World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Real-time traffic object detection using detectron 2 with faster R-CNN

Abiamamela Obi-Obuoha [1, *], Victor Samuel Rizama [2], Ifeanyichukwu Okafor [2], Haggai Edore Ovwenkekpere [3], Kehinde Obe [4] and Jeremiah Ekundayo [5]

[1] *AI Facilitator, National Centre for Artificial Intelligence and Robotics, National Information Technology Development Agency, Abuja, Nigeria.*
[2] *Intern, National Centre for Artificial Intelligence and Robotics, National Information Technology Development Agency, Abuja, Nigeria.*
[3] *Department of Electrical and Electronics Engineering, Petroleum Training Institute, Effurun, Delta, Nigeria.*
[4] *Department of Computer Science with Mathematics, Obafemi Awolowo University, Osun, Nigeria.*
[5] *Department of Library and Information Science, Ahmadu Bello University, Zaria, Kaduna, Nigeria.*

World Journal of Advanced Research and Reviews, 2024, 24(02), 2173–2189

Publication history: Received on 14 October 2024; revised on 21 November 2024; accepted on 23 November 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.24.2.3559

Abstract

Object detection is becoming more and more important in daily life, especially in applications like advanced traffic analysis, intelligent driver assistance systems, and driverless cars. The accurate identification of objects from real-time video is crucial for effective traffic analysis. These systems play a vital role in providing drivers and authorities a comprehensive understanding of the road and surrounding environment. Modern algorithms and neural network-based architecture with extremely high detection accuracy, like Faster R-CNN are crucial to achieving this. This study investigates an advanced object detection system designed for urban traffic applications using an interactive Gradio interface and Detectron2's Faster R-CNN model. The research focuses on developing a model capable of identifying key traffic objects such as traffic lights, vehicles, buses, crossroads etc., with high accuracy and precision. A significant contribution of this study is the integration of Gradio-based interface that enables users to upload images or videos from their local storage or webcam and view the results in real time making the model both accessible and practical. Our findings demonstrate that the Detectron2 framework, paired with Gradio's interactive interface offers a reliable and scalable solution for traffic monitoring and safety applications.

Keywords: Detectron2; Faster R-CNN; Gradio; Intelligent Traffic System (ITS); Object Detection; Real-time Detection**.**

1. Introduction

The need for intelligent traffic management and real-time monitoring systems has increased due to the rapid growth of metropolitan areas. Many intelligent transportation systems (ITS), such as autonomous vehicles, advanced driver assistance systems (ADAS), and intelligent traffic management systems rely on vision-based traffic detection as a fundamental component. Conventional methods for direct picture identification frequently rely on motion analysis and manually created characteristics. However, recent improvements in deep convolutional neural networks (DCNNs) has transformed object detection tasks, yielding amazing performance in vehicle detection as well [1].

Despite the success of CNNs in detection, achieving real-time performance in driving environments still poses significant challenges which arise from factors such as partially visible vehicles, fragmented views, and significant variations in vehicle size within traffic images. Faster R-CNN [2] and SSD [3], two popular object detection models based on CNN, have not shown the best performance for detection tasks in their normal configurations.

* Corresponding author: Abiamamela Obi-Obuoha

These approaches often involve adapting the base network to handle varying scales using multiscale feature maps or leveraging input images with multiple resolutions [4]. Real time traffic detection is still a problem because these techniques still need a lot of processing power even if they have improved detection accuracy on numerous public test datasets when compared to conventional object detection models based on solely CNN [3].

This research presents an improved framework based on Faster R-CNN that is intended for real-time traffic object detection in order to address the aforementioned issues. It makes use of Detectron2's Faster R-CNN model, which is trained on a unique traffic dataset and has a ResNet-101 backbone. It also incorporates Gradio, an open source tool for developing machine learning model user interfaces that provides a real-time interface that is easy to use for interaction with the model. Gradio bridges the gap between advanced machine learning models and useful real-world applications by enabling the uploading of visual data and the retrieval of annotated output (s)

2. Literature Review

2.1. Machine Learning

It is a branch of artificial intelligence that allows computer systems to learn from their experiences and get better over time without the need for explicit programming. It makes use of algorithms that can identify patterns in data, frequently using input features to predict or categorize results. Computer vision has significantly advanced as a result of machine learning, particularly deep learning which allows systems to automatically learn intricate features from a variety of datasets. Because of its capacity to recognize spatial and hierarchical structures in images, Convolutional Neural Networks (CNNs), a branch of deep learning, has emerged as essential for image processing applications, such as object detection [5]. For object detection tasks, traditional methods used manually designed features like Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG). The intricacy of hand-crafted features, which frequently failed in dynamic situations particularly with fluctuating object scales and orientations typical of traffic scenes, was a limitation of these techniques [6, 7].

CNN architectural advancements like AlexNet [8], VDDNet [9], and ResNet [10] have greatly enhanced models' capacity to extract detailed representations from data. In order to solve the problem of vanishing gradients in deep networks and enable deeper designs that capture complex properties, ResNet in particular incorporated residual connections. These developments have been vital in enabling sophisticated applications that require great accuracy and fine trained information, such as traffic detection. When labeled data is scarce, transfer learning – the process of fine-tuning pre-trained models on huge datasets, such as Image Net, on task specific datasets further improves model performance [11].

2.2. Object Detection

As a key element of computer vision, object detection is a deep learning technique that includes identifying and locating items in an image. Sliding windows, which were computationally costly and prone to high false-positive rates, were used by traditional object detection algorithms like the Viola-Jones detector, to detect objects [12]. By employing CNNs to produce high quality region proposals, contemporary object detection frameworks like Region based Convolutional Neural Networks (R-CNN), Fast R-CNN, and Faster R-CNN have completely transformed the area. While Fast R-CNN increased efficiency by sharing computations across the full image, Faster R-CNNS introduced the idea of isolating region proposals and using CNNs to classify each region [13, 14].

By integrating a Region Proposal Network (RPN) that predicts object bounding boxes directly, Faster R-CNN improved speed and accuracy by doing away with the need for external region proposal algorithms [15]. Other real-time frameworks that do single pass detections, like Single Shot Multibox Detector (SSD) and You Only Look Once (YOLO), have been created to increase processing performance. YOLO, for example, processes the entire image in a single neural network pass, achieving real-time performance, albeit with some trade-offs in localization accuracy, particularly for small objects and crowded scenes [16, 17]. Faster R-CNN's balance between precision and processing speed makes it an ideal choice for high stake applications, such as traffic monitoring, where accuracy is prioritized over real-time performance.

2.3. Faster R-CNN

One popular two-stage detector that is well known for its excellent performance on a variety of datasets is Faster R-CNN, as shown in Figure 1. Regression branch classification, bounding box, Region Proposal Network (RPN) [18], and a backbone are its main constituents. Features are extracted from input images by the backbone network and sent to the RPN, which is made up of two subsets: one for object regression and the other for object classification. Classification

probabilities are predicted by the classification subset, and object locations are estimated by the regression subset utilizing anchor boxes as reference points.

Non-maximum suppression is used to eliminate proposals with large overlap and to choose the best proposals with the highest probabilities based on the classification. Then thresholds are established to ascertain whether or not an object is included in a proposal. Region of Interest (ROI) pooling or ROI aligning is used to standardize the form of these chosen proposals in conjunction with the backbone network's retrieved data [19]. The suggestions are then sent to two subsets: the regression subset, which estimates the proposals' locations, and the classification subset, which forecasts the proposals' categories.

The total loss is computed as shown in Equation 1.

$$L_{total} = L_{rpn} + L_{rcnn} \quad (1)$$

Where:

L_{rpn} and L_{rcnn} represent the class loss and the regression loss respectively.

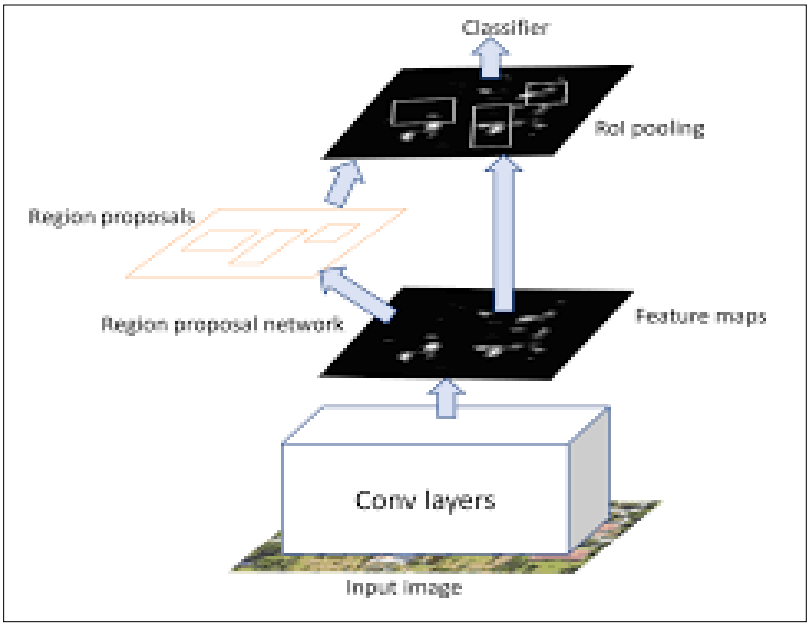

Figure 1 The Structure of a Faster R-CNN

2.4. ResNet-101 Backbone

Faster R-CNN is one of the most popular object detection frameworks because it strikes a balance between processing efficiency and accuracy, which makes it perfect for applications that need accurate object localization. In order to propose and categorize regions inside an image simultaneously, the design combines a Region Proposal Network (RPN) with a Fast R-CNN detection network [15]. Even in complicated images, the model can capture tiny features because of the ResNet-101 backbone, a deep convolution network with 101 layers.

Compared to conventional CNNs, ResNet-101 is more effective for deep architectures because it uses residual connections to alleviate the vanishing gradient issue. Research indicates that ResNet-101 and Faster R-CNN together offer better identification accuracy, particularly for jobs with many classes and different object scales [19]. Numerous applications of this combination have been made in domains such as autonomous driving and video surveillance, where accurate, multi-class detection is essential. By using this backbone, the current study benefits from enhanced detection precision, which is critical for identifying and localizing multiple objects in urban traffic scenes.

2.5. Detectron2 Framework

Detectron2 as depicted in Figure 2 developed by Facebook AI Research (FAIR) group, is a state of art, modular framework designed to support diverse object detection and segmentation tasks. Developed using Pytorch, Detectron2

is highly flexible, enabling easy adaptation to custom datasets and support for advanced architectures, including Faster R-CNN, Mask R-CNN, and Retina Net [20]. Detctron2's effectiveness and strong model training capabilities have led to its widespread use, making it a popular choice for academic research and industrial applications. The framework includes built in utilities for dataset handling, evaluation metrics, and visualization, streamlining the training pipeline. In addition to Faster R-CNN, Detectron2 supports feature pyramid networks (FPN) and high capacity backbones like ResNet, enabling it to handle multi scale detection challenges effectively. Its application extends across domains, including medical imaging, autonomous driving and surveillance, demonstrating its adaptability in complex detection tasks.

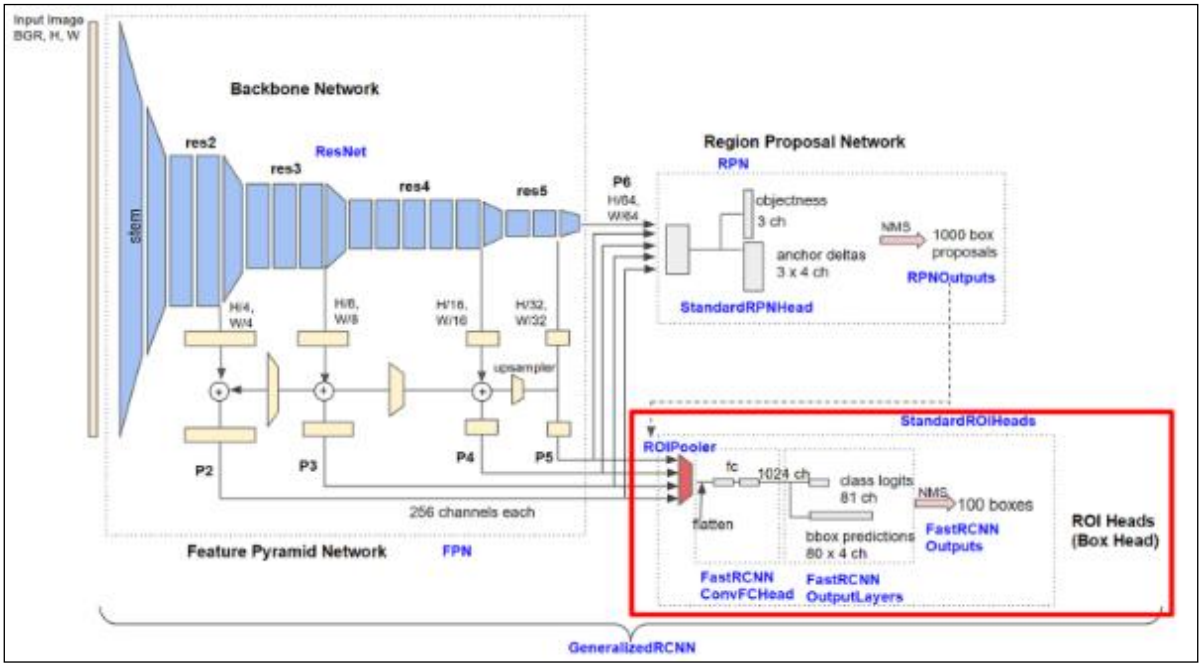

Figure 2 Detectron2's Architecture

2.6. Gradio

Gradio is an open source python library that makes it easier to build web-based interfaces for machine learning models and at the same time enables real-time user interaction [21]. By providing a web-based platform for uploading images or videos and viewing model outputs, Gradio enhances model accessibility making it useful for both demonstrations and practical applications. Non-technical users can explore model functions thanks to Gradio's interactive nature, which is especially useful for domains like ITS that gain from public involvement.

Gradio has proved to be useful in fields where real-time feedback is essential, such as medical diagnosis, language processing, and image categorization. Gradio's integration in this work improves model usability for traffic monitoring and safety applications by enabling users to upload custom data of traffic scenes and observe the annotated detections in real-time. The use of Gradio in this study is an attempt to bridge the gap between accessible real-world applications and technical model implementations.

2.7. Related Works

Numerous techniques have been developed to handle the peculiar difficulties posed by complicated traffic scenes and urban traffic monitoring. Object detection has been the subject of much of this research. In Intelligent Transportation Systems (ITS), where real-time monitoring of cars, pedestrians, traffic lights, and other road elements is crucial for applications like autonomous driving, urban safety, and traffic management, traffic object detection is crucial. A number of models have been created to detect objects in urban environments.

[22] explored a 3D multi-view object detection network designed specifically for self-driving cars, achieving excellent detection accuracy for both cars and pedestrians. This method utilized a combination of LiDAR sensor and visual camera data, enabling effective object detection in 3D space. While this study successfully demonstrated the potential of multi-modal object detection, it relied on specialized hardware limiting its applicability to systems with LiDAR.

[23] applied faster R-CNN in an urban traffic context, emphasizing high accuracy in detecting vehicles and pedestrians. The study used a feature extraction approach through a ResNet backbone, enabling precise detection across different object types. However, the focus was not optimizing accuracy and couldn't address model accessibility or real-time user interaction, which are essential for practical implementation in public facing applications. The model's dependence on high performance hardware also limited its scalability to real world resource constrained environments.

In a related study, [16] introduced YOLO, a real time detection framework capable of high-speed object detection. While YOLO achieved significant speed improvements by processing entire images in a single pass, it faced challenges with localization accuracy, particularly for smaller objects in dense scenes. Given the limitations in detecting objects at varying scales and with high precision, YOLO is often less suited for scenarios requiring fine-grained accuracy such as urban traffic monitoring, where distinguishing small or distant objects (e.g. traffic lights) is critical.

Detecting small and distant objects, such as traffic lights or pedestrians far from the camera remains a challenging task in object detection. To address this, [24] investigated a multi-scale feature fusion technique that used multiple CNN layers to record varying levels of information, hence increasing the detection of small objects in urban environments. Multi-scale methods, however frequently resulted in higher computing complexity, which could be detrimental for real-time applications that demand minimal latency.

In addition to challenges with small objects, variance in environmental conditions (e.g. low-light scenarios, fogs or rain) poses challenges for object detection models. [18] also put forward an enhancement to this by training models with augmented data, simulating diverse weather conditions to improve robustness. Although data augmentation has showed effectiveness in making models more adaptable to environmental changes, it did not fully address the performance trade-offs, particularly in real-time systems where resource constraints limit extensive data augmentation.

While there have been significant advancements in model accuracy and speed, few studies integrate components that make models accessible to non-technical users. Traditional object detection systems are often designed for deployment on high-performance servers or within tightly controlled autonomous vehicle systems, where only technical personnel can test or evaluate the model's performance. This limited accessibility restricts the model's utility for public demonstrations, educational purposes, or direct testing by stakeholders in ITS planning.

In traffic object detection research, interactive machine learning interfaces like those made possible by Gradio are still relatively new and unexplored. By providing a platform that allows users to upload images or videos and retrieve annotated outputs, Gradio enables a more user-friendly experience that can facilitate testing, feedback and adoption. Studies such as [21] highlight Gradio's effectiveness in making complex models interpretable and usable, but its application in real-time object detection, particularly for ITS has been limited.

This study focuses on implementation of a high-accuracy object detection model that can monitor traffic in real-time. The system utilizes a computer vision model to accurately identify and localize multiple objects like vehicles, traffic lights, fire hydrants, pedestrians, and crossroads in complex dynamic environments. It goes further to incorporate an intuitive user interface so as to improve accessibility and user interaction.

3. Material and methods

The system as shown in Figure 3 starts by capturing visual information of the environment using specialized image sensor cameras which would be mounted at specific areas of the traffic scene. A thorough grasp of the surroundings is then produced by combining the data collected by various cameras. The data is then sent to the trained Faster R-CNN model where it is processed. If the system correctly identifies and localizes the objects detected, it proceeds to send this data to the control centre. However, the model reprocesses the data for better outcomes if it is unable to accurately recognize and localize objects.

The components of the system are divided in to hardware and software components. The hardware components include; image sensor cameras mounted at specific areas of the traffic scene and the processor device (GPU), while the software components include; the trained Faster R-CNN model and the control interface (Gradio).

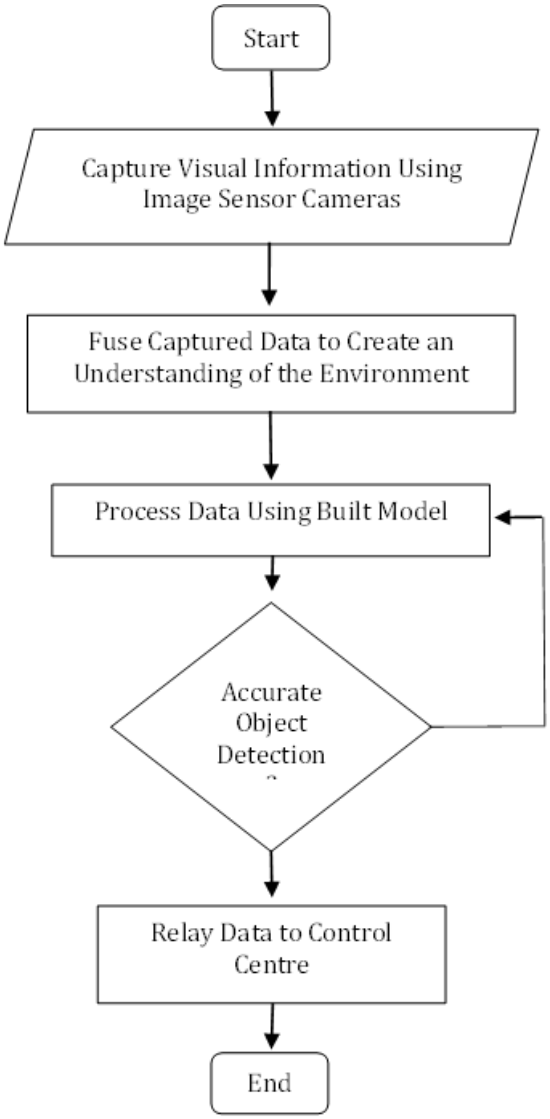

Figure 3 The Flowchart of the System

3.1. Model Overview

The most important part of the system is the trained model, which is responsible for accurately recognizing and localizing objects in real-time. Using mounted cameras to capture traffic scene data at specific vehicle locations is the first stage in real-time object detection and localization. This input data is then pre-processed to ensure the right format of the data is provided into the model.

After the pre-processing stage, the model receives the data input ad uses it to extract high-level properties including object shapes, edges, textures and spatial connections. After analysing the feature map, the Convolutional Neural Network (CNN) model looks for potential regions of interest (ROI) that could hold the desired objects.

After that, the Region Proposal Network (RPN) surrounds those regions with bounding boxes, giving preference to those that are most likely to include the objects of interest. This output is then provided as an input to the area which uses the Convolutional Neural Network (CNN) to produce fixed size feature maps that correspond to each area proposal. For bounding box regression and classification, the fully connected layers process the derived fixed size feature maps.

The model's classification determines the object class, such as car, traffic signals, crossroads, fire hydrant, and motorcycle, while the bounding box regression layers refine the coordinates of the proposed bounding boxes to more nearly coincide with the ground truth box placements. After identifying the objects, the system relays the data to the

control centre to plan safe routes, avoid collisions, adapt to traffic circumstances etc. Figure 4 is a detailed representation of the model's object detection process.

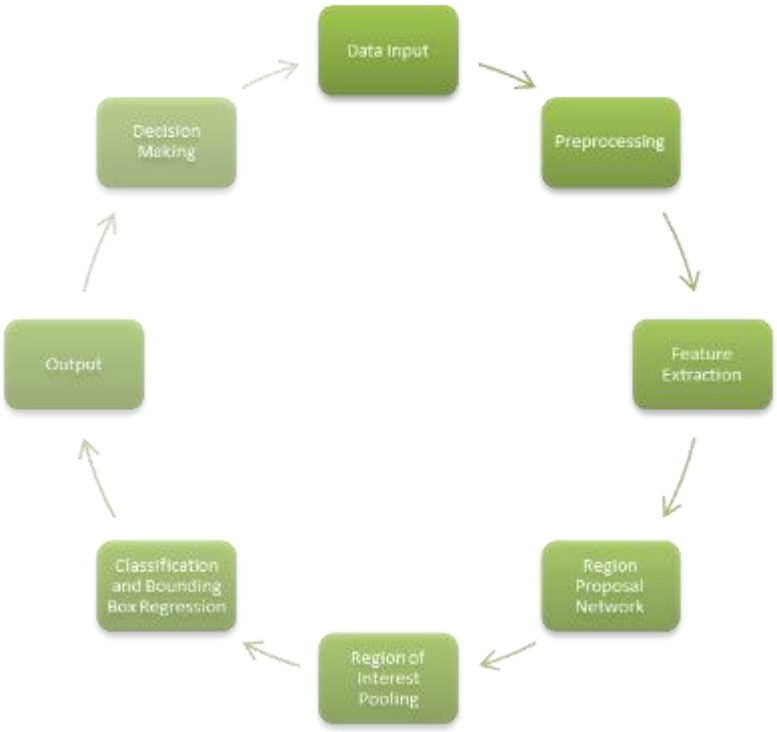

Figure 4 The Model's Object Detection Process

3.2. Image Data Acquisition

Roboflow, a computer vision online data source provided a total of 814 photos that included different classifications of traffic objects, including bicycles, cars, motorbikes, crossroads, fire hydrants, and traffic lights. Common Objects in Context (COCO) JSON format, which is fully compatible with the Detectron2 Faster R-CNN architecture, contained the entire dataset. A Sample image from the dataset is shown in Figure 5.

Figure 5 Sample Images of the Dataset

3.3. Data Preparation and Annotation

The dataset used was prepared according to the COCO format, a widely adopted standard in object detection. The process included annotation, registration in Detectron2 and data visualization.

The COCO JSON format is a widely used approach for annotating object detection and localization datasets. Studies have demonstrated that COCO style of annotations improves model robustness by supporting multi-class detection and handling occlusions effectively, which are frequent challenges in traffic scenes [25]. This involved the process of adding pertinent information or metadata to raw data so that the machine learning algorithm could understand and use it. COCO style annotations involved bounding boxes and class labels for every object in an image. This dataset included classes relevant to traffic scenes such as vehicles, cross walks, buses, motorcycles, bicycles, fire hydrants and traffic lights. Each object instance was annotated with specific coordinates such as *x, y, width* and *height* and assigned a category facilitating supervised training on multi-class data.

The annotations which included bounding boxes and class labels were essential for training the model to recognize and localize objects effectively. The COCO dataset served as a standard for the object detection task due to its rich annotations that encompassed a variety of object classes, complex scenes, and object overlap [18]. Annotation quality was critical for model accuracy, as poor-quality annotations could lead to inaccuracies in object localization and classification.

3.4. Data Cleaning and Structuring

For optimization, the data cleaning and structuring procedure was crucial. It established the groundwork for the machine learning task to be successful. Five object types were eliminated from the COCO JSON format dataset in order to mitigate the noise and irregularities. Due to the underrepresentation of these classes, they were eliminated, rendering them wholly unnecessary for the training procedure. Adopting this strategy will undoubtedly improve model performance and avoid potential problems during data processing.

3.5. Data Pre-processing

This involved resizing each image so that its shortest edge is 800 pixels, with a maximum dimension of 1333 pixels. This resizing step ensured uniform image dimensions, which is vital for maintaining model accuracy. Inconsistent image sizes could interfere with feature extraction and bounding box localization, so standardizing the size helped to optimize performance.

3.6. Environmental Setup and Required Libraries

The environmental setup was critical to ensure efficient processing, particularly with GPU support. As a foundational library for deep learning and neural networks, Pytorch provided tensor operations and GPU support. Pytorch version 1.12.1 was utilized which was compatible with CUDA version 11.3 for GPU acceleration. Matplotlib was also used to create static, animated and interactive visualizations of the COCO dataset and monitor the training process while providing insights to the model's performance.

In addition, to handle image and video processing during object detection, Open Source Computer Vision Library (OpenCV), a comprehensive library for computer vision tasks was used to read images from the dataset registered on Detectron2 as it provided pre-trained model configurations and utilities, supporting the Faster R-CNN architecture with a ResNET-101 backbone.

Lastly, Gradio was installed to enable a web-based user interface that would enable users to interact with the model in real time. Gradio makes machine learning deployment easier by offering input/output components that are simple to configure and essential for easily accessible model demos.

3.7. Train Test Split

The dataset was divided into train, test and validation sets in a ratio of 61:16:23 prior to training. This preserved the robustness of the model while ensuring sufficient data for the test and validation sets.

3.8. Model Configuration and Training

Using pre-trained weights from the COCO dataset, a faster R-CNN architecture with a ResNet-101 backbone was used to enhance generalization and speed up convergence.

To enhance data loading speed, the number of CPU threads for data loading was set to 4. By increasing the number of worker threads, the model could load data in parallel, thereby minimizing idle GPU time and optimizing the training process. This configuration was particularly beneficial when handling large datasets, as it reduced Input/output bottlenecks.

ResNet, a deep Convolutional Neural Network (DCNN) renowned for its potent feature extraction capabilities served as the foundation for the Faster R-CNN model architecture. By utilizing transfer learning, weights that had previously been trained on the COCO detection dataset were loaded. As opposed to training from scratch, using the pre-trained weights accelerated the convergence of the model which significantly improved accuracy.

The batch size (i.e. number of images processed per iteration) was set to 4. This choice balanced memory constraint with gradient estimation accuracy, providing stable training updates.

In order to properly handle the trade-off between slow training and instability, the initial learning rate was set to 0.001. Also, in order to improve the model's performance and reduce the possibility of over fitting, a warm up phase was also included to stabilize training during the early stages. During this phase, the learning rate progressively increased over the first 1000 iterations before gradually decreasing between 1000 and 1400 iterations. Also, the reduction factor γ = 0.05 applied at each step further assisted in smoothing out training and achieving a better convergence.

In addition, the number of proposal samples per image was set to 64 in order to improve the model's object detection learning and to balance between the two classes while increasing object detection accuracy. This aided in regulating the quantity of positive and negative data collected during Region of Interest (ROI) pooling.

Finally, periodic evaluation was also done at every 500 iterations which were essential for monitoring model performance and adjusting parameters as necessary. This helped track the loss curve and other performance metrics, making it possible to intervene if over fitting or under fitting was detected. The hyper parameters used during training can be seen in Table 1.

Table 1 The Model's Hyper parameters used for Training

SN.	Parameter	Value
1	Data Loader Workers	4
2	Batch Size	4
3	Initial Learning Rate	0.001
4	Warm-up Iterations	1000
5	Maximum Iterations	1500
6	Learning Rate Step Schedule	1000, 1400
7	Learning Rate Decay Factor	0.05
8	ROI Batch Size per Image	64
9	Evaluation Period	500

4. Results and discussion

4.1. Metrics for Model Evaluation

The model's performance on the test dataset was assessed following training. By using COCO metrics for thorough performance insights, object detection accuracy was evaluated using Detectron2's default trainer. As indicated in Table 2, these metrics provided a thorough evaluation across various Intersection over Union (IoU) thresholds and object sizes.

4.2. Intersection over Union (IOU)

This is the basic computation that determines whether detection is a true positive. It is the overlap between the ground truth box (Bgt) and the actual box measured. Its mathematical definition can be seen in Equation 2.

$$\text{IOU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{2}$$

Where:

$|B_p \cap B_{gt}|$ is region where the ground truth boxes and the actual boxes overlap.

$|B_p \cup B_{gt}|$ is the combined area of the two boxes.

Precision (P)

The ratio of true positive detections to the total detections. It gauges how well the model recognizes the right objects. It is mathematically defined as seen in Equation 3.

$$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)+False Positives (FP)}} \tag{3}$$

Recall (R)

The proportion of true positive detections to the total number of real objects. It shows that the model can locate all pertinent objects. It is mathematically defined as seen in Equation 4.

$$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)+False Negative (FN)}} \tag{4}$$

Average Precision (AP)

The Average Precision (AP) is calculated as the mean of the highest precisions at various recall levels and represents the area under the precision-recall (PR) curve. AP is computed in COCO evaluation across a number of IOU thresholds, usually with a step size of 0.05 and a range of 0.5 to 0.95. The overall AP of 0.442 (44.2%) indicates a balanced degree of object detection accuracy. It is mathematically defined as seen in Equation 5:

$$AP = \int_0^1 P(r)dr \tag{5}$$

Where:

P(r) represents the precision as a function of the recall.

Mean Average Precision (mAP)

The mAP is the average of the AP values calculated over multiple IoU thresholds and across all object classes.

It is mathematically defined as seen in Equation 6.

$$mAP = \frac{1}{T}\sum_{t=1}^{T} \text{AP}_{\text{IoU=t}}$$

(6)

Where:

T is the number of IOU thresholds.

Average Precision (AP) at 50% and 75%

The AP values at IOU thresholds of 0.5 and 0.75 (i.e. 50% and 75% respectively) highlights the model's localization performance. The higher AP (@50) – 76.9% suggests good localization at relaxed thresholds but the lower AP (@75) – 43.9% suggests stricter bounding box which indicates that higher thresholds will result in lower values hence posing a challenge.

4.3. Average Precision by Object Size

The AP for small objects (18.20%) is significantly lower than for medium (35.80%) and large objects (53.10%). This disparity is typical in object detection models, as small objects often pose greater challenges due to their limited pixel representation, making it harder for the model to accurately identify and localize them. In contrast, larger objects provide more visual information, which facilitates better detection performance.

4.4. Average Recall (AR)

It is the average of the recall calculated at various IOU criteria. AR provides an overall sensitivity assessment by taking into account all true positives across various thresholds. Like AP, AR in COCO is averaged over multiple IOU thresholds, yielding a final AR score. The Recall of 57.90% shows reasonable object coverage, but indicates that further improvements in recall could enhance detection completeness.

Mathematically, it is expressed in Equation 7.

$$AR = \frac{1}{T}\sum_{t=1}^{T} \mathrm{R}_{\mathrm{IoU=t}} \tag{7}$$

Where:

$\mathrm{R}_{\mathrm{IoU=t}}$ is the recall at each IOU threshold.

Table 2 Model Evaluation Metrics

SN.	Metric	Value (%)
1	AP@50-95	44.2
2	AP@50	76.9
3	AP@75	43.9
4	APs (Small Objects)	18.2
5	APm (Medium Objects)	35.8
6	APl (Large Objects)	53.1
7	AR@50-95	57.9

These metrics provided a multi-faceted evaluation of the model, capturing not only its accuracy in detecting objects but also its localization precision and scale sensitivity. This comprehensive approach provides a robust framework for assessing and comparing model performance across diverse detection challenges.

4.5. Training Loss Analysis

4.5.1. Total Loss

This is the total loss across all tasks. Including losses from Region Proposal Networks (RPNs), box regression, and classification. As the model learns from the data, it becomes increasingly accurate at predicting object classes and bounding boxes as observed by a consistent drop in total loss.

4.5.2. Classification Loss

This is the difference between the actual and anticipated class labels. As the classification loss trend decreases, it indicates that the model is improving its capacity to distinguish between different kinds of objects in the traffic environment by learning to identify classes accurately over time.

4.5.3. Box Regression Loss

This loss which measures the difference between the ground truth and anticipated bounding boxes is essential for accurate item localization and detection. The stability of this loss indicates the model is successfully enhancing its capacity to precisely place bounding boxes around identified objects.

4.5.4. Region Proposal Network (RPN) Classification Loss

The RPN suggests potential object regions within an image. A low and stable RPN classification loss indicates that the model efficiently identifies potential object locations, which aids in reducing false positives and helps focus on relevant regions.

4.5.5. Region Proposal Network (RPN) Localization Loss

This is the degree to which the suggested regions resemble the locations of the ground truth objects. In this case, for the model to be considered stable, the regions that are recommended must be in good alignment with the actual locations of the objects to improve the accuracy of the object detection.

In summary, a consistent decline in losses suggests that the model is effectively learning from the training set. Low RPN localization and classification implies that the model has stabilized successfully; hence reducing mistakes and identifying patterns for object detection. A graphical breakdown of all losses is shown in Figure 6.

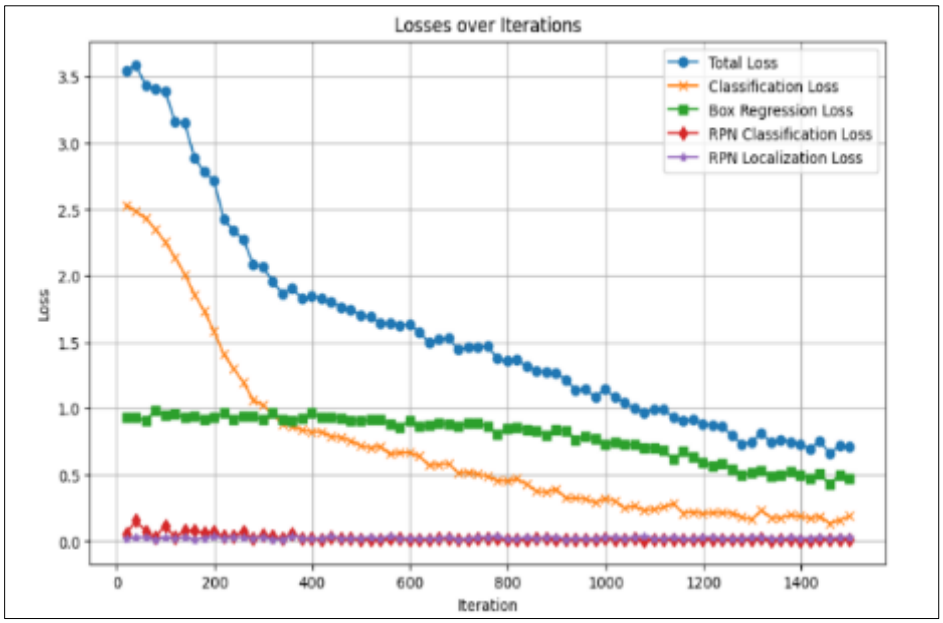

Figure 6 Graphical Representations of the Losses

4.6. Data Analysis

Table 3 is a distribution of the test dataset across eight (8) distinct categories in COCO format was which featured a variety of object classes pertinent to road and traffic environments. The distribution reveals that some classes, such a Vehicles and Traffic lights were mostly represented, with 209 and 129 instances respectively, while others, like Road-Traffic, had no instances at all. The absence of Road-Traffic instances indicated that no images were available for this class in the dataset. Furthermore, underrepresented classes like Buses and Bicycles might have experienced lower detection accuracy because of the limited number of training examples. This class imbalance suggests that expanding the dataset could enhance the model performance and robustness across all categories.

Table 3 Distribution of Dataset across Different Categories

SN.	Category	Instances	Percentage (%)
1	Road-Traffic	0	0.00
2	Cross Walks	81	16.01
3	Traffic Lights	129	25.49
4	Bicycles	22	04.35
5	Fire Hydrants	23	04.55
6	Vehicles	209	41.30
7	Buses	12	02.37
8	Motorcycles	30	05.93
TOTAL		506	100

4.6.1. Inference Process

The model's inference process involved evaluating 133 batches with each batch corresponding to a single test image.

4.6.2. Data Loading Time

The time it took to load the data was 0.0016 seconds which indicated efficient dataset loading, allowing for smooth data throughput.

4.6.3. Inference Time

An inference time of 0.189 seconds per iteration represented the model's computation time for each image. This was critical as it reflected the model's efficiency in real-time settings.

4.6.4. Evaluation Time

The evaluation time of 0.0003 seconds per iteration reflected the time taken to evaluate predictions after inference.

4.6.5. Per-Class Category Precision Analysis

Figure 7 provides the per-class category Average Precision (AP) values, allowing for deeper performance across different object categories.

The model showed strong performance on high performing classes like Fire Hydrants (59.30%), Buses (54.80%), and Vehicles (49.90%). These high scores indicate that the model effectively learned to identify these objects, likely because they contained more distinct and easily recognizable features. In addition, these classes might have been better represented in the train set, providing the model a larger variety of examples to learn from, which also contributed to their high detection accuracy.

Alternatively, low performing classes like crossroads showed a much lower AP of 12.90% reflecting the challenges the model faced in accurately identifying these objects. Crossroads often present complex and highly variable visual patters, which can vary greatly based on the environment. The lack of sufficient examples and clear visual cues in the dataset might have likely contributed to the model's difficulty in detecting them with high precision.

Meanwhile, medium performing classes like Traffic Lights, Motorcycles, and Bicycles achieved moderate AP values, suggesting that while the model performed reasonably well in detecting these objects, there was still room for improvement. These objects present a wider range of visual appearances or environmental context, and the model could have benefited from additional training data or more refined detection techniques to improve performance across these categories.

The variation in per-class AP values indicated that some object categories were easier to detect due to higher representation or distinctive visual features while other might have required additional data for better performance.

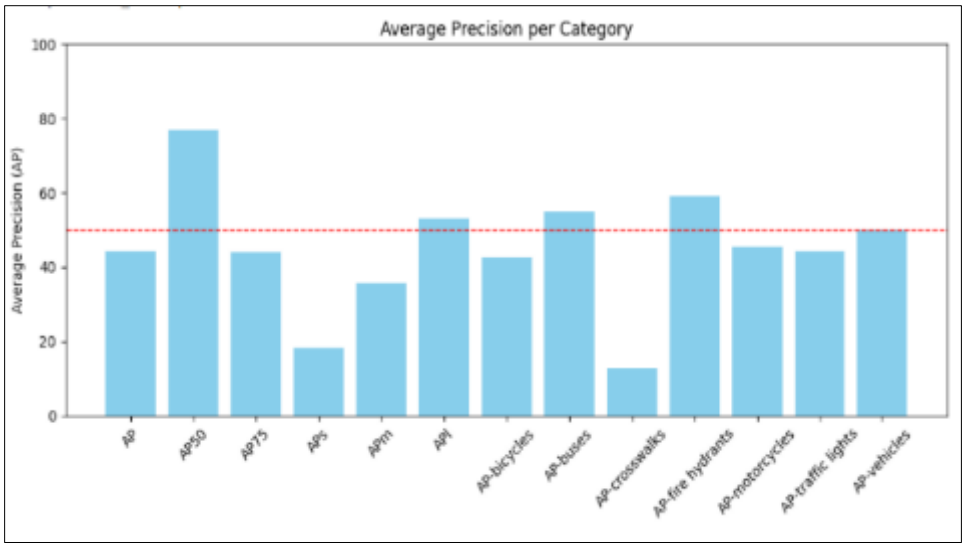

Figure 7 Average Precision by Class Category

4.7. Gradio Interface Implementation

Gradio enables the implementation of a user-friendly interface for real-time object detection. It enables deploying the trained model to a web interface, allowing users to interact with the model either via an image or video input. To set up the model for inference, the configuration is initialized and the model weights are loaded. The model's number of classes is set to 12, and a list of class names relevant to the dataset is defined for easy identification. The default predictor instance then processes images and video frames through the trained model. Using the following functions, individual images and video frames are handled for real-time object detection.

4.8. Predict and Display Frame Function

This function processes each frame by converting it to Red-Green-Blue (RGB) format, runs inference with the predictor and draws instance predictions (i.e. bounding boxes, labels) on the frame using Detectron2's visualize. The processed frame is then returned in RGB format and to the Gradio interface for display.

4.8.1. Live Tracking Function

Here each frame from a video source is processed iteratively. By utilizing OpenCV, the function captures frames from a specified video input.

4.8.2. Detect Objects inn Image Function

This converts an input image to the required format and processes it, returning an annotated image with detected objects.

4.8.3. Gradio Interface Design

Real-time object detection is made easier by the Gradio interface's distinct options for video and image data inputs. The application is made available through a browser after the local Gradio server has been launched. Users can submit an image or video input for object detection, or record video from their camera. By triggering the live tracking function, the start and detect buttons begin object detection on the video input. Similarly, they trigger the detect object in an image function to begin object detection on the image. The processed outputs show either the annotated images detected or the processed video with detected objects, depending on the scenario.

4.8.4. Visualization of Detected Outputs

Detection of output visualizations provided qualitative insights into the model's performance. Presented below are samples of output images processes through the trained model using the Gradio interface, which showcased its detection accuracy across a variety of traffic scenarios.

The model demonstrated its ability to accurately detect vehicles at varying distances, effectively distinguishing them even in the presence of occlusions and overlapping instances. It also successfully identified traffic lights with high confidence scores.

Figure 8 Visualization from Highly Dense Scenes

The bounding boxes were precisely aligned with vehicle contours, and confidence scores consistently exceeded 80%, reflecting the model's reliability and robustness in complex scenarios.

Furthermore, the model effectively detected the majority of vehicles and traffic lights across various pose and sizes in previously unseen images. The higher confidence scores for vehicle detection likely stemmed from their extensive representation in the training dataset, depicting the model's adaptability and strong generalization capabilities.

In addition, the model correctly identified the presence of vehicles in the uploaded data, such as the one shown in Figure 9 although the detection accuracy varied depending on factors like distance and illumination.

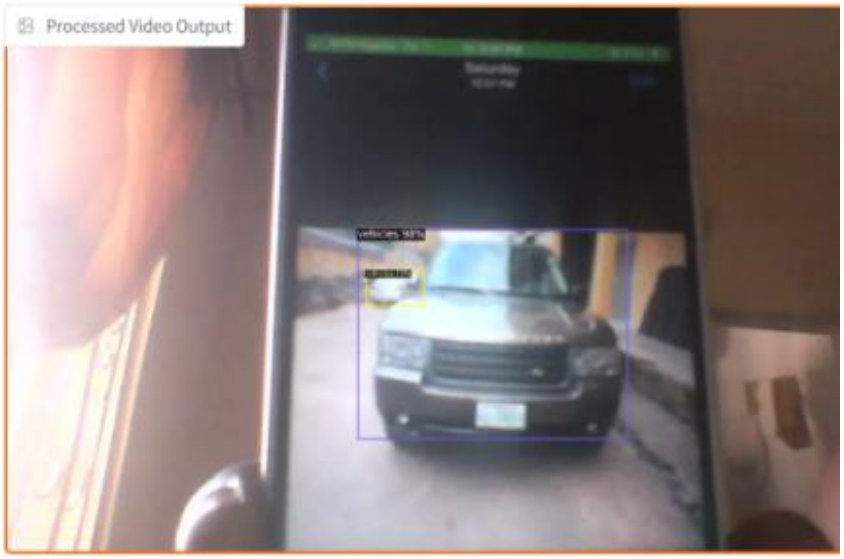

Figure 9 Model's Performance on Uploaded Data

4.9. Gradio Interface Interaction and Insights

The Gradio interface provided seamless experience for users to upload traffic scene images or videos, either from their local storage or via a webcam, with annotated results displayed almost instantly. Its straightforward design featuring an upload option and an output display made it highly accessible, even for users without programming experience. The interface enabled efficient and prompt inference, underscoring its practicality for real-world traffic monitoring.

This user-friendly design, combined with the model's impressive efficiency, highlights its potential for diverse applications in traffic management. Suggestions for enhancements, such as batch processing capabilities and improved integration for video streams, could further elevate its functionality and broaden its real-world application. The Gradio interface not only facilitates real-time testing but also serves as a clear demonstration of the model's utility.

5. Conclusion

This study successfully illustrated how to recognize traffic objects in real time using Dtectron2's Faster R-CNN model with a ReNet-101 backbone, improved by an intuitive Gradio interface. In various traffic situations, the model correctly identified several object types, such as cars, buses, bicycles, crossroads, motorcycles, fire hydrants and traffic signals. The strong performance in handling dynamic, multi object urban traffic environments was demonstrated by its excellent performance.

With the aid of the web based Gradio platform for real-time detection, the Gradio interface enabled users to examine annotated outputs or even provide visual data from webcams or local storage, increasing its usefulness for intelligent transportation systems (ITS) and urban planning stake holders.

The study also identified areas for improvement while highlighting the synergy between Gradio and Detectron2 for traffic monitoring. Improving inference speed for real-time edge deployment and optimizing identification for tiny objects such as far-off traffic lights were among the difficulties.

In order to increase adaptability, future research should concentrate on improving continuous video stream processing, use more data for training to improve generalization and accuracy, and add more traffic classes.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 7310-7311).

[2] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 2015 Dec(pp. 91-99)

[3] Chen X, Kundu K, Zhu Y, Berneshawi AG, Ma H, Fidler S, Urtasun R. 3d object proposals for accurate object class detection. Advances in neural information processing systems. 2015;28.

[4] Cai Z, Fan Q, Feris RS, Vasconcelos N. A unified multi-scale deep convolutional neural network for fast object detection. InComputer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14 2016 (pp. 354-370). Springer International Publishing.

[5] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

[6] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) 2005 Jun 20 (Vol. 1, pp. 886-893). IEEE.

[7] Lowe DG. Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision 1999 Sep 20 (Vol. 2, pp. 1150-1157). IEEE.

[8] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25.

[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[10] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[11] Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks?. Advances in neural information processing systems. 2014;27.

[12] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 2001 Dec 8 (Vol. 1, pp. I-I). IEEE.

[13] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 580-587).

[14] Girshick R. Fast r-cnn. arXiv preprint arXiv:1504.08083. 2015.

[15] Ren S. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497. 2015.

[16] Redmon J. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition 2016.

[17] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. Ssd: Single shot multibox detector. InComputer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 2016 (pp. 21-37). Springer International Publishing.

[18] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft coco: Common objects in context. InComputer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 2014 (pp. 740-755). Springer International Publishing.

[19] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. InProceedings of the IEEE international conference on computer vision 2017 (pp. 2961-2969).

[20] Wu Y, Kirillov A, Massa F, Lo W.-Y, Girshick, R. (2019). "Detectron2." Facebook AI Research. Retrieved from https://github.com/facebookresearch/detectron2

[21] Abid A, Abdalla A, Abid A, Khan D, Alfozan A, Zou J. Gradio: Hassle-free sharing and testing of ml models in the wild. arXiv preprint arXiv:1906.02569. 2019 Jun 6.

[22] Chen X, Ma H, Wan J, Li B, Xia T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition 2017 (pp. 1907-1915).

[23] Luo W, Yang B, Urtasun R, Ma H: Traffic object detection in urban scenes using Faster R-CNN. Journal of Autonomous Driving. 2020; 2(1): pp 15-16.

[24] Liang X, Zhang J, Zhuo L, Li Y, Tian Q. Small object detection in unmanned aerial vehicle images using feature fusion and scaling-based single shot detector with spatial context analysis. IEEE Transactions on Circuits and Systems for Video Technology. 2019 Mar 20;30(6):1758-70.

[25] Zhu X, Wu X, Wei Y, Zhang W: Improving Small Object Detection Through Multi-Scale Feature Fusion. 2017.