Counting vehicles might sound like a simple task, but it's crucial for managing traffic effectively. Think about it: knowing how many vehicles are on the road helps us plan better roads, manage congestion, and improve safety. Traditionally, we've used things like sensors embedded in the road or cameras to count vehicles. These methods work, but they're expensive and sometimes unreliable, especially when the weather is bad or traffic is heavy.

Now, imagine if we could count vehicles automatically, just by analyzing videos of the traffic. That's where computer algorithms come in. These algorithms use fancy math and clever tricks to look at videos and figure out how many cars are in them. It's like having a super-smart traffic monitor that works 24/7 without needing expensive sensors all over the place.

There are different types of algorithms for counting vehicles. Some use simple techniques like subtracting the background from the video to find moving objects, while others use more complex methods like deep learning, which is kind of like teaching a computer to think like a human brain. One popular algorithm in this field is called YOLO, which stands for "You Only Look Once." YOLO is like the superhero of car-counting algorithms because it's really fast and accurate.

In our research, we want to see how well YOLO works for counting cars in videos. We'll be using it to analyze videos of traffic and see if it can count cars as accurately as a human can. We'll also compare it to other methods to see if it's better or worse.

One cool thing about YOLO is that it doesn't just count cars – it can also tell what type of car it is. So, not only will we be able to count how many cars are on the road, but we'll also know if they're trucks, vans, or even motorcycles. This can be really helpful for planning roads and intersections that are safer and more efficient for different types of vehicles.

Our research will involve developing computer programs that use YOLO to count vehicles in videos. We'll then test these programs on a bunch of different videos to see how well they perform. We'll pay close attention to things like whether the program misses any vehicle or counts them incorrectly. After all, we want our traffic monitor to be as accurate as possible!

But it's not just about accuracy – we also need our program to be fast. Imagine if it took hours to count the vehicle in just one video! That's why we'll be testing how quickly our program can analyze the videos. The faster it can count vehicles, the more useful it'll be for real-time traffic monitoring.

Once we've tested our program and made sure it's accurate and fast, we'll compare it to other methods of counting cars. We'll look at things like cost – after all, it doesn't matter how good our program is if it costs a fortune to use. We'll also look at how easy it is to set up and use our program compared to other methods.

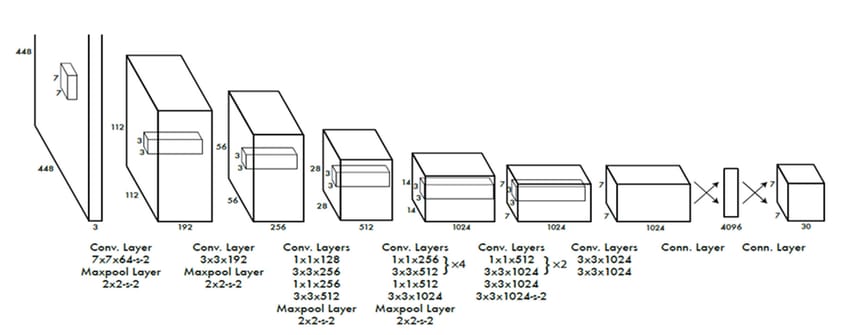
Overall, our research aims to make counting vehicles easier, cheaper, and more accurate. By using YOLO and other advanced techniques, we hope to develop a tool that can help cities and transportation agencies manage traffic more effectively. Whether it's reducing congestion, improving safety, or planning better roads, accurate car counting is the first step towards building a smarter, more efficient transportation system.

## **Materials and Methods**

#### **Collection of Video Data**

We gathered video data from our college premises, focusing on capturing the movement of vehicles such as buses, bikes, and cars exiting the campus. This footage served as the basis for our research. Later on, we utilized a method developed by our team to automatically count the traffic and assess its accuracy. By processing this video data, we aimed to streamline the process of traffic counting and ensure the reliability of our results.

**Working of YOLO**



Our system divides the input image into N \* N grid. If the centroid of object falls into a grid cell, that grid cell is responsible for detecting that object.

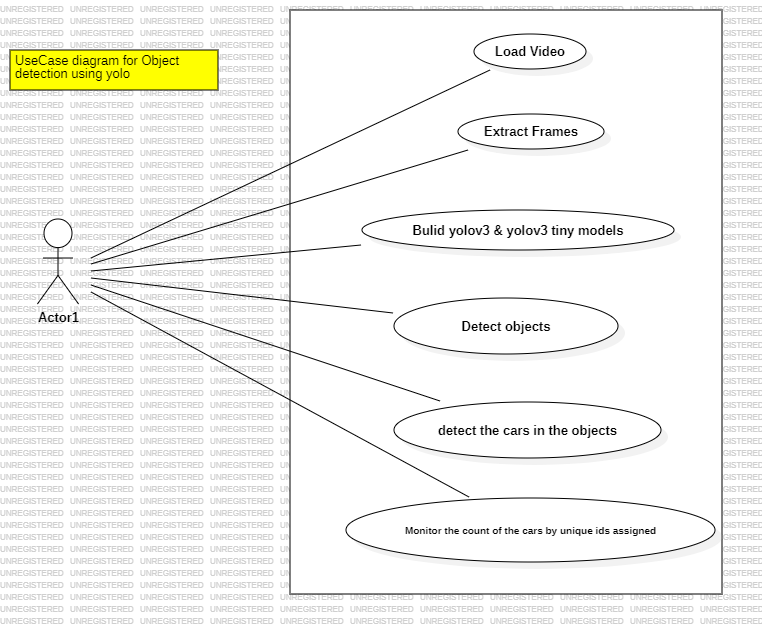
Each grid cell predicts B bounding boxes and confidence scores of those boxes. The confidence score shows how confident the model is that the box contains an object and how accurate it thinks the predicted box is. In case of no object within that cell, the confidence score must be zero.

To use YOLO for vehicle counting, the algorithms were developed to draw a reference line when the video frame appears. The bounding boxes for each vehicle in the frame is processed by YOLO. When the centroid of these bounding boxes meets the reference line, the count of vehicle increases.

**Selection of YOLO Version**

We went with YOLO version 8 for our project because it's fast, accurate, and easy to use. Even though newer versions like YOLOv4, v5, v6, and v7 are out, we found that YOLOv8 suits our needs best. YOLOv7 may be the most accurate, but it's overkill for our task since we don't need to detect tiny objects. Our method focuses on counting vehicles, not pinpointing small details.

Our choice of YOLOv8 doesn't affect our results compared to other methods either. The newer version of a method by Dey et al. claims 90% accuracy, but our model consistently hits 90% too. Other techniques, like a background subtraction model by Yang and Qu, or a vision-based method by Redmill, have slightly lower accuracy and slower speeds than YOLOv3. While traditional methods like piezoelectric sensors or pneumatic road tubes are more accurate, they're also expensive, so they're not really competition for our approach.



#### **Conversion of YOLO Weight File to TensorFlow API**

To make the YOLO model compatible with Python, we had to convert its weight file from its original format, which was developed using C/C++, to TensorFlow architecture. This conversion process required several packages: TensorFlow 18.0, which is a Numpy, a Python package that supports various mathematical functions; OpenCV, an open-source computer vision library TQDM, a progress bar library useful for nested loops and of course, Python itself.

We obtained the YOLO weight file from the official YOLO website, pjreddie.com, as it's an open-source resource.

Working with Python offers convenience and flexibility, especially for projects like ours that involve deep learning and computer vision. Python's readability and extensive libraries make it a preferred choice for many developers. So, we opted for Python to streamline our project and make the conversion process smoo

#### **Settings for Input Files**

First, we set up the YOLO weight file with its necessary parameters. Then, we created a way to handle input video files, making sure they're in the right format. After processing the videos, we uploaded them to our program. Finally, we checked everything was in order before moving on to process the videos.

#### **Detection of Vehicles**

We created algorithms to mark reference lines and spot vehicles in each video frame. Users can set a reference line by clicking on the screen when the first frame appears. These lines help the program count vehicles accurately later on. We used the YOLO weight file for vehicle detection.

#### **Settings for Output File**

So, we need a way to count vehicles over specific time periods and directions, but YOLO (an object detection algorithm) doesn't always process frames at the same speed as real time. To work around this, we figured out how long it takes YOLO to process each frame and used that to estimate how long it would take to process all the frames in the time period we need. This helped us adjust for YOLO's speed and accurately count vehicles entering and exiting in different directions.

#### **Algorithm for Counting Vehicle**

If a vehicle crosses a specific line first, it's counted from that side. The counts are displayed on the screen and saved in a spreadsheet.

**Apply YOLO with OpenCV**

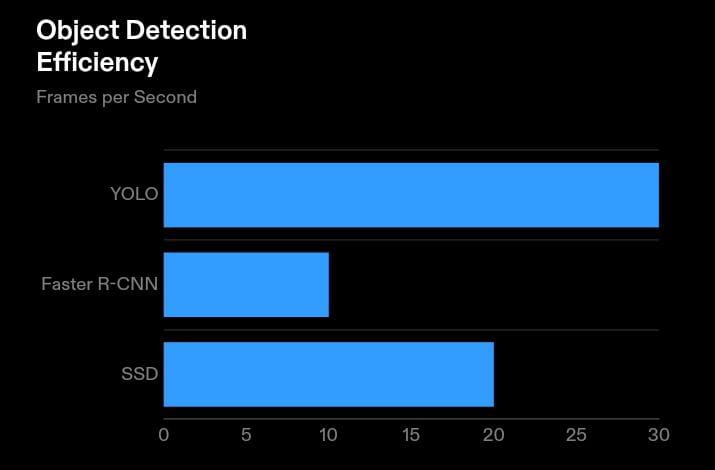
To make YOLO run faster when processing videos, it needs to work with OpenCV on a GPU. To set this up, you'll need to get cuDNN and CUDA, which you can download from NVIDIA's website. It's important to install CUDA correctly because YOLO relies on it to work with OpenCV and cuDNN. After installation, you can check if everything's working by running a test video file. If it's set up properly, the program will show graphics details at the bottom.

**Real-World Efficiency:**

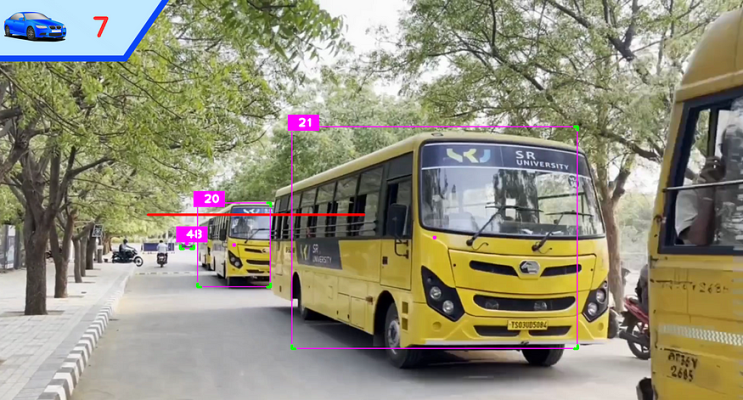
**YOLO** stands out as the fastest, processing 30 frames every second.

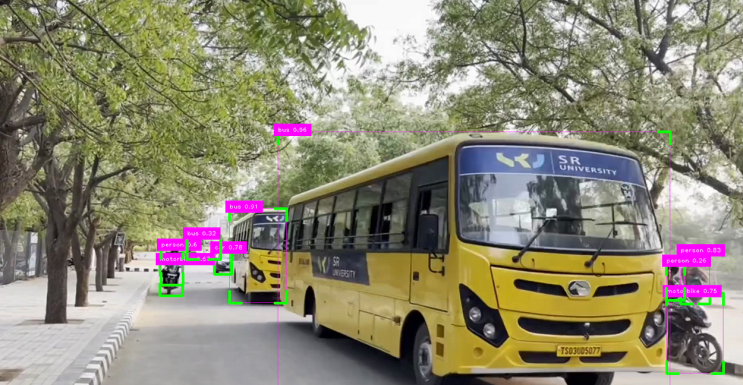
**Faster R-CNN** lags behind YOLO, only managing 10 frames per second.

**SSD** falls in between YOLO and Faster R-CNN, handling 20 frames per second.



## **Results:**







**CONCLUSION:**

In short, object detection using YOLO offers real-time processing and high accuracy, but faces challenges with precise localization, small objects, and resource-intensive training. Despite limitations, YOLO is widely used in applications like surveillance and autonomous driving, with ongoing efforts to improve its performance and versatility.

The project described, identification and tracking of vehicles within the college campus using YOLO technology, has several potential avenues for future expansion and enhancement.

The future scope for the project involves continuous innovation and expansion to address evolving challenges and opportunities in campus transportation management and smart city development. By embracing emerging technologies and collaborating with stakeholders, the project can contribute to creating safer, more efficient, and sustainable campus environments.

##### **References**

[1] Kusimo, K. O., & Okafor, F. O. (2016). Comparative analysis of mechanical and manual modes of traffic survey for traffic load determination. *Nigerian Journal of Technology*, *35*(2), 226-233.

[2] Leduc, G. (2008). Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*, *1*(55), 1-55.

[3] Leduc, G. (2008). Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*, *1*(55), 1-55.

[4] Achmad, A. (2015, May). Gaussian Mixture Models optimization for counting the numbers of vehicle by adjusting the Region of Interest under heavy traffic condition. In *2015 International Seminar on Intelligent Technology and Its Applications (ISITIA)* (pp. 245-250). IEEE.

[5] Chauhan, M. S., Singh, A., Khemka, M., Prateek, A., & Sen, R. (2019, January). Embedded CNN based vehicle classification and counting in non-laned road traffic. In *Proceedings of the tenth international conference on information and communication technologies and development* (pp. 1-11).

[6] Tan, L., Huangfu, T., Wu, L., & Chen, W. (2021). Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification.

[7] Yang, H., & Qu, S. (2018). Real‐time vehicle detection and counting in complex traffic scenes using background subtraction model with low‐rank decomposition. *IET Intelligent Transport Systems*, *12*(1), 75-85.

[8] Hjelm, S., & Gustafsson, M. (2018). Vehicle Counting Using Video Metadata. *LU-CS-EX 2018-13*.

[9]Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7464-7475).

[10] Kundu, R. (2023). YOLO: algorithm for object detection explained [+ Examples]. *V7, Feb*, *3*.