DOI: https://doi.org/10.52756/ijerr.2024.v38.005 *Int. J. Exp. Res. Rev.*, Vol. 38: 46-60(2024)

Original Article | Peer Reviewed | Open Access

International Journal of Experimental Research and Review (IJERR)
© Copyright by International Academic Publishing House (IAPH)
ISSN: 2455-4855 (Online)
www.iaph.in

Performance Evaluation of YOLOv5-based Custom Object Detection Model for Campus-Specific Scenario

Dontabhaktuni Jayakumar and Samineni Peddakrishna*

School of Electronics Engineering, VIT-AP University, Amaravati-522241, Andhra Pradesh, India

E-mail/Orcid Id:

DJ, jayajkd@gmail.com, https://orcid.org/0000-0003-3779-9904;
SP, krishna.samineni@gmail.com, https://orcid.org/0000-0002-5925-8124

Article History:

Received: 04th Jun., 2023
Accepted: 10th March, 2024
Published: 30th Apr., 2024

Keywords:
Autonomous electric vehicles, computer vision, custom data, object detection, YOLO

How to cite this Article:
Dontabhaktuni Jayakumar and Samineni Peddakrishna (2024). Performance Evaluation of YOLOv5-based Custom Object Detection Model for Campus-Specific Scenario. *International Journal of Experimental Research and Review*, *38*, 46-60.
DOI: https://doi.org/10.52756/ijerr.2024.v38.005

Abstract: This study evaluates the performance of a custom object detection model based on the YOLOv5 architecture, specifically tailored for autonomous electric vehicles. The model undergoes pre-processing using the Roboflow computer vision platform, which offers a wide range of tools for data pre-processing and model training. The experiments were conducted on a diverse dataset comprising various objects encountered in campus-specific driving scenarios, such as pedestrians, vehicles, buildings, and obstacles. The performance of the custom object detection model is assessed using standard metrics, including precision, recall, mean average precision (mAP), and intersection-over-union (IoU) at different thresholds. The training process was conducted in a controlled environment, resulting in a Precision of 0.851, a Recall of 0.831, and a mAP of 0.843. These metrics were analyzed to evaluate the YOLOv5-based custom object detection model's ability to detect and categorize objects accurately, its precision in predicting bounding boxes, and its capability to handle various object categories. We also examined the effects of different hyperparameters and data augmentation techniques on the model's performance, including variations in learning rate, batch size, and optimizer algorithms to determine their impact on accuracy and convergence. This analysis provided valuable insights into the model's strengths and weaknesses, highlighting areas for improvement and optimization. These findings are instrumental in developing and deploying advanced object detection systems to enhance the safety and reliability of autonomous electric vehicles.

Introduction

The autonomous vehicle technology industry has experienced significant advancements in recent years due to the increasing need for reliable and precise object detection systems. Object detection is a fundamental task in computer vision and has recently undergone significant developments. (Amjoud et al., 2023; Diwan et al., 2023; Srivastava and Tripathi, 2023). Recent advancements in deep learning and computer vision have greatly enhanced object detection performance, making it a vital technology widely utilized across various industries. There are different techniques for object detection, which can be categorized into two primary groups. The first group consists of algorithms based on classifications, while the second group relies on regression methods. Object localization is utilized within the first group to precisely locate objects within images, particularly in real-time systems requiring rapid detection of multiple objects. This approach involves the use of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to identify regions of interest in the image and categorize them (Dhruv et al., 2020). However, this method has limitations, especially in real-time systems where speed is paramount. The computational expense of running predictions for each selected region can slow down the object detection process, rendering it less suitable for real-time applications.

On the other hand, algorithms based on regression, such as You Only Look Once (YOLO), are widely used in object detection. YOLO treats object detection as a regression problem, predicting bounding boxes and class probabilities in a single forward pass through the neural network. This approach is more computationally efficient than classification-based methods, as it eliminates the need to run predictions for each region individually. Instead, it simultaneously detects objects in the entire image, making it ideal for real-time applications. Over time, the YOLO algorithm has gone through various iterations and improvements to enhance its performance in object detection tasks. These different versions of YOLO have led to advancements in accuracy, efficiency and model architecture.

YOLOv2, the second version of YOLO, introduced several key improvements compared to its predecessor. It incorporated the concept of anchor boxes, which allowed the model to predict objects of different sizes and aspect ratios more accurately. This improvement led to better localization and increased detection performance (Redmon et al., 2017; Banerjee et al., 2023). Expanding upon the advances of YOLOv2, YOLOv3 introduced enhancements in the field of object detection. It implemented a multi-scale prediction approach utilizing feature maps at varying resolutions. Additionally, YOLOv3 incorporated the "darknet-53" technique, which employs a more intricate neural network architecture for enhanced feature extraction (Redmon et al., 2018). These architectural upgrades resulted in improved accuracy in object detection.

With the introduction of YOLOv4, performance was further enhanced through the adoption of innovative techniques such as the CSPDarknet53 backbone architecture, PANet (Path Aggregation Network) neck, and the Mish activation function (Bochkovskiy et al., 2020). These advancements significantly improved the model's detection accuracy and made it more efficient for real-time applications. The latest version, YOLOv5, brought further improvements in object detection performance. It incorporated the EfficientDet architecture, which combined the benefits of EfficientNet and EfficientDet models (Wang et al., 2021). EfficientDet is known for its state-of-the-art performance in object detection tasks. By leveraging the EfficientDet architecture, YOLOv5 achieved better accuracy and generalization compared to its predecessors. In addition to architectural improvements, each version of YOLO has also introduced enhancements to the loss functions and training strategies. For instance, YOLOv4 introduced the complete intersection over union (CIoU) loss function, which specifically addressed the challenge of imbalanced datasets and further improved the model's performance.

Considering the advancements and success of the YOLO series, the decision to implement the YOLOv5 deep neural network has proven to be beneficial in our work. By utilizing this robust and dependable detection method, the research combines established techniques (YOLOv5) with a unique dataset and AEVs application in a campus environment to tackle a specific challenge within the autonomous vehicle industry. This study involves:

- Developing and categorising a new image dataset that focuses on objects typically encountered on a college campus. This data is more pertinent to our intended application than more general datasets.
- The utilization of the YOLOv5 model for AEV object detection within a campus environment is a unique and valuable contribution. Our research investigates the effects of different hyperparameters and data augmentation techniques on the model's performance to improve accuracy and model convergence.

The remainder of this paper will explore the relevant object detection research. Section 2, related literature will first explore classical machine learning techniques used and then provide a detailed examination of deep learning-based object detectors. Section 3, object detection methodology and evaluation, will then detail the proposed object detection approach and its evaluation strategy. Here, we will describe the specific pre-processing steps to prepare the data for training with the YOLOv5 model architecture. Following this, Section 4, results and discussion will present the experimental results and their evaluation using various metrics. We will then provide a thorough discussion of these results, explaining their significance and any potential limitations. Finally, the concluding section will summarize this study's key findings and contributions. Here, we will emphasize the effectiveness of the YOLOv5 model for object detection in a campus environment. We will also discuss potential avenues for future research to improve object detection capabilities further.

Related Literature

Before the advent of deep learning in 2013, object detection predominantly utilized classical machine learning techniques. Some common methods included the Viola-Jones object detection technique (Viol et al., 2001), scale-invariant feature transforms (SIFT) (Lowe et al., 2004), and histogram of oriented gradients (HOG) (Dalal et al., 2005). These methods relied on handcrafted features and traditional algorithms tailored to identify objects in images based on specific characteristics and patterns. Although these classical techniques were effective for certain applications and constrained scenarios, these classical techniques had limitations in addressing complex and varied object detection tasks. The Viola-Jones technique, designed for real-time face detection, was particularly successful in scenarios focused on detecting faces in images or video streams, especially in well-lit and controlled environments (Viola et al., 2001). On the other hand, SIFT was well-suited for scenarios necessitating matching key points between images finding applications in tasks like image alignment, panorama stitching, and object recognition (Yan Ke et al., 2004). HOG gained popularity for pedestrian detection, performing well in situations with simple backgrounds and upright pedestrians. The challenge with these methods was creating features that could accurately represent objects despite variations in appearance, scale, and orientation in real-world images.

However, the design of handcrafted features posed a challenge in classical object detection methods. Crafting features that could effectively represent objects across various appearances, scales, and orientations was difficult (Viola et al., 2001). Real-world images presented a wide range of variations, such as changes in lighting conditions, viewpoints, poses, and object sizes. Handcrafted features struggled to adapt to these variations, leading to the need for feature representations that could adjust to changing object appearances and environmental conditions (Girshick et al., 2014). This need led to the development of deep learning-based approaches, which have significantly advanced object detection by automatically learning and adapting feature representations from raw image data (LeCun et al., 2015). This advancement has revolutionized object detection, improving accuracy, robustness, and adaptability.

The introduction of deep learning, particularly the potential of Deep Convolutional Neural Networks (DCNN) for object detection, has significantly impacted the research community. This transformation followed the influential work of Krizhevsky et al. (2017), Alex Krizhevsky et al. (2012), Sermanet et al. (2013) and Pierre Sermanet et al. (2013) on the challenging ImageNet dataset. Krizhevsky et al. (2017) demonstrated the capabilities of DCNN for object localization and detection tasks, showcasing the effectiveness of identifying object locations within images. Sermanet et al. further expanded on this by illustrating how DCNNs could be used to locate and detect instances of objects, emphasizing the advantages of combining classification, localization, and detection tasks. The integration of these tasks within the DCNN framework led to improved performance across all tasks, highlighting the potential for a unified approach to object detection challenges. These contributions have underscored the effectiveness and versatility of DCNNs in handling complex detection tasks, leading to the widespread adoption of deep learning-based approaches in object detection (Krizhevsky et al., 2017; Sermanet et al., 2013). As a result of these developments, significant advancements have been made in creating highly accurate and efficient object detection systems for a variety of applications.

Building on this progress, Girshick et al. introduced a Region-based Convolutional Neural Network (R-CNN) (Girshick et al., 2015). They utilized the AlexNet architecture and achieved significant improvements in detection performance. However, R-CNN had limitations. It was computationally expensive and slow to train, taking days even for small datasets (Girshick et al., 2015). This led to further research and the development of faster, more efficient object detection models, such as Fast R-CNN (Girshick et al., 2015). Fast R-CNN addressed processing speed issues by sharing convolutional feature computations across region proposals, though challenges remained with region proposal generation speed.

To address this issue, the faster R-CNN model, introduced in the same year by improving the region proposal process with the Region Proposal Network (RPN) (Shaoqing Ren et al., 2015), (J. Long et al., 2014). Integrating RPN directly within the network eliminated the need for separate regional proposal methods, enhancing overall efficiency and streamlining the architecture. Another notable advancement came with the introduction of the Region-based Fully Convolutional Network (R-FCN) (Dai et al., 2016). This innovative approach shared computations within the network and employed position-sensitive score maps to address translation invariance. By combining Faster R-CNN and FCN, R-FCN achieved faster and more precise object detection, albeit with modest gains in accuracy. These developments significantly propelled the field of computer vision, facilitating the creation of quicker and more accurate detection systems. Most notably, Mask R-CNN extended the capabilities of Faster R-CNN with a parallel branch for pixel-level object instance segmentation (He et al., 2018). Subsequent to the advancements of Mask R-CNN, the domain of object detection continued to witness considerable progress with the introduction of the Single Shot MultiBox Detector (SSD) (Liu et al., 2016) and YOLO series (Joseph Redmon et al., 2016), which revolutionized real-time object detection capabilities. SSD combined the benefits of one-stage object detectors with multi-scale feature maps, allowing for real-time object detection on various objects across different scales. On the other hand, YOLO adopted a one-stage approach that allowed it to simultaneously predict object locations and class probabilities in a single pass, resulting in real-time object detection capabilities. More sophisticated models have emerged from this research advancements in object detection, such as YOLOv2, YOLOv3 and YOLOv4, each building upon the strengths of its predecessors to achieve even higher levels of accuracy and efficiency.

While previous studies using datasets like PASCAL VOC and COCO achieved promising results with methods like Faster R-CNN and YOLOv2, their accuracy ranged from 19.7% to 83.6%. This indicates a need for further advancements in object detection, particularly for custom datasets. The authors' addresses this gap by employing the latest YOLOv5 model on a custom dataset. This approach has the potential to surpass existing accuracy rates by achieving 84.3% and improve object detection capabilities for specific use cases not covered by the standard datasets.

The advancements in the YOLO series, particularly compared to two-stage detectors like Faster R-CNN and R-FCN, strongly motivated the use of YOLOv5 in the significant challenges for object detection in campus environment, especially considering the application in AEVs for robust object detection systems for enhanced safety and reliability. The main contributions of this research include assembling and labelling an image dataset, training YOLOv5 with a custom dataset for object detection, and implementing the YOLOv5s model to achieve high accuracy through test inference in real-world scenarios.

Object Detection Methodology and Evaluation

The implementation process begins by collecting a comprehensive dataset of 5246 images taken from various locations on campus. A block diagram illustrating the proposed implementation process is shown in Figure 1. The first step involves annotating each image by outlining regions containing different objects with bounding boxes and assigning accurate labels for subsequent stages. Following annotation, data augmentation is applied to create varied versions of the original images in the dataset.

The YOLOv5 model is chosen as the foundational architecture for data training due to its effectiveness and precision in object detection tasks. The training involves cycling through the augmented training dataset. The YOLOv5 model takes in an image at each cycle, generates predicted bounding boxes, and calculates class probabilities. The model's performance is assessed by comparing these predictions with the actual annotations and calculating the loss.

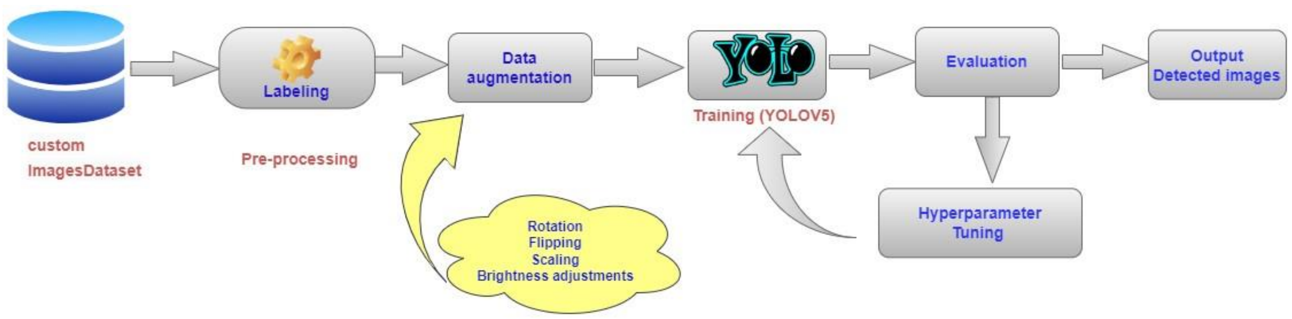

Figure 1. Block diagram of the proposed model.

The final trained model is tested on a separate test set upon completion of the training phase. The test set comprises images that the model has not previously encountered and is used to evaluate the model's performance on unfamiliar data. If the model's performance on the test set is unsatisfactory, adjustments can be made through fine-tuning. Fine-tuning involves modifying the model's parameters to enhance its performance on the specific task at hand. Finally, the deployed YOLOv5 model is utilized for real-time object detection in the campus environment.

Data Annotation

Effective data preparation is a crucial initial step in developing accurate deep-learning models. Image annotation is a key component of this process, involving precisely identifying and labelling objects within images. This typically entails creating bounding boxes around objects and assigning appropriate labels. Such annotations provide essential information for machine learning models to learn object identification and classification. We utilized the LabelImg tool, known for its user-friendly interface tailored for deep learning projects. The tool's simplicity facilitates efficient and accurate object labelling, supporting various output formats like PASCAL VOC, Txt and YOLO. Figure 2 illustrates the labelling process using LabelImg, which involves drawing boxes around objects for machine recognition and assigning corresponding labels. In our specific environment, we have identified a set of 53 labels corresponding to specific locations or items within our campus, creating a direct association between labels and campus areas.

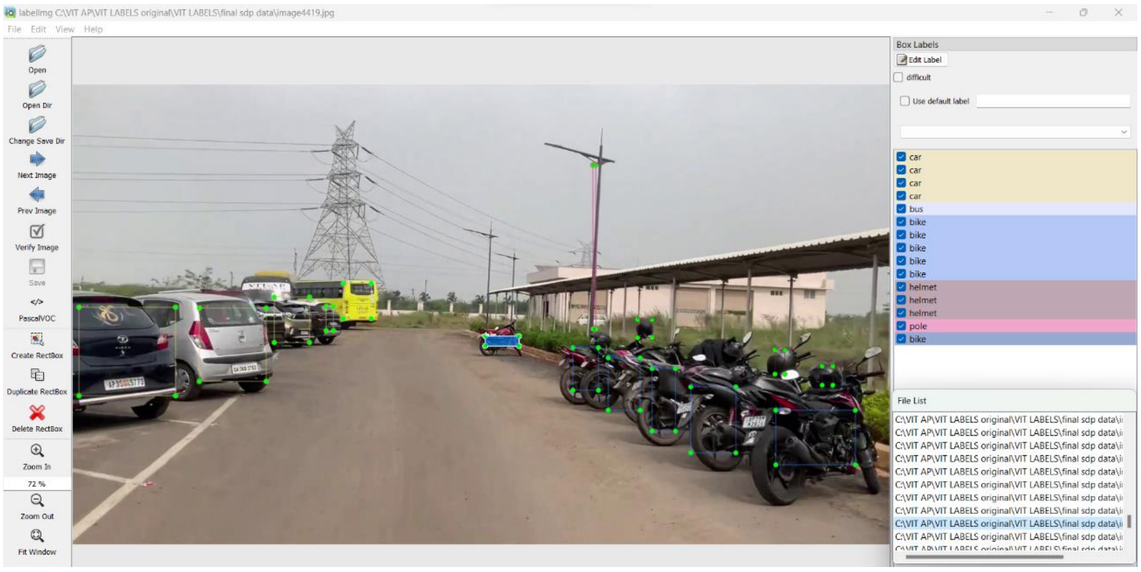

Figure 2. Image annotation process.

Data Augmentation

With the annotated data in hand, the next step in the workflow involves curating and preprocessing the dataset to ready it for training. Data augmentation is a critical component of this process, as it helps enhance the diversity of the dataset. The steps for data curation and preparation are outlined in Figure 3. Data augmentation includes techniques like rotation, flipping, scaling, and brightness adjustments, which generate various versions of the original images. The model is exposed to a wide range of visual patterns and configurations by augmenting the data, as shown in Figure 4. This exposure improves the model's ability to generalize its learning, enhancing object recognition in different conditions. Furthermore, the model becomes more resilient and adept at handling real-world situations, regardless of object orientation, size, or brightness changes.

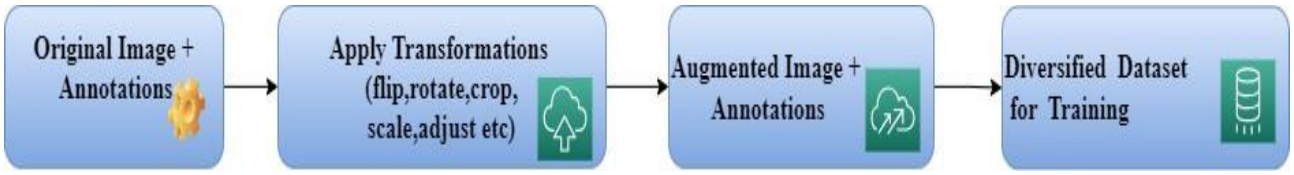
Original Image + Annotations
Apply Transformations (flip,rotate,crop, scale,adjust etc)
Augmented Image + Annotations
Diversified Dataset for Training

Figure 3. Preprocessing the dataset to prepare for training.

Data Splitting

After performing data augmentation, the dataset was expanded to include 5264 samples spanning 53 classes. This enhancement incorporates a variety of variations, thereby enhancing the model's capacity to generalize and enhance object detection accuracy. The subsequent important step involves partitioning this augmented dataset into three distinct subsets: training, validation, and testing sets, maintaining a distribution ratio of 7:2:1. To guarantee an equal distribution of samples across all classes and prevent any potential biases that may affect the model's performance during training and evaluation, the entire dataset of 5264 samples is initially randomized.

The dataset is divided according to the specified ratio following the random shuffling. While random shuffling guarantees sample distribution across classes, maintaining class balance within each subset is equally important. Stratified sampling techniques are employed to achieve this balance. Stratified sampling is a statistical method commonly used to ensure that the distribution of samples in the subsets mirrors the proportions of different classes in the original dataset. In the proposed work, stratified sampling upholds class balance within the training, validation, and testing sets even after data augmentation. This strategy enables the model to generalize effectively and provide accurate predictions across all classes, including those with limited samples. The combination of stratified sampling and data augmentation results in a well-balanced and diverse dataset, optimizing the deep learning model's potential for accurate detection in the campus environment, regardless of class imbalances.

Once the dataset is divided, the training set is utilized to train the deep learning model, the validation set assists in monitoring the model's performance and adjusting hyperparameters, and the testing set is used for independent evaluation to assess the model's generalization on unseen data.

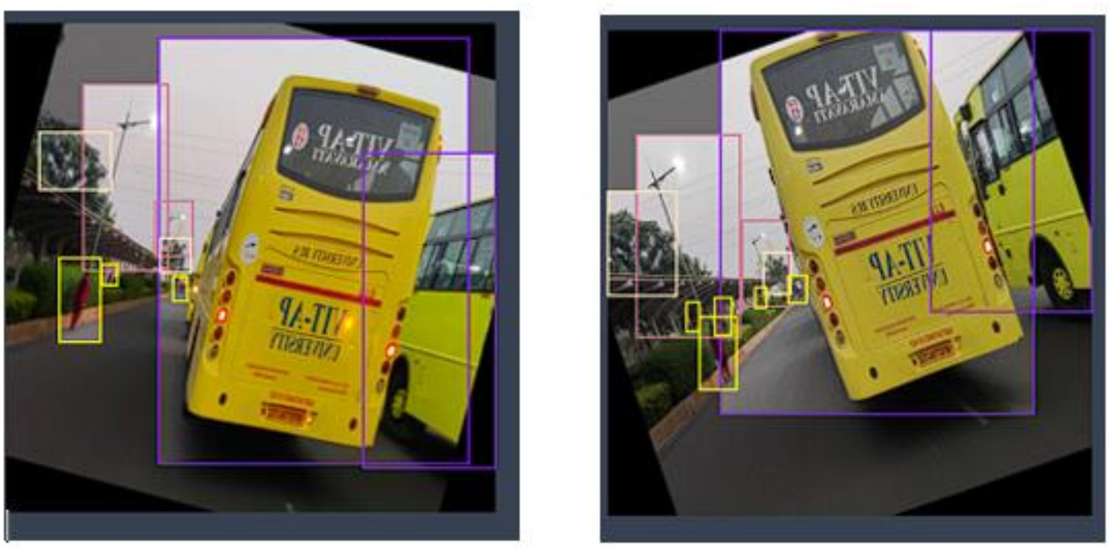

Figure 4. Data augmentation: cropping and horizontal flipping techniques

Proposed architecture of Yolov5 model

YOLOv5 is a state-of-the-art object detection model that can be trained on custom datasets. It offers a balance between performance and speed. The architecture of Yolov5 is shown in Figure 5. The YOLOv5 consists of two main components, the backbone and the head. The backbone of YOLOv5 comprises various modules that work together to extract features from the input image. As shown in Figure5, it starts with the Focus module, a lightweight convolutional layer that efficiently processes the image by focusing on relevant regions. The Focus module is used to increase the receptive field of the network without increasing the number of parameters. This is done by combining the features from different layers of the network. The Conv module follows, consisting of standard convolutional layers that learn basic features like edges, shapes, and textures. Next, the backbone employs the C3 module, which stands for CSPNet bottleneck with three convolutions. The C3 module incorporates cross-stage connections, promoting efficient feature fusion and enhancing information flow throughout the network. This results in improved gradient flow during backpropagation, leading to stable and efficient training.

The CSPDarknet53 architecture, consisting of 53 layers, hierarchically uses these modules. Each layer in the hierarchy builds upon the features learned by the previous layers, allowing for the extraction of increasingly complex and abstract features from the input image. The final layer of the CSPDarknet53 architecture produces a feature map that is used by the neck and head modules to predict the bounding boxes and class labels of the objects in the image. Additionally, YOLOv5 includes the Spatial Pyramid Pooling (SPP) module in its backbone. The SPP module captures information at multiple scales by pooling features at different levels, enabling the model to detect objects of various sizes and scales in the input image. The final layer of CSPDarknet53 produces a feature map, which the head modules utilise for object detection. The head of the YOLOv5 architecture is responsible for making predictions based on the features extracted from the previous stages. It predicts bounding boxes, class probabilities, and objectness scores for different anchor boxes. In the head, up sample employs the nearest neighbour interpolation method. In the nearest-neighbour method, the value of a new pixel is determined by the value of the nearest pixel in the original feature map. This method is computationally efficient and preserves the pixel values. The concat operation integrates features from diverse layers effectively. Finally, the detect module employs anchor boxes to predict object location and class in the image. During training, the smart optimizer initializes the parameters like learning rate and momentum to tune the hyperparameters. The model is stabilizing by averaging model weights.

Hyper Parameters Tuning

Following the initialization of specific parameters into training, the subsequent steps involve tuning hyperparameters to optimize the model's performance. Among various hyperparameters, the learning rate plays a key role in determining the step size during gradient descent, influencing the model's convergence rate and overall training stability. A learning rate that is too high might lead to overshooting and unstable training, while a rate that is too low could slow down convergence. This hyperparameter is finely tuned to strike a balance between swift convergence and stability, ensuring efficient training progress.

Another critical parameter is the batch size, which governs the number of samples used in each iteration of training. A larger batch size might accelerate training but could consume excessive memory and hinder convergence. Conversely, a smaller batch size might offer better generalization, particularly with limited data, but could lead to longer training times. The optimal batch size is determined through experimentation, considering computational constraints and desired convergence behaviour. Additionally, the choice of optimizer which is used here is Stochastic Gradient Descent (SGD) plays a key role in shaping the optimization process. The optimizer's adaptive learning rate adjustments impact how quickly the model learns and navigates the loss landscape. Here, the learning rate was set to 0.01.

Regularization techniques like weight decay (set to 0.0005) and dropout are also fine-tuned to prevent overfitting, enhancing the model's ability to generalize from the training data to unseen examples. By carefully configuring these hyperparameters based on validation performance, the deep learning model can be fine-tuned to achieve its highest potential accuracy.

A Grid Search tuning method was utilized with a batch size of 12, 16, and 32, SGD optimizer, 0.01 learning rate, 0.0005 weight decay to find the optimal configuration of these hyperparameters. This exploration allows us to identify the combination that yields the best performance on the validation dataset. The impact of these choices, particularly the effect of batch size on mAP, will be visualized and discussed in the results and discussion section.

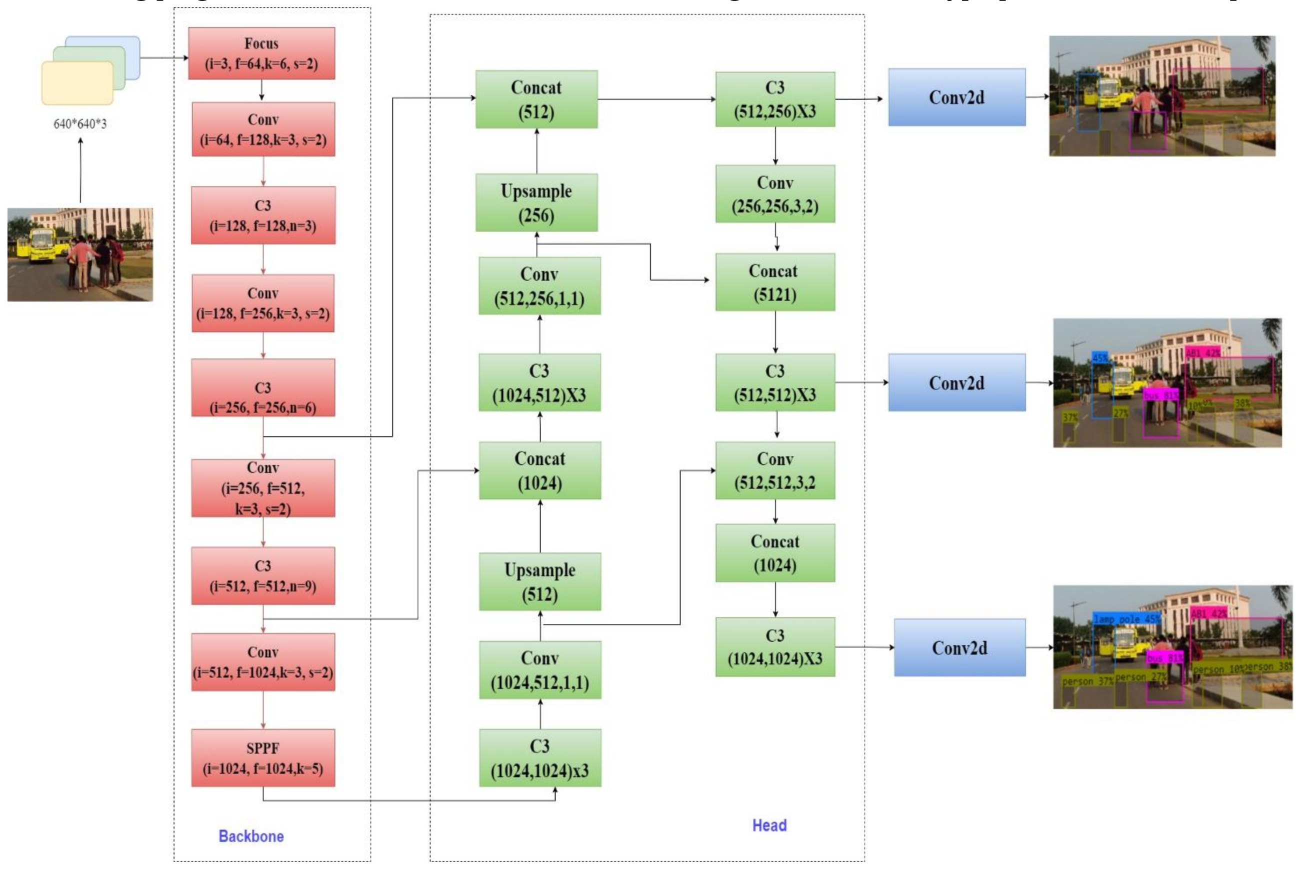

Figure 5. Yolo model architecture.

Results and Discussion

This section discusses the findings of the YOLOv5 object detection model trained on the campus environment dataset. We analysed the model's performance using a variety of evaluation metrics commonly employed in object detection tasks. These metrics include precision, recall, mean Average Precision (mAP), and Intersection over Union (IoU) at different thresholds. Precision reflects the proportion of correctly identified objects among the predicted ones. Recall, on the other hand, indicates the percentage of actual objects that the model successfully detected. mAP provides a more comprehensive view of the model's performance by summarizing the average precision (AP) across all object categories at various IoU thresholds. IoU, measured between the predicted and ground truth bounding boxes, signifies the degree of overlap between them, indicating the accuracy of object localization. By examining these metrics, we can assess the effectiveness of the YOLOv5 model in accurately detecting and classifying objects within the campus environment.

Evaluation Metrics and Performance Analysis

Choosing appropriate metrics for evaluating object detection systems is key to assessing their effectiveness. As precise bounding boxes must be delineated around identified objects in an image, object detection poses a formidable challenge. Object detection accuracy is commonly measured using Equations (1) through (4).

$$\text{Precision}=\frac{\text{True Positive (TP)}}{\text{True Positive (TP)+False Positive (FP)}} \quad (1)$$

$$\text{Recall}=\frac{\text{True Positive (TP)}}{\text{True Positive (TP)+False Negative (FN)}} \quad (2)$$

$$\text{mAP}=\sum_{N}^{i=1}\text{AP}_i \quad (3)$$

$$\text{IoU}=\frac{\text{True Positive(TP)}}{\text{True Positive(TP)+False Negative(FN)+False Positive(FP)}} \quad (4)$$

True positive (TP) are accurate detections of objects that are actually present in the photograph. An error in detection is known as a false positive (FP), which occurs when the network detects an object that does not exist in the picture. False negatives (FN) occur when the network does not detect an object in the picture that actually exists. If the network detects an object correctly in an image, it is a TP. If the network detects an object incorrectly, it will mark it as an FP, which means it will not determine whether or not it is present in the image. In FN, objects exist in an image but are not detected by the network. The detection of a TN occurs when the image does not contain an object.

Further, the detection is classified as correct or incorrect by comparing the IoU with a given threshold. It is necessary to specify the AP metric for each IoU threshold value. A person's average accuracy rate is determined by the integral of the P index and the R index, which is the area under the P–R curve; an average accuracy of a mean is determined by summing the AP values of all categories and then dividing them by each category, i.e., by the average.

Precision, Recall and mAP

Figure 6 presents accuracy curves for precision, recall, and mAP scores. These metrics are plotted for both the training and validation sets, allowing us to observe the model's improvement during the training process. The model achieves higher precision and recall values on the training set compared to the validation set. However, the validation set performance indicates the model's ability to generalize to unseen data, which is crucial for real-world applications.

To explore the effect of batch size on training performance, we evaluated the model with three different batch sizes (12, 16, and 32). The training duration was adjusted accordingly, with 50 epochs for a batch size of 12, 100 epochs for 16, and 100 epochs for 32. The observations from this experiment are presented in Table 1 and visualized in Figure 7. As shown in Table 1, a batch size of 32 generally yielded the best performance in terms of precision (0.862), although recall (0.761) was slightly lower compared to a batch size of 16 (0.775). The mAP values followed a similar trend, with the highest mAP50 (0.843) achieved with a batch size 16. However, it's important to consider the trade-off between performance and training time. A larger batch size (32) might lead to faster training convergence, while a smaller batch size (12) might require less memory but could take longer to train.

Further, we evaluated the model's performance on a set of 5264 images containing a total of 11335 instances. This evaluation was conducted using various IoU thresholds. Table 2 displays the corresponding precision, recall, and mAP50 scores for each IoU threshold. The model achieves higher precision and recall values at lower IoU thresholds (indicating less strict bounding box overlap requirements). For instance, at an IoU of 0.25, the model obtains a precision of 0.677, a recall of 0.519, and a mAP50 of 0.556. These scores decrease as the IoU threshold increases (becoming more strict), indicating a trade-off between the number of detections and their

Table 1. Metrics with varying batch size and epoch.

Batch size	epoch	Precision	Recall	mAP 50	mAP 50-95
12	50	0.81	0.705	0.732	0.446
16	100	0.826	0.775	0.843	0.521
32	100	0.862	0.761	0.821	0.504

localization accuracy. By analyzing these results, we gain valuable insights into the effectiveness of the YOLOv5 model for object detection in a campus environment. The model demonstrates a good balance between precision and recall, and its performance adapts based on the chosen IoU threshold. The exploration of different batch sizes further highlights the importance of hyperparameter tuning for optimal model performance.

Loss function

The training process also involves calculating a loss function to assess the model's performance on each iteration. This loss is then used to update the model's weights through an optimizer like Stochastic Gradient

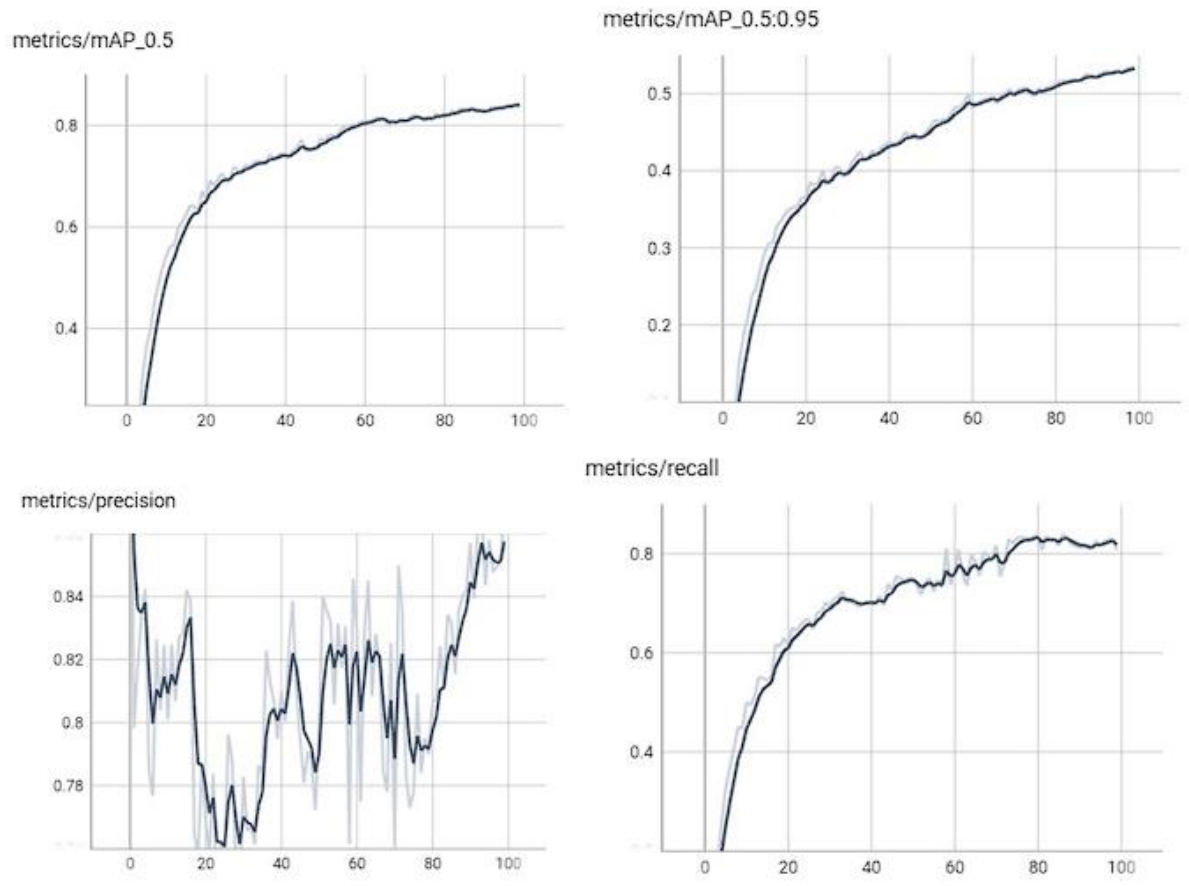

Figure 6. Precision, recall, mAP (0.5) parameters, and class object loss for training epochs.

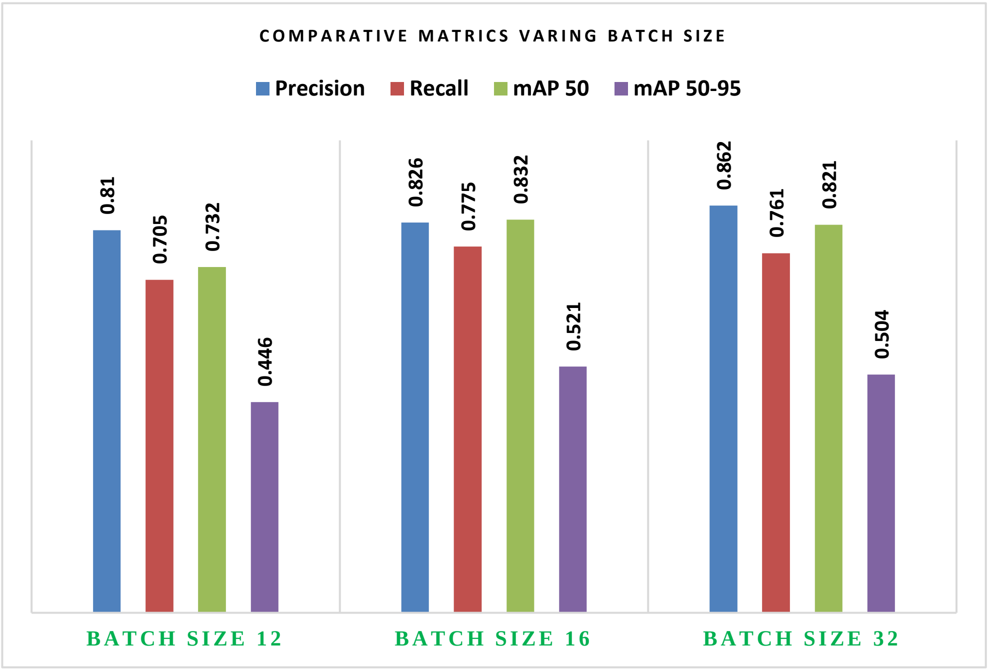

Figure 7. Performance metrics for different batch sizes.

Descent (SGD) mentioned earlier. The goal is to minimize the loss function over the training iterations, leading to improved object detection accuracy. Validation and testing are performed with the weights that achieve the minimum loss during training. These weights represent the best-performing model on the training data. Finally, the model with these optimal weights is used to predict objects in unseen test data.

Table 2. Performance of P, R, mAP for various values of IoU.

IoU	Precision	Recall	mAP50
0.25	0.677	0.519	0.556
0.5	0.672	0.525	0.57
0.75	0.659	0.5	0.547

The total loss function in YOLOv5 is a combination of three individual loss components: the one responsible for finding the bounding-box coordinates **(**coord loss**)**, the bounding-box score prediction **(**objectness loss**)**, and the class-score prediction (class loss). Each of these components is weighted and combined to form the total loss. The architecture aims to minimize this loss during training to improve object detection accuracy.

Coord loss is accurate in relation to the ground truth bounding box coordinates. The loss is calculated using mean squared error (MSE) between the predicted coordinates and the ground truth coordinates as shown in equation (5)

$$Coord\_Loss = \lambda_{coord} \times \sum[i,j,k]\left((x_{pred} - x_{gt})^2 + (y_{pred} - y_{gt})^2 + (\sqrt{w_{pred}} - \sqrt{w_{gt}})^2 + (\sqrt{h_{pred}} - \sqrt{h_{gt}})^2\right) \quad (5)$$

Here, (x_{pred}, y_{pred}) are the predicted bounding box center coordinates, (w_{pred}, h_{pred}) are the predicted width and height of the bounding box, and $(x_{gt}, y_{gt}, w_{gt}, h_{gt})$ are the corresponding ground truth values. The summation is performed over the grid cells (i, j) and anchor boxes (k).

Objectness loss encourages the model to predict high scores for grid cells containing objects and low scores for cells without objects. It utilizes the Mean Squared Error (MSE) between the predicted objectness score and the ground truth objectness score as shown in equation (6).

$$Obj\_Loss = \lambda_{obj} \times \sum[i,j,k]\left((obj\_pred - obj\_gt)^2\right) \quad (6)$$

Here, obj_pred is the predicted objectness score, and obj_gt is the ground truth objectness score for the corresponding grid cell and anchor box. Class loss ensures that the predicted class probabilities align well with the actual class labels. It's calculated using the Mean Squared Error (MSE) between the predicted class scores and the ground truth class scores as shown in equation (7)

$$Class\_Loss = \lambda_{cls} \times \sum[i,j,k,c]\left((class\_pred - class\_gt)^2\right) \quad (7)$$

Here, class_pred is the predicted class score for class c, and class_gt is the ground truth class score for the same class. The summation is performed over grid cells (i,j), anchor boxes (k), and classes (c). The total loss is a linear combination of the three components, each multiplied by a corresponding scalar parameter (λ_coord, λ_obj, λ_class). These scalar parameters control the relative importance of each component in the loss function. The total loss can be summations of above three loses and it can be expressed in equation (8).

$$Total\_Loss = Coord\_Loss + Obj\_Loss + Class\_Loss \quad (8)$$

Additionally, an IoU factor is further incorporated to modulate the contributions of these loss components. This factor adjusts the loss based on the accuracy of the predicted bounding boxes compared to the ground truth. Higher IoU (better overlap) leads to lower loss, guiding the model towards more precise object localization. YOLOv5 uses these loss components and IoU modulation to guide the training process towards accurate object detection. The model learns to minimize this combined loss by adjusting its parameters during training, as shown in Figure 8. The loss curves consistently display a descending trajectory, signifying the successful minimization of training and validation losses throughout the training process. Simultaneously, the metrics curves exhibit a consistent upward trend, indicating progressive model enhancement over successive training iterations. A YOLOv5 model that is the smallest and fastest was selected for the proposed system (YOLOv5s).

Table 3. Performance of the model YOLOv5s for custom data validation.

Class	Precision	Recall	mAP50	mAP50-95
All	0.851	0.831	0.843	0.534

Table 3 summarizes entire validation set the performance of the YOLOv5s model. It shows the model achieved a precision of 0.851, recall of 0.831, and mAP50 of 0.843 across all classes and instances in the validation set. The mAP50-95, which considers a broader IoU threshold range (0.5 to 0.95), is 0.534. These results indicate that the YOLOv5 model, trained with the combined loss function and IoU modulation, achieved good performance in object detection for the specific campus environment dataset.

These results on the custom dataset are further demonstrated to existing object detection techniques, as shown in Table 4. The proposed approach achieves mAP of 84.3%. This method has shown improvement over previously established methods such as YOLO9000 (19.7% mAP), earlier versions of Faster R-CNN (53.3% mAP in 2014 and 35.9% mAP in 2015). It also demonstrates improvement over recent works such as SSD (74.3% mAP) and Faster R-CNN (70.9% mAP in 2015) on similar datasets (PASCAL VOC and COCO). Our approach outperforms CNN-based methods like CNN-MS COCO (65.7% mAP), highlighting the effectiveness of deep learning architectures like YOLOv5 for object detection tasks. In 2023, Jifeng Dai et al. demonstrated strong performance (83.6% mAP) utilizing the PASCAL VOC dataset, which is comparable to proposed metric. it's important to consider potential differences in the datasets and task complexities. Our custom dataset might pose unique challenges compared to the publicly available datasets used in other works. Due to its customizability, the proposed YOLOv5-based approach demonstrates promising results, achieving high mAP and potentially offering advantages in specific application domains.

Table 4. Comparison of Object Detection Methods with existing literature

Author and Year	Method and Dataset	Limitations	mAP
Jifeng Dai et al., 2023	PASCAL -VOC datasets	The R-FCN system presented in the paper was intentionally kept simple, potentially limiting the exploration of more complex extensions or improvements	83.60%
Alexey Bochkovskiy et al., 2020	CNN- MS COCO dataset	Does not extensively discuss the potential limitations or drawbacks of the YOLOv4 model.	65.70%
Joseph Redmon et al., 2017	YOLO9000 - COCO dataset	Down sampling factor of 32 results in an output feature map of 13 x 13, which may affect the detection of objects occupying the center of the image.	19.70%
Wei Liu et al., 2016	SSD-VOC2007 dataset	SSD faces challenges in classifying small objects without a follow-up feature resampling step, impacting performance	74.30%
Ross Girshick et al., 2015	PASCAL – COCO dataset	SPPnet has limitations in updating convolutional layers, which may affect the accuracy of very deep networks	35.90%
Jifeng Dai et al., 2015	PASCAL –VOC 2012 trainval	Lack of significant improvement in accuracy with feature sharing, absence of exploration into post-processing techniques such as CRF for refining instance mask boundaries	70.90%
Ross Girshick et al., 2014	PASCAL- VOC dataset	The method relies on labelled training data, which can be scarce, impacting its generalizability	53.3%.
Present work	YoloV5-Custom dataset		84.30%

Prediction of objects in image

After training the models and evaluating them on a separate testing set, we validated their performance on entirely new images unseen during training. These unseen images are crucial for assessing how well the models generalize to real-world scenarios. Figure 9(a) visually depicts the model's object detection results. Bounding boxes are identified around the objects identified in the image. A summary of a label identifying the detected object that appears within each bounding box for all classes present in the dataset is shown in Fig.9(b). This visualization demonstrates the model's ability to detect objects in new images, along with the assigned class labels and potential confidence scores (percentages) indicating the model's certainty in its detections.

Discussion

The dataset contains images from various environments, divided into 70% for training, 20% for testing, and 10% for validation purposes. The architecture begins with a series of convolutional layers to extract features from input images. These layers have different filter sizes and strides to extract hierarchical features gradually. Multiple convolutional layers enhance feature representation, and the SPPF module captures context information across scales to identify objects of varying sizes. Inspired by YOLO, the detect layer handles the final object detection using anchor boxes. The model has 214 layers and 7,181,449 parameters, indicating its capacity to learn complex features. The number of gradients matches the parameters, emphasizing the training complexity. The model described with 214 layers and 7 million parameters suggests a high level of complexity. While this can be beneficial for capturing intricate features, it also leads to increased training time (as shown in Table 5) and potentially higher computational demands for running the model on new images. Exploring techniques for model compression or knowledge distillation could be beneficial for reducing complexity while maintaining good performance.

Figure 10 (a) provides a comprehensive analysis of the bounding box distribution within the dataset. It shows a bar chart, where each bar represents a class in the dataset. The height of each bar corresponds to the number of annotations associated with that particular class. This visualization helps identify potential class imbalances in the dataset, where some classes might have significantly fewer or more annotations compared to others. Based on the location and size of each bounding box in the dataset, Figure 10(b) illustrates the distribution of bounding boxes in the dataset. To ensure that the model recognizes objects properly, there should be enough variation in the position and size of the bounding boxes in the dataset. The figures in Figure 10 (c) and (d) illustrate how the bounding boxes are distributed across the dataset on the basis of their position and size. Using this graph, it is possible to determine whether the bounding boxes are evenly distributed throughout the dataset or whether there are areas that are heavier than others. Due to the variability in size and position of objects in the image, it

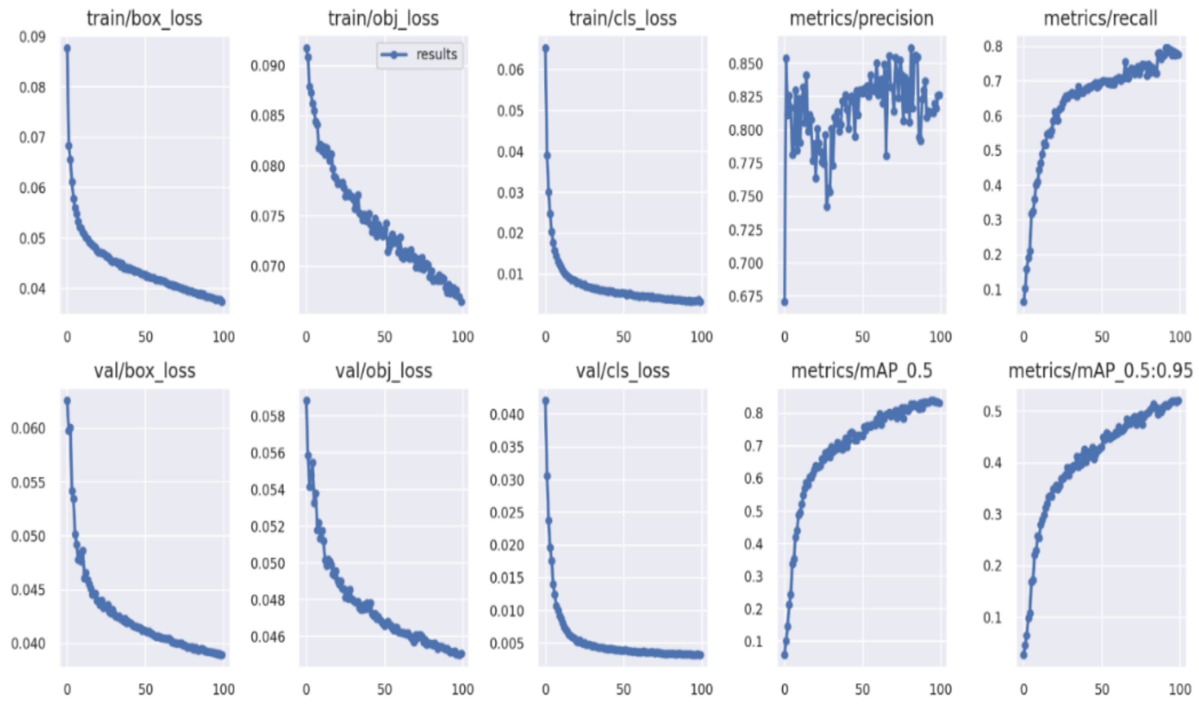

Figure 8. Box loss, objectness loss, classification loss, precision, recall and mAP over the training and validation set's training epochs.

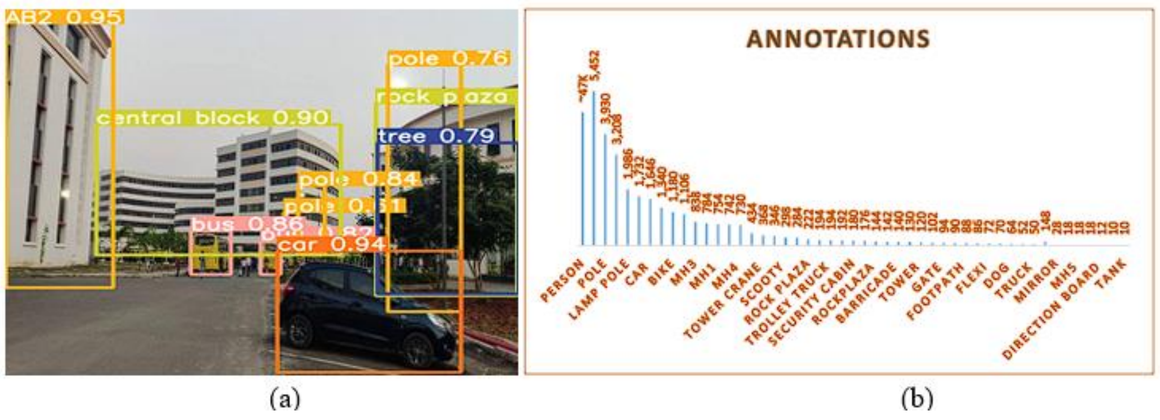

Figure 9. a) Output of object detection and classification in real-time b) Class distribution of the detected objects.

is essential that the model will be able to detect them correctly.

Moreover, the analysis of bounding box distribution (Figure 10) helps identify potential biases in the dataset. If the bounding boxes are concentrated in specific image regions or exhibit limited size variations, the model might struggle with objects located differently or having different sizes during deployment. Addressing this could involve collecting additional data to achieve a more diverse distribution.

The training process utilized both the training and validation sets to optimize the YOLOv5 model. Various loss functions were computed during training to measure the model's performance. Throughout the training, the combined loss function value exhibited a decreasing trend. This suggests SGD optimizer effectively adjusted the network's weights and parameters, leading to improved performance. As reflected in the decrease in loss, the model likely achieved significant gains in accuracy, recall rate, and average accuracy.

Table 5 summarizes the training times observed for different training epochs using the YOLOv5s model. As the number of epochs increases, the training time also increases. This is because each epoch involves processing the entire training dataset. The table presents training times for different epoch values: 1.377 hours for 50 epochs, 2.568 hours for 100 epochs, and 3.738 hours for 150 epochs. This highlights the growing computational cost associated with more extensive training iterations.

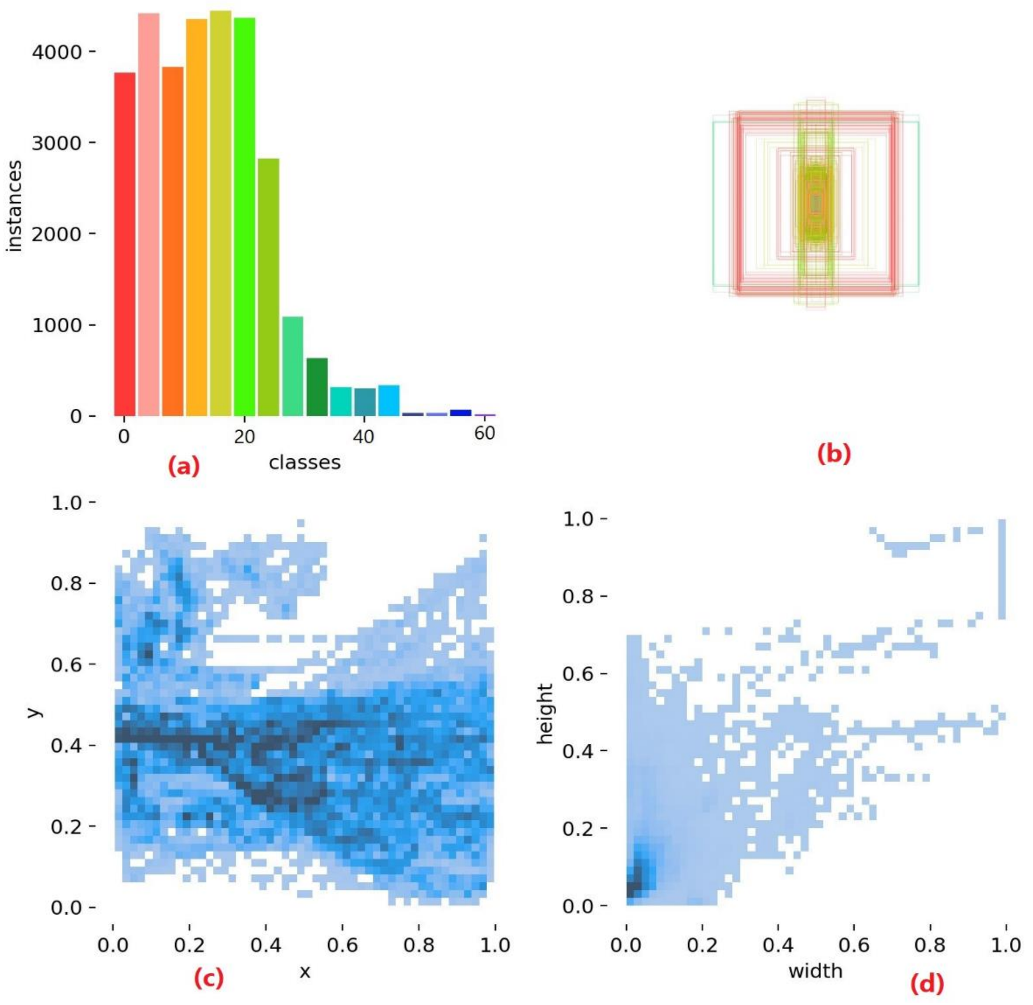

Figure 10. A visual representation of the dataset. (a) An indication of the number of annotations per class, (b) visualization of the bounding box location and size, (c) statistics about the bounding box position, and (d) Size of bounding boxes statistically distributed.

Conclusion

This paper proposes an object detection algorithm building upon YOLOv5. We evaluated our custom model using standard metrics like precision, recall, mAP, and IoU at various thresholds. These metrics assessed the model's ability to detect and classify objects accurately, predict bounding boxes precisely, and handle diverse object categories. While achieving a good precision of 0.851, the significant improvement lies in the high recall of 0.831 and mAP of 0.843. The high recall signifies the model's success in identifying custom objects in the test data, leading to tangible and meaningful results. While this research utilized the YOLOv5s model, exploring newer architectures like could potentially enhance accuracy or efficiency depending on the specific task and

Table 5. Training time vs number of epoch

S.No	epoch	Training-time in hours	mAP %
1	50	1.377	0.741
2	100	2.568	0.843
3	150	3.738	0.852

computational resources. Furthermore, investigating techniques for faster inference through hardware acceleration or model optimization could be a valuable future direction for real-time applications.

Conflicts of Interest

The authors declare no conflict of interest.

References

Alexe, B., Deselaers, T., & Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(11), 2189-2202. https://doi.org/10.1109/tpami.2012.28

Amjoud, A. B., & Amrouch, M. (2023). Object detection using deep learning, CNNs and vision transformers: a review. *IEEE Access*, 35479-35516. https://doi.org/10.1109/access.2023.3266093

Banerjee, M., Goyal, R., Gupta, P., & Tripathi, A. (2023). Real-Time Face Recognition System with Enhanced Security Features using Deep Learning. *Int. J. Exp. Res. Rev.*, *32*, 131-144. https://doi.org/10.52756/ijerr.2023.v32.011

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv: 2004, 10934*, 1-17. https://doi.org/10.48550/arXiv.2004.10934

Busta, M., Neumann, L., & Matas, J. (2017). Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision*, *2017*, 2204-2212. https://doi.org/10.1109/iccv.2017.242

Dai, J., He, K., & Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, *2016*, 3150-3158. https://doi.org/10.1109/CVPR.2016.343

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems*, *29*. https://doi.org/10.48550/arXiv.1605.06409

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems*, *29*. https://doi.org/10.48550/arXiv.1605.06409

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 886-893. https://doi.org/10.1109/cvpr.2005.177

Dhruv, P., & Naskar, S. (2020). Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): A review. *Machine learning and information processing: proceedings of ICMLIP*, *2019*, 367-381. https://doi.org/10.1007/978-981-15-1884-3_34

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. *multimedia Tools and Applications*, *82*(6), 9243-9275. https://doi.org/10.1007/s11042-022-13644-y

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, *2015*, 1440-1448. https://doi.org/10.1109/iccv.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, *2014*, 580-587. https://doi.org/10.1109/cvpr.2014.81

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, *2014*, 580-587.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, *2017*, 2961-2969. https://doi.org/10.1109/iccv.2017.322 https://doi.org/10.1109/cvpr.2014.81

Ke, Y., & Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. 2*, II-II. https://doi.org/10.1109/cvpr.2004.1315206

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/ 10.1038/nature14539

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. Springer International Publishing, In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21-37. https://doi.org/10.48550/arXiv.1512.02325

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, *2015*, 3431-3440. https://doi.org/10.1109/cvpr.2015.7298965

Naganuma, K., & Ono, S. (2022). A general destriping framework for remote sensing images using flatness constraint. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1-16. https://doi.org/10.48550/arXiv.2104.02845

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, *2017*, 7263-7271. https://doi.org/10.1109/cvpr.2017.690

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. https://doi.org/10.48550/arXiv.1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing systems*, *28*. https://doi.org/10.48550/arXiv.1506.01497

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information Processing Systems*, *28*. https://doi.org/10.48550/arXiv.1506.01497

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. https://doi.org/10.48550/arXiv.1312.6229

Srivastava, R., & Tripathi, M. (2023). Systematic Exploration Using Intelligent Computing Techniques for Clinical Diagnosis of Gastrointestinal Disorder: A Review. *Int. J. Exp. Res. Rev.*, *36*, 265-284. https://doi.org/10.52756/ijerr.2023.v36.026

Viola, P., & Jones, M. (2001). 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*. *1*, I-I. https://doi.org/10.1109/cvpr.2001.990517

Viola, P., & Jones, M. (2001). 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*. *1*, I-I. https://doi.org/10.1109/cvpr.2001.990517

How to cite this Article:

Dontabhaktuni Jayakumar and Samineni Peddakrishna (2024). Performance Evaluation of YOLOv5-based Custom Object Detection Model for Campus-Specific Scenario. *International Journal of Experimental Research and Review*, *38*, 46-60.

DOI: https://doi.org/10.52756/ijerr.2024.v38.005