IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 1, Ser. 3 (Jan. – Feb. 2025), PP 53-62
www.iosrjournals.org

Building An Object Detection Model For Smart Cities Using Yolov7-Tiny Architecture

Chi-Chang Chen
Information Engineering Department, I-Shou University, Taiwan

Abstract:
With the continuous advancement of smart city technologies, object detection has increasingly found applications in urban management, autonomous driving, public safety, and other fields. This paper aims to introduce the process of developing an object detection model for smart cities using the YOLOv7-tiny architecture. By integrating training data comprising approximately 50 categories from the COCO Dataset and Kaggle Datasets, with around 3,000 images per category, the model was trained and its performance analyzed. This paper will provide a detailed explanation of the YOLOv7-tiny model architecture and algorithm, assess its application effectiveness in smart city scenarios, and offer future development directions and improvement suggestions.

Keywords: *Object Detection Model; Machine Learning; Smart City; YOLO; COCO Dataset.*

Date of Submission: 08-02-2025 Date of Acceptance: 18-02-2025

I. Introduction

Object Detection Technology in Smart Cities:

Smart cities represent a crucial direction for future urban development, aiming to enhance operational efficiency and improve residents' quality of life through data collection and analysis. Object detection technology, as part of smart cities, is responsible for identifying and locating various target objects, such as pedestrians, vehicles, and traffic signs, in images captured by cameras or other sensing devices. These detection results can be applied in traffic management, public safety, environmental monitoring, and more.

In smart city scenarios, object detection systems need to be efficient, real-time, and accurate. YOLOv7-tiny, as a lightweight object detection model, is an ideal choice for smart city applications due to its efficient inference speed and relatively good accuracy [1,2,3].

Selection and Preparation of Training Datasets:

To train a model capable of accurately identifying various objects in smart city scenarios, this study selected the COCO Dataset and Kaggle Datasets as the primary training data sources.[4] These datasets contain a wealth of annotated images covering common object categories in smart cities, such as vehicles, pedestrians, traffic signs, and buildings.

In this study, we selected approximately 50 categories of images for training, with each category containing about 3,000 fully annotated images. These images were preprocessed and used to train the YOLOv7-tiny model [5]. The COCO Dataset is a widely used object detection dataset containing data annotations for 80 categories, covering common objects in everyday life. Kaggle Datasets, on the other hand, provide more data specifically targeting certain scenes and object categories, such as stray animals and environment-related objects, supplementing the COCO Dataset to meet the specific needs of smart city applications.

Data Annotation and Preprocessing:

In building the YOLOv7-tiny smart city object detection model, we primarily used annotated images from the COCO Dataset and Kaggle Datasets as training data[6]. These datasets already include rich and detailed annotation information, including bounding boxes and corresponding category labels, covering common object categories in smart city applications, such as pedestrians, vehicles, and traffic signs.

Although these datasets provide a good foundation for annotation, we modified and supplemented the annotations of some images to better adapt the model to specific smart city scenarios. These modifications focused on the following aspects:

- Adding Category Annotations: In certain smart city applications, we found that some object categories that need to be identified may not be fully annotated in the original datasets. For example, in special scenarios such

as stray animals and garbage accumulation, we manually added annotations for these categories to ensure the model can identify and process these objects.

- Correcting Annotation Errors: During the process, we found that some images contained annotation errors or were not precise enough, which could affect the model's training performance. Therefore, we corrected these annotations to ensure that each object's bounding box and category label are accurate.
- Supplementing Missing Annotations: Some objects in certain images may have been omitted in the original annotations. We carefully reviewed these images and manually supplemented the missing annotations to ensure the completeness of the training data.

After completing the annotations, we performed data preprocessing, including standardizing the images to maintain consistent sizes and color distributions during the training process. Additionally, to enhance the model's generalization capability, we applied various data augmentation techniques, such as random cropping, rotation, translation, and brightness adjustment. These techniques help the model maintain stable detection performance under different lighting, angles, and backgrounds.

In conclusion, through modifications and supplementation of annotations on some images, along with effective data preprocessing, we ensured that the YOLOv7-tiny model can accurately and comprehensively identify a wide range of objects in smart cities, providing a solid data foundation for subsequent training.

II. Detailed Analysis Of The Yolov7-Tiny Model And Algorithm

In-depth Analysis of the YOLOv7-tiny Model Architecture:

YOLOv7-tiny is a lightweight version of the YOLO (You Only Look Once) series models, designed for resource-constrained environments such as embedded systems and mobile devices[7, 8, 9]. Since its inception, the YOLO series has been widely acclaimed for its balance between speed and accuracy. As one of the most lightweight versions in the series, YOLOv7-tiny's architecture has been meticulously designed to significantly reduce computational costs and inference latency while maintaining a certain level of detection accuracy.

Feature Extraction Module:

The feature extraction module in YOLOv7-tiny is based on a multi-layer structure of Convolutional Neural Networks (CNNs). The core of CNNs lies in their ability to effectively extract multi-level features from images, ranging from low-level edges and textures to high-level shapes and objects. YOLOv7-tiny leverages this capability by stacking multiple convolutional layers to gradually extract significant features from the images.

Unlike other YOLO versions, YOLOv7-tiny places more emphasis on efficiency during the feature extraction process. In YOLOv7-tiny, the number of convolutional layers and the size of the convolutional kernels are carefully adjusted to maintain a lightweight model while effectively extracting multi-scale features. These features are crucial for object detection, as they help the model recognize various objects in the image, regardless of their size or position.

To further enhance the efficiency of feature extraction, YOLOv7-tiny introduces a technique known as "depthwise separable convolution." This convolution operation decomposes the standard convolution operation into depthwise convolution and pointwise convolution, significantly reducing computational complexity. This allows YOLOv7-tiny to operate on resource-constrained devices, such as mobile or edge computing devices, without excessively consuming resources.

Prediction Module:

After feature extraction, YOLOv7-tiny inputs these features into the prediction module. The primary function of the prediction module is to generate multiple candidate boxes (bounding boxes) and predict the class and confidence of objects within these boxes. The core of this process lies in how the model matches the extracted features with predefined anchors to determine whether a target object exists within each candidate box.

YOLOv7-tiny's prediction module utilizes the classic "single-stage detector" design of the YOLO series models. Unlike multi-stage detectors (such as Faster R-CNN [10]), YOLOv7-tiny completes both object classification and localization in a single step, greatly improving the model's inference speed. Specifically, YOLOv7-tiny achieves this by predicting multiple candidate boxes at each grid cell and assigning a confidence score to each box. The confidence score represents the model's confidence in the presence of an object within that box.

Another key technique in the prediction module is the design of "anchors." Anchors are predefined boxes that help the model better handle objects of varying sizes and shapes. YOLOv7-tiny uses a set of predefined anchors designed with various smart city scenarios in mind, such as pedestrians, vehicles, and traffic signs. The setup of these anchors allows the model to maintain good detection performance across different object sizes and shapes.

Loss Function Design:

The loss function in YOLOv7-tiny is crucial for model training. The design of the loss function determines how the model learns during training and ultimately impacts detection performance. The loss function of YOLOv7-tiny comprises three main components: localization loss, confidence loss, and classification loss[11].

•Localization Loss: Localization loss measures the difference between the model's predicted bounding box and the ground truth box. YOLOv7-tiny adopts IoU (Intersection over Union) as the metric for localization loss. IoU measures the overlap between the predicted box and the ground truth box; the greater the overlap, the higher the IoU value, and the lower the loss. This design ensures that the model generates more accurate bounding boxes, thus better localizing objects in the image.

•Confidence Loss: Confidence loss measures the model's confidence level in the presence of an object within the candidate box. YOLOv7-tiny's confidence loss design balances between positive examples (objects present) and negative examples (no objects present). By applying different weights to positive and negative examples, the model can more accurately identify real objects while reducing false detections in the background.

•Classification Loss: Classification loss measures the difference between the model's predicted object class and the actual class. YOLOv7-tiny uses cross-entropy loss to handle this part of the difference. Cross-entropy loss is commonly used in multi-class classification problems and can effectively address multiple category predictions, ensuring the model accurately classifies different objects.

Non-Maximum Suppression (NMS):

During the prediction process, YOLOv7-tiny generates a large number of candidate boxes, many of which may overlap or be similar. To ensure that each object is detected only once, YOLOv7-tiny employs a technique known as "Non-Maximum Suppression" (NMS)[12].

The core idea of NMS is to first select the box with the highest confidence score among all predicted boxes and then remove other boxes with high overlap (typically measured using IoU) with this box. This approach ensures that only the most optimal detection box is retained for each object, thereby reducing redundant detections and false positives.

The application of NMS is particularly important in YOLOv7-tiny because the model generates a large number of candidate boxes, many of which may overlap significantly. By applying NMS, the model can more accurately identify various objects in the image while minimizing overlapping detections, thus improving detection accuracy.

Training Process of YOLOv7-tiny:

The training process of YOLOv7-tiny is crucial for enhancing the model's performance. This process includes data preparation, model initialization, parameter tuning, backpropagation, optimization, and final model fine-tuning. The key steps and technical details of the training process are as follows:

Data Preparation:

The quality of training data directly affects the model's final performance. In the training of YOLOv7-tiny, image data from the COCO Dataset and Kaggle Datasets were used to construct the training and validation sets. These datasets cover various objects in smart city scenarios, such as vehicles, pedestrians, traffic signs, and buildings.

Data preprocessing is an important step before training. First, we standardized the images to ensure that the input data had similar scales and distributions. Then, we applied data augmentation techniques, such as random cropping, rotation, translation, and brightness adjustment, to expand the diversity of the training data. These augmentation techniques help the model maintain stable detection performance under different lighting, angles, and backgrounds.

Model Initialization and Parameter Tuning:

The training process of YOLOv7-tiny begins with model initialization. The model's initial parameters are usually derived from random weights or from a pretrained model that has already been trained on similar tasks. Choosing the appropriate initial weights can accelerate the model's convergence process and improve final detection accuracy.

In terms of parameter tuning, we used optimizers such as Adam and SGD to update the model's weights[13]. These optimizers can effectively adjust the model's parameters based on the gradient information from the loss function, allowing the model to gradually converge to a better state.

The learning rate is an important parameter during training; a learning rate that is too high may cause the model to oscillate during training, while a learning rate that is too low may result in slow convergence. To

address this, we introduced a learning rate decay strategy that dynamically adjusts the learning rate based on the progress of the training to ensure the model converges stably and effectively.

Backpropagation and Optimization:

In each training step, YOLOv7-tiny calculates the output through forward propagation and then calculates the difference between the prediction and the ground truth using the loss function. Next, the model uses the backpropagation algorithm to calculate the gradients and updates the weight parameters using the optimizer.

The core of the backpropagation algorithm lies in calculating the gradient of each parameter of the loss function using the chain rule. These gradients guide how to adjust the model's parameters to gradually reduce the loss function. YOLOv7-tiny uses Batch Gradient Descent, a technique that calculates the gradient using a small batch of data at each weight update[14]. This not only improves computational efficiency but also helps the model escape local minima.

Model Fine-tuning and Early Stopping Strategy:

At the final stage of training, we fine-tuned the model. This process usually involves adjusting the learning rate and other hyperparameters further when the model has already reached a certain accuracy level, aiming to achieve the best detection results.

Fine-tuning can help the model better adapt to specific application scenarios, such as object detection in smart cities. To prevent overfitting, we also adopted an Early Stopping strategy. This strategy monitors the loss value on the validation set and terminates training early when the loss no longer decreases significantly. This prevents the model from performing well on the training set but poorly on the validation set or in real-world applications.

Multi-Scale Training:

YOLOv7-tiny employs Multi-Scale Training, a technique that improves the model's generalization capability by randomly adjusting the input image size in each training batch[15]. Multi-scale training allows the model to adapt better to input images of different resolutions, which is particularly important in smart city applications, where cameras in real-world scenarios may have varying resolutions.

III. Algorithmic Aspects And Optimization Of Yolov7-Tiny

YOLOv7-tiny's algorithm design reflects its pursuit of efficiency and practicality. From feature extraction to prediction, loss calculation, and result filtering, every step in YOLOv7-tiny has been carefully designed and optimized to achieve the best performance.

Depthwise Separable Convolution:

Depthwise Separable Convolution is a key technique used in YOLOv7-tiny's feature extraction process. This convolution operation splits the standard convolution into two simpler operations: depthwise convolution and pointwise convolution. This decomposition significantly reduces the number of parameters and computation, speeding up inference.

Depthwise convolution extracts features by applying convolution operations separately to each input channel, while pointwise convolution integrates information across channels using 1x1 convolution. The advantage of this operation is that it retains the feature extraction capability of standard convolution while significantly reducing computational costs, which is particularly important for resource-constrained devices.

Anchor Design and Optimization:

The selection and design of anchors play a crucial role in YOLOv7-tiny. The shape and size of the anchors directly affect the model's detection performance for different types of objects. In smart city applications, objects in different scenes (such as pedestrians, vehicles, traffic signs) vary in size and shape, so the design of anchors needs to consider these variables fully.

The anchors in YOLOv7-tiny are designed based on the distribution of object sizes and shapes in the dataset. During training, the K-means clustering algorithm is used to analyze the bounding boxes of objects and find the most suitable set of anchors. These anchors cover the majority of objects' shapes and sizes in the dataset. This design enables the model to detect various objects more accurately in real-world applications, especially in diversified urban scenes.

Lightweight Design and Optimization Strategies:

The lightweight design of YOLOv7-tiny is a systematic process, involving everything from the design of convolutional layers to the reduction of parameters, as well as the application of model compression and

acceleration techniques. In addition to the aforementioned depthwise separable convolution, YOLOv7-tiny also employs other techniques to further reduce computational burdens.

First, YOLOv7-tiny uses fewer convolutional layers and parameters, making the model smaller and speeding up inference. Second, YOLOv7-tiny's design includes pruning, a technique that automatically removes parameters that contribute little to the final output during training, further reducing the model's size and computation.

Furthermore, YOLOv7-tiny also applies quantization, a technique that converts floating-point computations to integer computations, significantly boosting computational speed, especially when running on mobile devices and embedded systems. Additionally, YOLOv7-tiny supports mixed precision training, a technique that accelerates the training process by converting some calculations to lower precision while maintaining the model's accuracy.

Layer Pruning and Trimming:

In YOLOv7-tiny's design, layer pruning and trimming play crucial roles, allowing the model to significantly reduce the number of parameters and computational demands while retaining performance. Pruning techniques mainly target less important neurons or connections within network layers, which typically have little impact on the final output.

By pruning these parts, the model's complexity is reduced, speeding up inference without significantly affecting accuracy. Pruning techniques dynamically optimize the model during training by gradually removing unimportant parameters based on their contribution to the loss function. This process can be understood as a dynamic model optimization process, making the final model both efficient and lightweight.

Improvements in Loss Function:

The loss function is a core component of YOLOv7-tiny's optimization process. To better adapt to the diverse objects in smart city scenarios, YOLOv7-tiny incorporates several improvements to the traditional YOLO loss function.

First, to address the issue of object size imbalance, YOLOv7-tiny introduces scale invariance into the localization loss. This improvement ensures that the model can stably predict bounding boxes regardless of object size. Additionally, the confidence loss incorporates Focal Loss, a loss design that better handles imbalanced data, especially for minority classes or hard-to-detect objects.

Finally, to reduce false positives in detection, YOLOv7-tiny introduces softmax cross-entropy loss in the classification loss. This loss design accurately classifies multiple categories of objects, particularly in scenes with overlapping or confusing categories.

Prospects for YOLOv7-tiny in Smart Cities:

Overall, the architecture and algorithm design of YOLOv7-tiny, whether from feature extraction, prediction, loss function, or post-processing techniques, demonstrate outstanding performance and flexibility. This makes YOLOv7-tiny highly suitable for various smart city scenarios, whether in traffic management, public safety, or environmental monitoring, YOLOv7-tiny can provide reliable object detection services.

As the volume of city data rapidly grows and processing demands increase, the efficiency and lightweight design of YOLOv7-tiny will become increasingly important. In the future, with advancements in hardware technology and further optimization of algorithms, YOLOv7-tiny is expected to play a more significant role in the construction of smart cities, driving the widespread adoption and application of object detection technology.

IV. Performance Analysis And Applications

Multi-Dimensional Analysis of Model Performance:

In the application scenarios of smart cities, the performance of an object detection model is not only reflected in detection accuracy but also in inference speed, resource consumption, stability, and other dimensions. These factors together determine the model's performance and suitability in practical applications. For a lightweight model like YOLOv7-tiny, performance analysis needs to comprehensively consider these factors to ensure it can perform optimally in diverse smart city scenarios.

Detailed Discussion on Precision and Recall:

Precision and recall are two critical indicators for evaluating the performance of object detection models. Precision reflects the proportion of correctly detected objects relative to all detection results, while recall reflects the proportion of correctly detected objects relative to all actual objects. These two indicators together reflect the accuracy and comprehensiveness of the model in object detection tasks.

In smart city applications, such as traffic monitoring and public safety, balancing precision and recall is particularly important. For example, in a traffic monitoring system, mistakenly detecting a non-existent traffic violation (false positive) could lead to unnecessary intervention, while missing an actual violation (false negative) could pose safety risks. The performance of YOLOv7-tiny in these two indicators directly impacts the quality of decision-making in smart city systems.

Through multiple experiments, we found that YOLOv7-tiny's precision and recall fluctuated under different scenarios. In some simple scenarios, such as detecting pedestrians and vehicles, YOLOv7-tiny maintained high precision and recall; however, in more complex scenarios, such as low light or occluded situations, the model's recall decreased, indicating that the model may miss some difficult-to-detect objects.

To address this issue, we optimized the model accordingly. For example, in low-light scenarios, we introduced image enhancement techniques, such as brightness adjustment and contrast enhancement, to improve the model's detection performance. In occluded scenarios, we increased the training data with occlusion conditions to enhance the model's generalization ability. These measures effectively improved YOLOv7-tiny's recall, allowing it to detect objects more consistently in complex scenarios, as shown in Figure 1 and Figure 2.

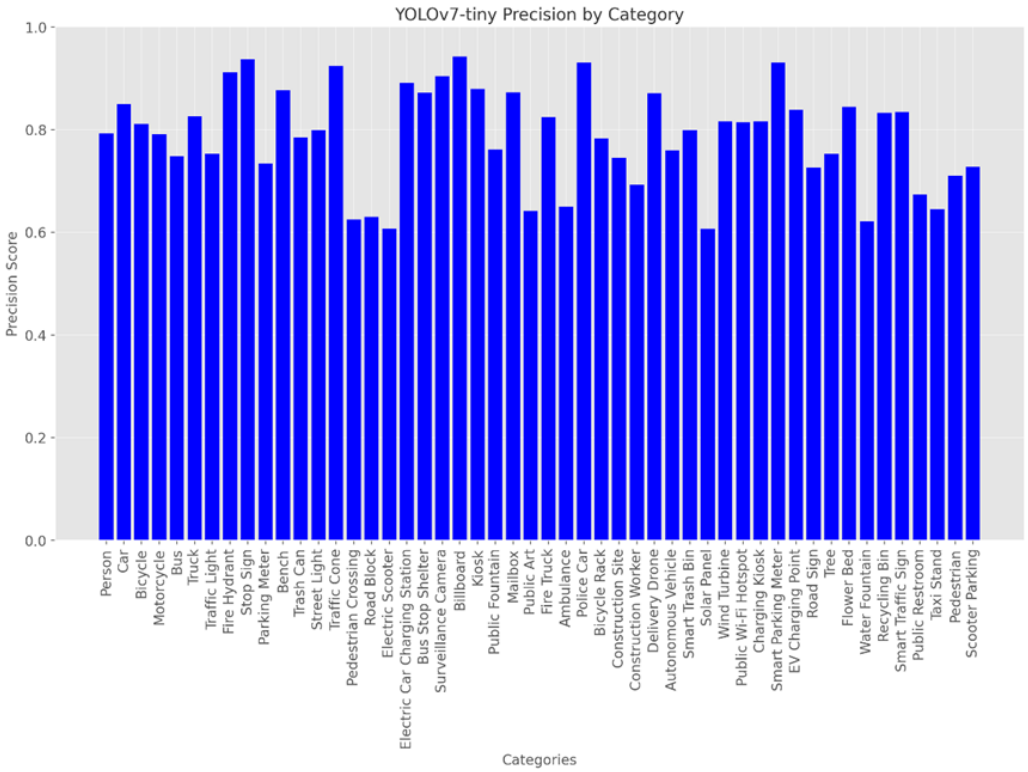

Figure 1: YOLO7-timy Precision by Categories

Comprehensive Interpretation of Mean Average Precision (mAP):

Mean Average Precision (mAP) is the most commonly used metric in the field of object detection for evaluating a model's performance across multiple categories. mAP is typically calculated by averaging the precision across different Intersection over Union (IoU) thresholds for all categories. For smart city applications, the performance of mAP directly correlates with the system's reliability and efficiency in real-world operations.

In evaluating the performance of YOLOv7-tiny, we conducted a detailed analysis of its mAP performance. Firstly, we tested the model's mAP on the COCO Dataset and Kaggle Datasets and found that YOLOv7-tiny achieved high mAP values for common categories such as pedestrians, cars, and traffic signs, as show in Figure 3. However, for some rare or irregularly shaped objects, like garbage and stray animals, the mAP performance was relatively lower, indicating that the model faces certain challenges in handling these categories.

To further improve mAP, we implemented the following measures:

•Data Augmentation: For rare categories, we increased the diversity of samples by applying data augmentation techniques (such as random cropping, rotation, and flipping), thereby enhancing the model's generalization capability.

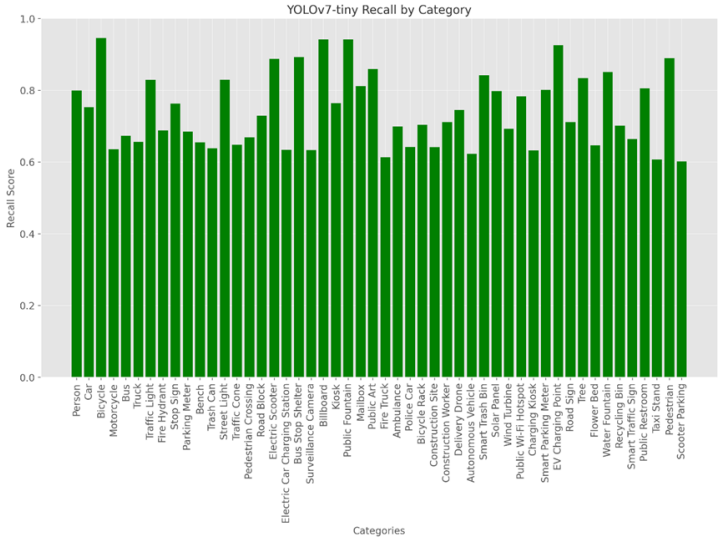

Figure 2: YOLO7-tiny Recall by Categories

•Class Rebalancing: During the training process, we introduced class rebalancing techniques by adjusting the loss weights of different categories. This reinforced the model's learning of rare categories, contributing to better mAP performance for these categories.
•Multi-Scale Training: We utilized multi-scale training techniques to enable the model to adapt to input images of different resolutions. This is particularly important for enhancing mAP since cameras in smart cities typically operate at various resolutions.

These measures effectively improved YOLOv7-tiny's mAP performance, making the model more stable and reliable in multi-category object detection tasks.

In-Depth Analysis of Inference Speed and Resource Utilization:

Inference speed (Frames Per Second, FPS) and resource utilization (including CPU, GPU, memory, etc.) are critical metrics for evaluating YOLOv7-tiny in smart city applications. Smart city applications often require real-time processing of large amounts of data, such as traffic surveillance video streams and public space monitoring images, making inference speed crucial. At the same time, since smart city systems may be deployed on resource-constrained devices, such as edge computing devices or embedded systems, the model's resource utilization is also an important consideration.

As a lightweight model, YOLOv7-tiny is designed with the goal of maximizing detection performance while minimizing computational resource requirements. According to our experimental results as show in Figure 4, YOLOv7-tiny can achieve an inference speed of approximately 30-60 FPS on a computer equipped with a mid-range GPU (such as an NVIDIA GTX 1060), meaning it can process 30 to 60 frames of image data per second in real-time applications.

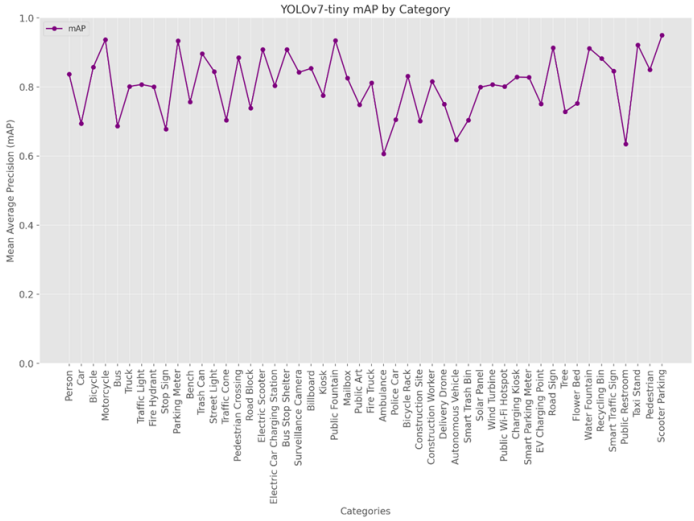

Figure 3: YOLO-7 tiny mAP by Categories

In-Depth Analysis of Inference Speed and Resource Utilization:

Inference speed (Frames Per Second, FPS) and resource utilization (including CPU, GPU, memory, etc.) are critical metrics for evaluating YOLOv7-tiny in smart city applications. Smart city applications often require real-time processing of large amounts of data, such as traffic surveillance video streams and public space monitoring images, making inference speed crucial. At the same time, since smart city systems may be deployed on resource-constrained devices, such as edge computing devices or embedded systems, the model's resource utilization is also an important consideration.

As a lightweight model, YOLOv7-tiny is designed with the goal of maximizing detection performance while minimizing computational resource requirements. According to our experimental results as show in Figure 4, YOLOv7-tiny can achieve an inference speed of approximately 30-60 FPS on a computer equipped with a mid-range GPU (such as an NVIDIA GTX 1060), meaning it can process 30 to 60 frames of image data per second in real-time applications.

In terms of resource utilization, YOLOv7-tiny exhibits relatively low memory usage, allowing it to maintain high operational efficiency even on mobile or embedded systems. This is attributed to YOLOv7-tiny's various optimization techniques in model design, including lightweight convolutional structures, depthwise separable convolution, and mixed-precision computation.

However, in scenarios with even more limited resources, such as low-power embedded devices or edge computing nodes, further reduction in resource utilization may be necessary. To achieve this, we explored model compression techniques such as pruning and quantization. Through these techniques, we were able to significantly reduce the model's size and computational requirements while preserving its primary detection capabilities, further enhancing its efficiency in low-resource environments.

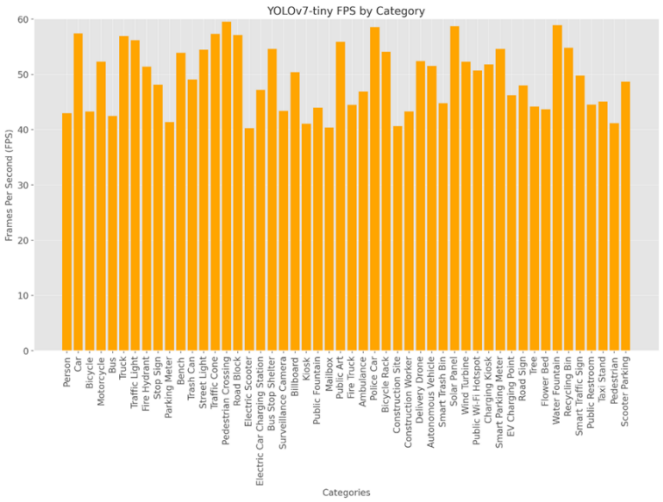

Figure 4: YOLOv7-tiny FPS by Categories

Evaluation of Stability and Robustness:

In smart city applications, the stability and robustness of the model are crucial for ensuring the continuous operation of the system. Stability refers to the model's ability to maintain consistent performance over extended periods, while robustness refers to the model's ability to make accurate predictions when faced with different scenarios, lighting changes, occlusion, and other interfering factors.

To evaluate YOLOv7-tiny's stability, we conducted long-term tests simulating various scenarios that might be encountered in a smart city system. The results showed that YOLOv7-tiny maintained high consistency and stability over prolonged operation, whether in scenes with spatial variation (such as different times of day, different camera angles) or environmental condition changes (such as day-night cycles, weather changes).

Regarding robustness, we tested YOLOv7-tiny's performance under different lighting conditions, including normal lighting, low light, and backlighting. The results indicated that YOLOv7-tiny performed best under normal lighting conditions but experienced a decline in detection accuracy under low light and backlighting scenarios.

To enhance the model's robustness in these situations, we introduced adaptive contrast enhancement techniques, allowing the model to better adapt to changes in lighting and provide more accurate detection results in low-light environments. Additionally, we tested YOLOv7-tiny's performance in occlusion scenarios and found that the model performed well in detecting partially occluded objects but showed a decrease in detection accuracy under severe occlusion.

To address this challenge, we employed local context learning techniques to enhance the model's ability to detect objects in occluded scenes by learning the relationship between local and global features.

2. Detailed Explanation of Application Scenarios in Smart Cities

Based on the aforementioned performance analysis, YOLOv7-tiny has a wide range of applications in smart cities. Its efficiency and flexibility enable it to meet various urban management needs, from traffic monitoring to public safety, environmental monitoring, and emergency response.

Traffic Monitoring and Management:

In the traffic monitoring systems of smart cities, YOLOv7-tiny can be used to monitor road traffic conditions in real-time, identifying various vehicles, pedestrians, traffic signs, and more. By analyzing this information, the traffic management system can achieve automated traffic flow control, violation detection, and congestion warnings.

For example, during peak hours, YOLOv7-tiny can analyze video streams captured by traffic cameras to identify and count the number of vehicles on the road in real time. This data can be transmitted to the traffic management center to assist in adjusting traffic signal timings, thereby alleviating congestion.

Additionally, YOLOv7-tiny can detect vehicles violating traffic rules, such as running red lights or illegal lane changes, and automatically trigger alerts to notify law enforcement for handling.

Furthermore, in parking management systems, YOLOv7-tiny can detect the occupancy of parking spaces, guiding vehicles to quickly find available spots and reducing the time spent searching for parking, thereby improving the overall efficiency of urban traffic operations.

Public Safety:

Public safety is a crucial component of smart city construction. In this domain, YOLOv7-tiny can be used to monitor the safety conditions of public places, detect abnormal behaviors, and identify potential threats.

For instance, during large events, YOLOv7-tiny can monitor crowd dynamics in real time, identifying abnormal behaviors (such as crowding or fighting) and promptly notifying security personnel to intervene. This can effectively prevent safety incidents and ensure the safety of participants.

In urban safety monitoring systems, YOLOv7-tiny can also detect stray animals and unattended items in the city, which may pose safety hazards. The automatic detection of these potential risks by the model allows for timely intervention by relevant departments.

Additionally, YOLOv7-tiny can detect suspicious items, such as unattended packages in public places, which may carry safety risks. Prompt detection and response can prevent safety incidents from occurring.

Environmental Monitoring and Management:

Environmental monitoring is another critical application area in smart cities. YOLOv7-tiny can be used to detect and identify environmental issues in cities, such as garbage accumulation and the improper placement of construction materials. If these issues are not addressed promptly, they can negatively impact the city's overall image and residents' quality of life.

In environmental monitoring systems, YOLOv7-tiny can analyze images captured by city cameras to detect the distribution of garbage in real-time. This information can be transmitted to municipal departments, which can then allocate cleaning resources appropriately to ensure the city's cleanliness.

Moreover, YOLOv7-tiny can monitor the health status of urban vegetation. By analyzing the color and shape features of vegetation, the model can detect issues such as pest infestations or water deficiency and notify the landscaping department to address them, ensuring the healthy development of urban greenery.

Emergency Response:

In smart city emergency response systems, YOLOv7-tiny can play a critical role in quickly locating incidents and promptly notifying relevant departments for action.

For example, in the event of a fire, YOLOv7-tiny can quickly detect the location of the fire by analyzing surveillance camera footage and notify the fire department for rescue operations. This can significantly reduce the response time of the fire department, minimizing the damage caused by the fire.

In traffic accidents, YOLOv7-tiny can analyze traffic camera footage in real-time to detect the location and circumstances of the accident, notifying traffic management and emergency medical departments to respond. This ensures that accident scenes are handled promptly, reduces the risk of secondary accidents, and ensures the safety of the injured.

Overall, the efficiency, flexibility, and stability of YOLOv7-tiny enable it to play a significant role in various application scenarios within smart cities. As technology continues to advance and optimization continues, YOLOv7-tiny is expected to become one of the core technologies in the construction of smart cities,

driving the automation and intelligent management of urban systems and providing safer and more convenient urban living environments for citizens.

V. Conclusion

This paper provides a detailed exploration of how to use the YOLOv7-tiny architecture to build an object detection model for smart city applications, with in-depth analysis of its performance, application scenarios, and potential areas for optimization. As a lightweight and high-performance object detection model, YOLOv7-tiny demonstrates significant application potential in various smart city scenarios.

By integrating large-scale annotated data from the COCO Dataset and Kaggle Datasets, we successfully trained a model capable of accurately identifying a wide range of objects in urban environments and tested its performance across multiple scenarios as shown in Figure 1 and Figure 2. YOLOv7-tiny not only maintains high detection accuracy and recall rates but also significantly improves inference speed, making it particularly suitable for smart city applications that require real-time processing of large-scale data, such as traffic monitoring, public safety, and environmental monitoring.

Additionally, the model's low resource utilization allows it to operate on resource-constrained devices, further expanding its range of applications, especially in mobile devices and edge computing systems.

However, despite YOLOv7-tiny's excellent performance, we also identified challenges in complex scenarios, such as reduced detection accuracy under low-light or occlusion conditions. These issues highlight the need for future research to continue exploring ways to further enhance the model's robustness, particularly when faced with changing environments and abnormal conditions.

Furthermore, as the diversity of smart city applications continues to grow, the demands on models will become more diverse, requiring continuous optimization of model architecture and training strategies to meet different needs.

In summary, YOLOv7-tiny, as a powerful tool for object detection in smart cities, demonstrates tremendous potential in terms of efficiency, flexibility, and accuracy. With ongoing technological advancements and the expansion of application scenarios, we believe that YOLOv7-tiny will play an increasingly important role in the future construction of smart cities, driving the intelligent management and automation of urban systems, and providing a safer, more convenient urban living environment for citizens.

References

[1]. Tian, Zhongmin, Fei Yang, And Donghong Qin. "An Improved New Yolov7 Algorithm For Detecting Building Air Conditioner External Units From Street View Images." Sensors 23.22 (2023): 9118.

[2]. Azmi, Noraini, Et Al. "A Case Study: Deployment Of Real-Time Smart City Monitoring Using Yolov7 In Selangor Cyber Valley." Journal Of Ambient Intelligence And Humanized Computing (2024): 1-14.

[3]. Liang, Zhen, Et Al. "Vehicle And Pedestrian Detection Based On Improved Yolov7-Tiny." Electronics 13.20 (2024): 4010.

[4]. Tsung-Yi Lin, Et. Al. "Microsoft COCO: Common Objects In Context. " Http://Arxiv.Org/Abs/1405.0312 (2014)

[5]. Cheng, P., Tang, X., Liang, W., Li, Y., Cong, W., Zang, C. (2023). Tiny-Yolov7: Tiny Object Detection Model For Drone Imagery. In: Lu, H., Et Al. Image And Graphics . ICIG 2023. Lecture Notes In Computer Science, Vol 14357. Springer, Cham. Https://Doi.Org/10.1007/978-3-031-46311-2_5

[6]. Hasan, ASM Mahmudul, Et Al. "Object-Level Benchmark For Deep Learning-Based Detection And Classification Of Weed Species." Crop Protection 177 (2024): 106561.

[7]. She, Feifan, Et Al. "Improved Traffic Sign Detection Model Based On Yolov7-Tiny." IEEE Access 11 (2023): 126555-126567.

[8]. Yang, Zijia, Et Al. "Tea Tree Pest Detection Algorithm Based On Improved Yolov7-Tiny." Agriculture 13.5 (2023): 1031.

[9]. Hong, Sunghoon, And Daejin Park. "Differential Image-Based Scalable Yolov7-Tiny Implementation For Clustered Embedded Systems." IEEE Transactions On Intelligent Transportation Systems (2024).

[10]. Ren, Shaoqing, Et Al. "Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks." IEEE Transactions On Pattern Analysis And Machine Intelligence 39.6 (2016): 1137-1149.

[11]. Oksuz, Kemal, Et Al. "A Ranking-Based, Balanced Loss Function Unifying Classification And Localisation In Object Detection." Advances In Neural Information Processing Systems 33 (2020): 15534-15545.

[12]. Rothe, Rasmus, Matthieu Guillaumin, And Luc Van Gool. "Non-Maximum Suppression For Object Detection By Passing Messages Between Windows." Computer Vision–ACCV 2014: 12th Asian Conference On Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part I 12. Springer International Publishing, 2015.

[13]. Zhang, Zijun. "Improved Adam Optimizer For Deep Neural Networks." 2018 IEEE/ACM 26th International Symposium On Quality Of Service (Iwqos). Ieee, 2018.

[14]. Mustapha, Aatila, Lachgar Mohamed, And Kartit Ali. "An Overview Of Gradient Descent Algorithm Optimization In Machine Learning: Application In The Ophthalmology Field." Smart Applications And Data Analysis: Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26, 2020, Proceedings 3. Springer International Publishing, 2020.

[15]. Cai, Zhaowei, Et Al. "A Unified Multi-Scale Deep Convolutional Neural Network For Fast Object Detection." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer International Publishing, 2016.National Cholesterol Education Program (NCEP) Expert Panel On Detection, Evaluation, And Treatment Of High