KolKGP Convergence

Pritish Saha
Nabayan Saha
Rounak Nath
Srinjoy Ganguly
Upal Mazumder

Report on Vehicle Detection, Tracking, and Forecasting using YOLOv10, Kalman Filters, and ARIMA

Objective:

The objective of this project is to aid in solving Bengaluru's traffic congestion problem by analyzing vehicle movements captured by camera feeds. Specifically, the code is designed to detect and track different vehicle types across multiple road sections using YOLOv10 for object detection and a Kalman filter for tracking. The system calculates vehicle counts and movement patterns between road segments, providing insights into traffic flow and vehicle transitions. Additionally, the code implements ARIMA modeling to predict short-term future vehicle counts based on historical data, enabling effective traffic management and prediction of turning patterns at key road junctions, which aligns with the goals of the hackathon.

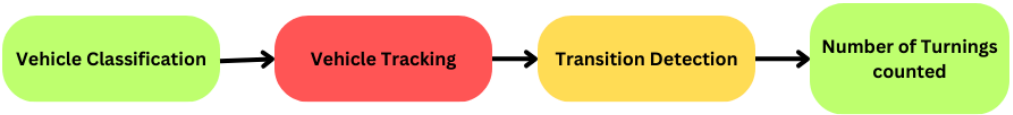
Approach:

Vehicle Classification: The first step involved detecting and classifying various types of vehicles, such as buses, cars, two-wheelers, and bicycles, using the YOLOv10 object detection model fine-tuned for this task.

Tracking Vehicle Transitions: By tracking the positions of the detected vehicles across frames, the Kalman filter was applied to smooth out the trajectories. The filter predicted and corrected vehicle positions while identifying transitions between different road segments.

Building the Dataset: Based on the detected vehicle transitions, data was gathered over 1.5-minute intervals, creating a detailed dataset that captures vehicle counts and movement patterns for each area segment.

ARIMA for Predictive Modeling: The historical dataset, collected over the 1.5-minute intervals, was fed into the ARIMA model to predict future vehicle counts. This enabled short-term forecasting of traffic, which can be used for real-time traffic management and decision-making at specific road junctions.



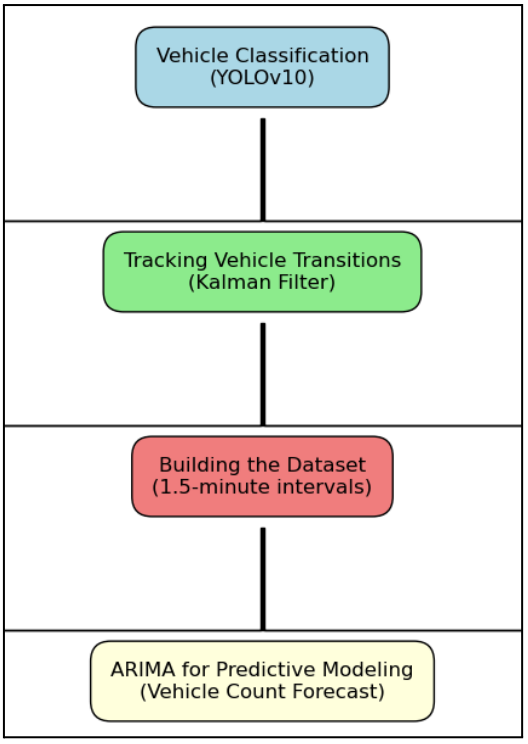


Fig1. The Video processing pipeline

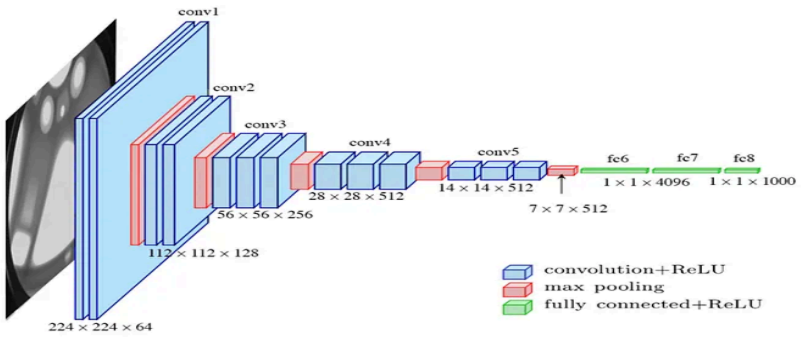
Vehicle Classification:

To adapt YOLOv10 for our vehicle classification task, we fine-tuned the model originally trained on the COCO dataset to recognize 7 custom vehicle classes. This process involved preparing a dataset with images annotated for the 7 vehicle types, updating YOLOv10's configuration to reflect these classes, and adjusting training parameters such as learning rate and epochs. By using pre-trained weights, we leveraged learned features from the COCO dataset, enabling the model to specialize in our specific vehicle classes. The fine-tuning process ensured that YOLOv10 effectively adapted to our dataset, enhancing its accuracy in vehicle detection and classification.

We used Roboflow to annotate different frames from the given dataset and used some data preprocessing steps to increase the dataset size (such as image augmentation). We used 912 images to finetune Yolov10 weights.

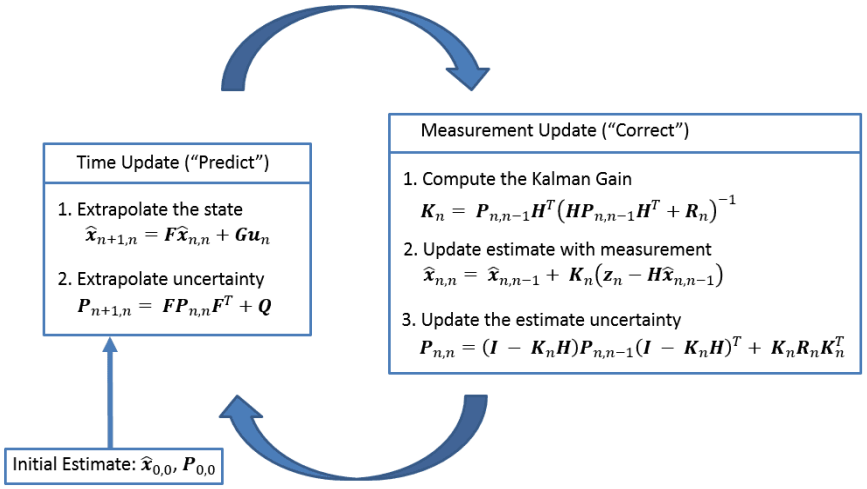
Yolov10:

YOLOv10 is an advanced version of the YOLO (You Only Look Once) object detection model, designed to improve both accuracy and speed compared to its predecessors. It achieves enhanced performance through refined network architectures and advanced feature extraction techniques, allowing for real-time object detection with high precision. YOLOv10 supports various datasets and tasks, making it versatile for applications like autonomous driving and surveillance. By leveraging transfer learning, YOLOv10 can be fine-tuned for specific detection needs, ensuring it remains effective across different scenarios and object types.



Why Kalman Filter?

Here the Kalman filter proved to be an excellent choice for tracking vehicle transitions due to its ability to effectively reduce measurement noise and provide accurate predictions of vehicle positions and velocities. By filtering out erratic movements and adapting to changes in vehicle speed and direction, the Kalman filter ensured reliable and continuous tracking of vehicles across different road segments. This capability was crucial for analyzing vehicle transitions accurately and maintaining high tracking performance in real-time video streams.



ARIMA:

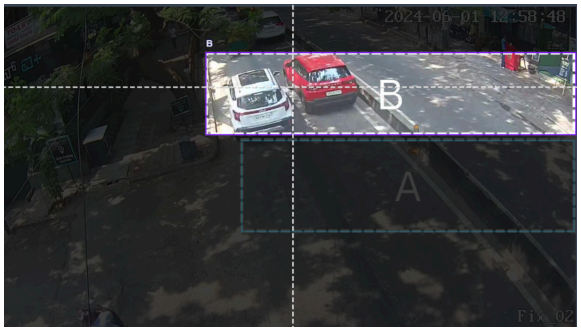
ARIMA is a time series forecasting method that models the relationship between past observations and future predictions. For your project, ARIMA is applied to predict vehicle counts at 1.5-minute intervals. The model works by analyzing historical data on vehicle counts, identifying patterns and trends, and using this information to forecast future counts. By integrating past observations (auto-regressive component), accounting for any trends (integrated component), and smoothing out random fluctuations (moving average component), ARIMA provides accurate predictions that help in understanding traffic patterns and managing vehicle flow effectively. This forecasting ability is crucial for making informed decisions based on predicted vehicle volumes in different road segments.

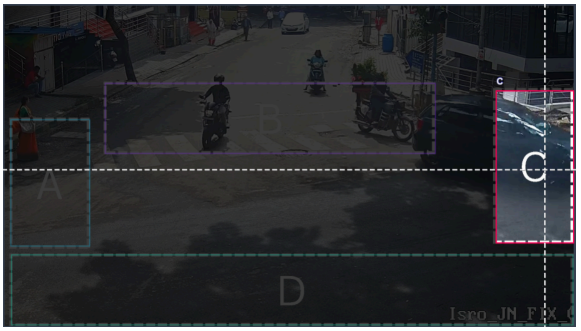
$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_0 Y_0 + \epsilon_t$$

$$Y_{t-1} = \beta_1 Y_{t-2} + \beta_2 Y_{t-3} + .. + \beta_0 Y_0 + \epsilon_{t-1}$$

Classifying Road area:

For seen and unseen camera views the data was given for the turning areas as A,B,C,D,E,F,G which needed to be taken into account by our model, for which we simply annotated the area for each region for each camera which essentially looks like follow.





Region Identification:

In the current approach, using centroids for vehicle identification in different regions is preferred over Intersection over Union (IoU) due to the challenge presented by vehicles of varying sizes. Each vehicle, based on its dimensions, might result in a different IoU threshold value. This variability can lead to inconsistencies in vehicle counting, as some vehicles may be overcounted while others are undercounted. The centroid-based approach simplifies this by focusing on the central point of each vehicle's detected position, providing a more consistent and reliable method for tracking and classifying vehicles across different regions.

Detection of Turning Patterns:

To detect the actual possible turning patterns we made a CSV containing area name and their possible turning patterns and used that information to optimize the identification of the turning patterns findings. There were problems related to large vehicles due to which they could be classified into multiple sections for example even though a BA turning pattern doesn’t exist the classifier will give some buses for BA.

Conclusion:

In this project, we developed a comprehensive system to address Bengaluru's traffic congestion by leveraging advanced technologies for vehicle detection, tracking, and forecasting. By integrating YOLOv10 for precise vehicle classification and the Kalman filter for reliable tracking, we effectively captured and analyzed vehicle movements across multiple road segments. Our approach, which involved fine-tuning YOLOv10 for specific vehicle types and utilizing the Kalman filter to smooth trajectories, ensured accurate tracking and insightful analysis of traffic flow and vehicle transitions. Additionally, ARIMA modeling provided valuable short-term forecasts of vehicle counts, facilitating proactive traffic management and better decision-making at critical road junctions. This integrated solution, enhanced by careful dataset preparation and region-specific annotations, offers a robust framework for managing and improving traffic conditions, aligning with the goals of the hackathon and contributing to more efficient urban transportation management.

Appendix:

YOLOv10-N: Architectural Overview and Key Features

1. Innovative Architecture:

- **Backbone:** Utilizes an enhanced version of CSPNet (Cross Stage Partial Network) to improve gradient flow and minimize computational redundancy. This enhancement leads to more efficient feature extraction.
- **Neck:** Incorporates PAN (Path Aggregation Network) layers for effective multiscale feature fusion, which aggregates features from different scales and helps in retaining fine details of objects.
- **One-to-Many and One-to-One Heads:** During training, multiple predictions per object (one-to-many) are generated to provide rich supervisory signals, improving learning accuracy. During inference, a single best prediction per object (one-to-one) is used, eliminating the need for Non-Maximum Suppression (NMS) and reducing latency.

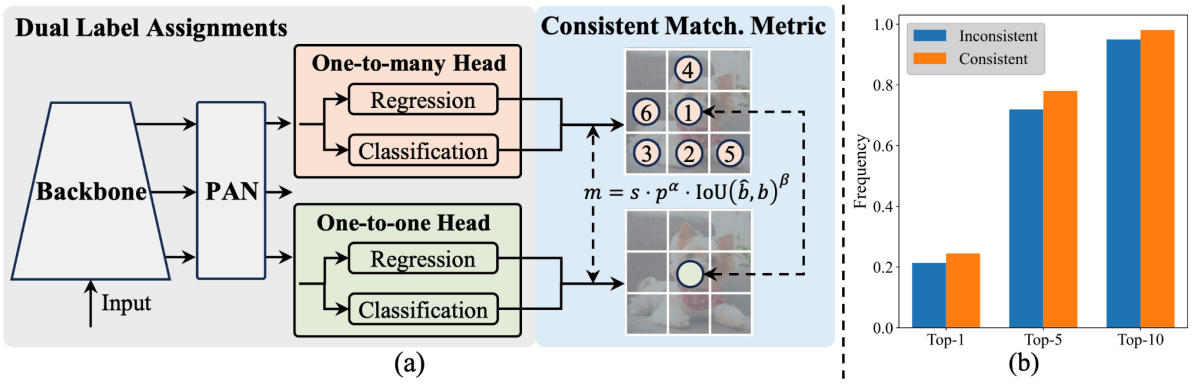


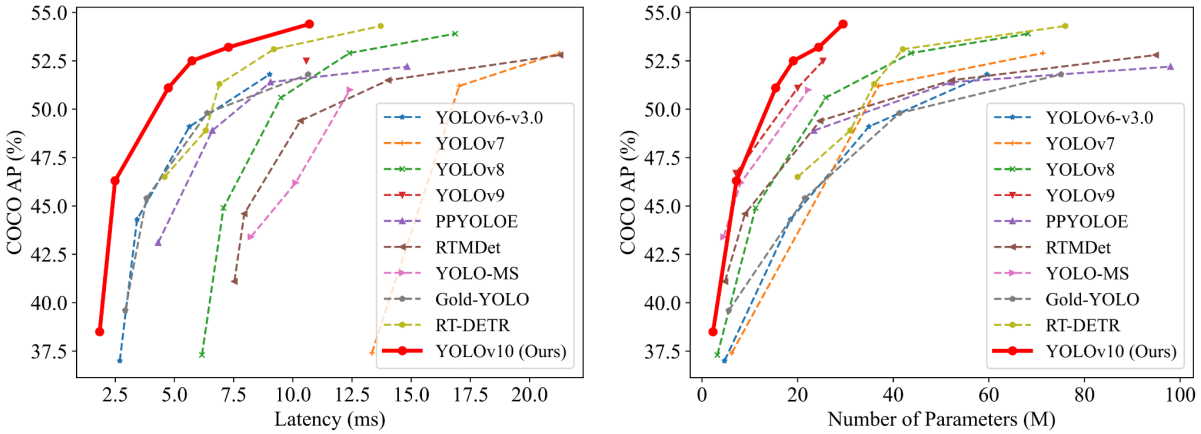
Figure 2: (a) Consistent dual assignments for NMS-free training. (b) Frequency of one-to-one assignments in Top-1/5/10 of one-to-many results for YOLOv8-S which employs α_{o2m}=0.5 and β_{o2m}=6 by default [20]. For consistency, α_{o2o}=0.5; β_{o2o}=6. For inconsistency, α_{o2o}=0.5; β_{o2o} =2.

2. Key Features:

- **NMS-Free Training:** YOLOv10 introduces consistent dual assignments, which replace the traditional NMS. This approach not only reduces inference latency but also ensures high-quality predictions.
- **Efficiency-Accuracy Optimization:** YOLOv10 is designed with a holistic approach that optimizes both efficiency and accuracy. Innovations such as large-kernel convolutions and partial self-attention modules enhance performance without significantly increasing computational cost.
- **Model Scalability:** YOLOv10 comes in various scales (e.g., YOLOv10-N, S, M, L, X) to cater to different computational requirements and application

needs. The YOLOv10-N variant is particularly optimized for resource-constrained environments, making it suitable for deployment on devices with limited computational power.

Comparison with Previous YOLO Versions and Other State-of-the-Art Models



Performance Benchmarks:

- **Accuracy vs. Latency:** YOLOv10 consistently outperforms previous YOLO versions and other state-of-the-art object detectors. For instance, YOLOv10-N achieves a mean Average Precision (mAP) of 39.5 at a latency of just 1.84 ms, demonstrating a superior accuracy-latency trade-off.
- **Model Efficiency:** Compared to models like YOLOv8-N and YOLOv6-3.0-N, YOLOv10-N has fewer parameters (2.3M) and lower FLOPs (6.7G), indicating a more efficient design that doesn't compromise on detection accuracy.

Conclusion

YOLOv10-N's novel architectural improvements and its NMS-free training strategy have significantly enhanced real-time object detection capabilities. By comparing its performance metrics with other state-of-the-art models, it's evident that YOLOv10-N provides a more efficient solution for object detection tasks, making it an ideal choice for applications requiring high accuracy and low latency.