Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.047901

ARTICLE

YOLOv5ST: A Lightweight and Fast Scene Text Detector

Yiwei Liu[1], Yingnan Zhao[1,*], Yi Chen[1], Zheng Hu[1] and Min Xia[2]

[1]School of Computer and Science, Nanjing University of Information Science and Technology, Nanjing, 210044, China

[2]School of Automation, Nanjing University of Information Science and Technology, Nanjing, 210044, China

*Corresponding Author: Yingnan Zhao. Email: zh_yingnan@126.com

Received: 21 November 2023 Accepted: 20 February 2024 Published: 25 April 2024

ABSTRACT

Scene text detection is an important task in computer vision. In this paper, we present YOLOv5 Scene Text (YOLOv5ST), an optimized architecture based on YOLOv5 v6.0 tailored for fast scene text detection. Our primary goal is to enhance inference speed without sacrificing significant detection accuracy, thereby enabling robust performance on resource-constrained devices like drones, closed-circuit television cameras, and other embedded systems. To achieve this, we propose key modifications to the network architecture to lighten the original backbone and improve feature aggregation, including replacing standard convolution with depth-wise convolution, adopting the C2 sequence module in place of C3, employing Spatial Pyramid Pooling Global (SPPG) instead of Spatial Pyramid Pooling Fast (SPPF) and integrating Bi-directional Feature Pyramid Network (BiFPN) into the neck. Experimental results demonstrate a remarkable 26% improvement in inference speed compared to the baseline, with only marginal reductions of 1.6% and 4.2% in mean average precision (mAP) at the intersection over union (IoU) thresholds of 0.5 and 0.5:0.95, respectively. Our work represents a significant advancement in scene text detection, striking a balance between speed and accuracy, making it well-suited for performance-constrained environments.

KEYWORDS

Scene text detection; YOLOv5; lightweight; object detection

1 Introduction

Scene text detection is an important application of object detection in computer vision. Traditionally, scene text detection methods tend to be characterized in terms of the text itself, such as considering the text as connected components [1], or considering the sequential orientation of the text [2]. However, due to the complex and diverse morphology of scene text, it is difficult to accurately design a detection method on its characteristics. Therefore, it is more efficient to consider it as a general objection detection task. In recent years, with the advancements in deep learning, significant progress has been made in this field. Particularly, the advent of one-stage high-performance object detection models like the you only look once (YOLO) model [3] has enabled lightweight, real-time, and efficient scene text recognition networks.

Recently, numerous lightweight enhancement methods have been developed for YOLO, featuring low parameter counts and floating-point operations (FLOPs), examples of which include MobileNet [4], ShuffleNet [5], and GhostNet [6]. However, experiments show that lower parameter numbers and FLOPs do not always lead to a higher inference speed [7]. Therefore, our main motivation is to enable fast and practical scene text detection, and our work emphasizes the practical application of lightweight scene text detection models, making inference speed the focal point.

In this paper, we take the recent v6.0 version of YOLOv5 [8] as the basis for lightweight improvement. YOLOv5 series has various models with different sizes, the smallest is YOLOv5s, which offers a small model size and fast inference speed. It is suitable for scenarios where computational resources are limited. Therefore, we choose YOLOv5s for optimization, aiming at improving the inference speed and keeping an acceptable detection accuracy to make the model more friendly to low-performance devices. Our work mainly aims to reduce the computational cost by simplifying the network architecture and implementing lightweight feature aggregation methods to keep accuracy without increasing too much computational burden.

In our proposed YOLOv5ST (YOLOv5 Scene Text) model, the main contributions are as follows:

- Introducing depth-wise convolution to reduce computational complexity and cost, and removing unnecessary parts of the network to achieve faster inference speed.
- Improving the feature aggregation in the network to keep the detection accuracy as much as possible.
- Training the model with the SynthText scene text image dataset [9], and conducting detection experiments on the RoadText scene text video dataset [10].

The experimental results demonstrate the model's ability to attain high inference speed while maintaining a relatively high level of accuracy. Additionally, we employ the Convolutional Recurrent Neural Network (CRNN) [11] scene text recognition model to realize the comprehensive functionality spanning efficient scene text detection through recognition.

2 Related Work

Traditional scene text detection methods are usually based on connected component analysis [1]. This approach employs various properties, including color, edge, stylus (density calculation and pixel gradient), and texture, to determine and examine the candidate areas of the text. Candidate components are first detected by color clustering, edge clipping, or extra area extraction (text candidate extraction). Then, non-text components are filtered through manual design rules or automated train classifications (candidate correction by deleting non-candidate Text) [12]. For example, Stroke width transform (SWT) [13] is a feature-extracting method that examines the density and thickness of characters to determine the edges. Stroke feature transform (SFT) [14] extends SWT to enhance performance on component separation and connection. Maximally stable extremal region (MSER) [15] detects stable regions in an image by analyzing their stability under varying thresholds and then extracting connected components. Those methods are efficient because number of connected components are usually small. However, clutter and thickening of texts will change the connected components, which reduces the accuracy and recall. To overcome this problem, the sliding window method [16] detects texts by shifting a window to all locations at several different scales to achieve a high recall; however, it has a high computational cost due to scanning all windows [12].

In recent years, neural networks, especially Convolutional Neural Network (CNN)-based networks, have shown great power in object detection. Therefore, it is feasible to consider scene text detection as a general object detection task. Object detection methods can be categorized into two-stage and one-stage methods. Two-stage methods divide the object detection task into two stages, where the first stage usually generates candidate object regions, and the second classifies and validates the candidate regions. Although two-stage methods enhance accuracy and robustness with the cooperation of two stages, they usually require more computational resources and time. On the other hand, one-stage methods detect objects directly from the input image, which are generally simpler and more efficient, but may have lower detection accuracy and recall in complex scenes.

Two-staged Regions with CNN features (R-CNN) series are typical early object detection models based on neural networks. R-CNN [17] adopts selective search to select anchor boxes (candidate regions), then extracts features for each anchor box using a CNN network and applies support vector machines (SVM) to classify the objects. Fast R-CNN [18] directly extracts the feature of the whole image, improving the efficiency. Faster R-CNN [19] introduces a region proposal network (RPN) to generate anchor boxes, predict object bounds and objectness scores at each position, and lead to high-quality region proposals. Mask R-CNN [20] proposes a fully convolutional network (FCN) for precise pixel-to-pixel prediction manner, enabling finer alignment and more accurate segmentation. There are also CNN-based methods designed for scene text detection [12], such as TextBoxes [2] and Rotational Region CNN (R2CNN) [21]. To detect lined text, TextBoxes designs a series of text-box layers that utilize convolution and pooling operations on multi-scale feature maps to predict text bounding boxes. R2CNN generates axis-aligned bounding boxes for text using RPN. For each axis-aligned text box proposed by RPN, R2CNN extracts its pooled features using different pooling sizes and employs simultaneous prediction of text/non-text scores, axis-aligned boxes, and tilted minimum area boxes. Finally, the detection candidates are post-processed using titled non-maximal suppression to obtain the final detection results, achieving better performance with inclined anchor boxes. Nonetheless, these detection methods still have relatively high computational costs and are difficult to achieve at high speed.

Real-time performance is crucial in numerous practical object detection applications, such as autonomous driving and real-time surveillance. Meanwhile, in terms of the trade-off between precision and recall, two-stage methods usually have certain misdetections and omissions in the first stage and need correction by the subsequent processing in the second stage. In contrast, through end-to-end training, one-stage methods offer a better balance between precision and recall, which reduces these issues. Therefore, the one-stage object detection methods have gained more popularity. One typical example is SSD (Single Shot MultiBox Detector) [22], known for its fast speed, high accuracy, and adaptability to various object sizes. It converts the bounding boxes into a set of default boxes with different shapes and scales over different feature map locations, generates scores for each object category in each default box, and adjusts the boxes to better match the object shapes. TextBoxes++ [23] specializes in detecting texts with arbitrary orientations, small sizes, and significantly variant aspect ratios. This specialization is achieved through adaptations on SSD, including the generation of arbitrarily oriented bounding boxes and the incorporation of multi-scale inputs.

As a one-stage object detection model, YOLO is proposed in [3] with good accuracy and fast speed. YOLO divides the input image into grids, with each grid cell responsible for detecting objects within it. For each cell, YOLO predicts whether an object is contained and the bounding box coordinates. It also predicts the class probability of the object for each bounding box. In contrast to traditional two-stage methods, YOLO makes direct predictions on the whole image, greatly reducing the computational cost. Additionally, YOLO benefits from global contextual information, improving

detection accuracy. Over time, the YOLO model is continuously being optimized and improved. YOLOv2 [24] introduces anchor boxes to increase recall and employs the Darknet-19 network to improve accuracy, leading to faster speed and better performance. YOLOv3 [25] extracts features from three different scales using a concept similar to feature pyramid networks (FPN) [26], enabling more accurate detection of objects of various sizes, and replacing the feature extraction network with Darknet-53 for better accuracy. YOLOv4 [27] combines numerous advanced features including Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT), Mish activation, Mosaic data augmentation, DropBlock regularization, and CIoU loss. For example, a CSPDarknet [28] backbone is utilized to achieve a larger receptive field and better multi-scale detection capability. YOLOv5 [8] is more deeply optimized in activation functions and optimizers and shows better performance. In the recent version YOLOv5 v6.0, BottleneckCSP [28] modules have been replaced by C3 [14], the Focus module in the backbone has been replaced by a Convolution-BatchNormalization-Sigmoid (CBS) block [29], and Spatial Pyramid Pooling (SPP) [30] has been replaced by SPPF [8]. These modifications aim at enhancing detection speed and accuracy, making YOLO more suitable for real-time object detection, including scene text detection.

In recent research, most of the recent approaches are driven by deep network models [31]. For example, Fast convergence speed and Accurate text localization (FANet) [32] is a scene text detection network that achieves fast convergence speed and accurate text localization. It combines the transformer feature learning and normalized Fourier descriptor modeling. Despite this, we still use YOLO in this paper because it is easy to use in real applications and can be extended to other object detection tasks.

3 Methodology

3.1 Network Architecture

The overall network of YOLOv5ST is based on YOLOv5s, a small version in the YOLOv5 v6.0 series. As depicted in Fig. 1 (modifications are in red), with the main object of increasing detection speed while maintaining detection accuracy as much as possible, four key modifications are made to the original YOLOv5 network architecture:

(1) Replacing standard convolution with depth-wise convolution. This substitution significantly reduces computational complexity by decreasing the count of convolution kernels.

(2) Adopting the C2 sequence module in place of C3. This streamlined module involves fewer convolutional operations, incorporating depth-wise convolution, thereby amplifying detection speed.

(3) Employing SPPG instead of SPPF. SPPG is purposefully designed to fuse global features from varying scales. Consequently, the backbone's output incorporates global information for subsequent stages in achieving higher accuracy.

(4) Integrates BiFPN into the neck. This integration facilitates the aggregation of a more extensive array of features within the feature pyramid, ultimately contributing to a heightened level of accuracy.

The modification (1) and (4) are existing approaches, and modifications (2) and (3) are our contributions in this research work. Meanwhile, we integrate these modifications into the specific network architecture. Overall, the network architecture of our proposed YOLOv5ST model is depicted in Fig. 1.

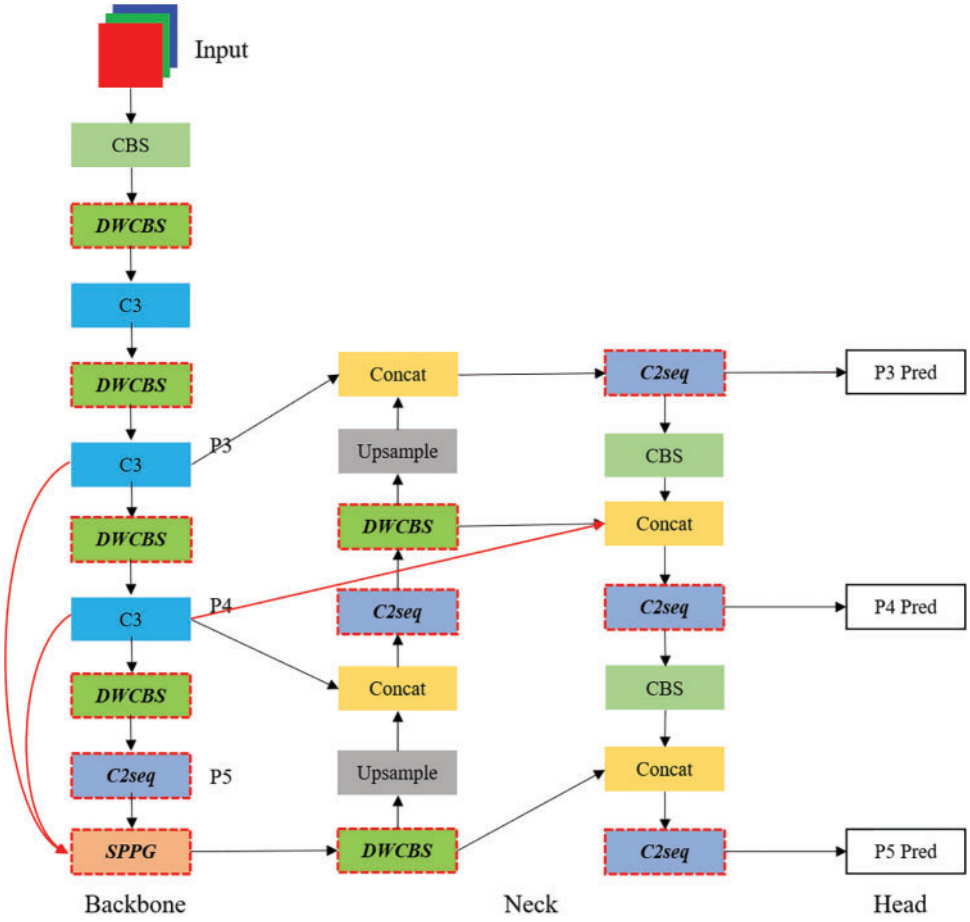

Figure 1: Overall architecture of YOLOv5ST model

3.2 Depth-Wise Convolution

DWConv (Depthwise convolution) is used in our model to reduce computational cost in standard convolutions.

In [33], the proposed convolutional operation, also known as DSC (depth-wise separable convolution), consists of two parts, depth-wise convolution and point-wise convolution. In depth-wise convolution, the convolutional filters are divided into N groups that are equal to the number of input channels, and point-wise convolution is a 1×1 convolution to connect previous output to output channels.

In a convolution operation, we denote the input feature map size as $D_F \times D_F$, kernel size as $D_k \times D_k$, number of input channels as C_{in}, and the number of output channels as C_{out}. For a standard convolution, the computation cost is

$$\boldsymbol{Cost}_{\boldsymbol{standardconv}} = \boldsymbol{D}_{\boldsymbol{F}}^{\boldsymbol{2}} \times \boldsymbol{D}_{\boldsymbol{k}}^{\boldsymbol{2}} \times \boldsymbol{C}_{\boldsymbol{in}} \times \boldsymbol{C}_{\boldsymbol{out}} \tag{1}$$

For depth-wise convolution, the computation cost is

$$Cost_{depth-wiseconv} = D_F^2 \times D_k^2 \times C_{in} \tag{2}$$

For point-wise convolution, the computation cost is

$$Cost_{point-wiseconv} = D_F^2 \times C_{in} \times C_{out} \tag{3}$$

Therefore, the overall computation cost of DSC is

$$Cost_{DSC} = D_F^2 \times D_k^2 \times C_{in} + D_F^2 \times C_{in} \times C_{out} \tag{4}$$

Different from the depth-wise separable convolution, we can let N equal the greatest common divisor of C_{in}, and C_{out}. Thus, the computation cost of DWConv is

$$Cost_{DWConv} = D_F^2 \times D_k^2 \times \frac{C_{in}}{\gcd(C_{in}, C_{out})} + D_F^2 \times C_{in} \times C_{out} \tag{5}$$

DWConv can be applied in the CBS (Conv-BatchNorm-SiLU) block which consists of a convolution layer, a batch norm layer, and a SiLU activation function [27] to form a DWCBS (DWConv-BatchNorm-SiLU) block.

3.3 C2 Sequence Module

The C2seq (C2 sequence) module is inspired by C2f [34] to simplify the original C3, these modules are depicted in Fig. 2.

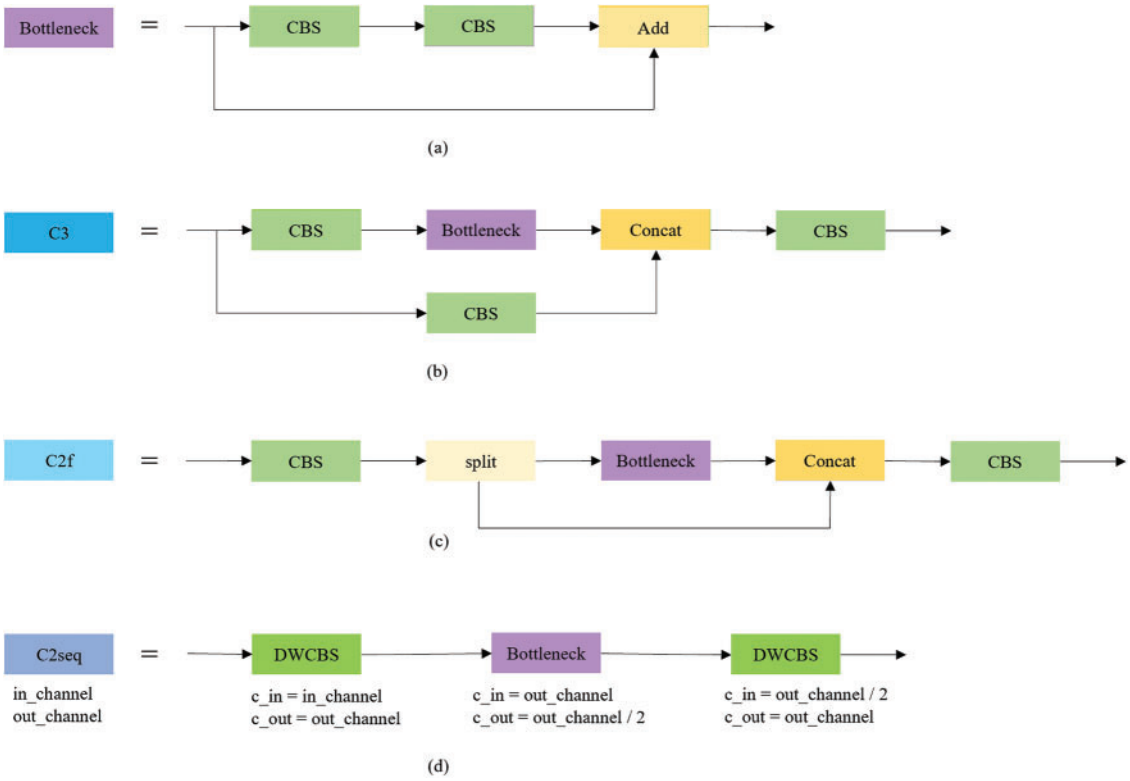

Figure 2: (a) Bottleneck module used in C3; (b) C3 module used in YOLOv5; (c) C2f module used in YOLOv8; (d) Our proposed C2seq module with specific c_in and c_out parameters

As shown in Fig. 2a, Bottleneck in C3 is designed to enhance feature extraction and information flow across different stages or layers of the network. As shown in Fig. 2b, C3 is a CSP block with 3 CBS blocks that let the input pass through into two parallel halves. As shown in Fig. 2c, C2f decreases the number of CBS blocks from 3 to 2 and uses split to do parallel computation. As shown in Fig. 2d, we propose C2seq that further simplifies C2f. Experiments show that split operation in C2f has an impact on the inference speed of the model, so we remove this operation. Then, to overcome the disadvantage of not using residual connection [28], we keep some C3 in our network, as shown in Fig. 1. In addition, we apply DWCBS to further simplify the module. In C2seq, the kernel size in 2 CBS blocks in Bottleneck is set to 3. Since in C2seq, the number of input and output channels of Bottleneck are set to different values, the shortcut in Bottleneck is not used, that is because C2seq modules are mainly used in neck, where original C3 does not use shortcut as well; and at the same time, it can have a higher channel compression ratio inside the Bottleneck to achieve better performance.

3.4 SPPG Module

In the YOLOv5s network, before feature aggregation in the neck, the output feature maps are sent to the SPPF (Spatial Pyramid Pooling Fast) module [8] to increase the receptive field and separate the most important features by pooling multi-scale versions of themselves [29].

However, multi-scale pooling can be applied not only on feature maps in a single scale. In the feature extraction process in the backbone, we have a sequence of feature maps at different scales. By considering those feature maps in one multi-scale pooling module, we can better aggregate global features, which is beneficial for accuracy. Therefore, in our network, we propose the SPPG (Spatial Pyramid Pooling Global) module, which is depicted in Fig. 3. In the PANet [35] of YOLOv5, there are three scales, P3, P4 and P5. The output feature maps of P3, P4, and P5 in the backbone are sent to SPPG simultaneously to fuse global features with different scales to take advantage of multiscale features [36]. For each input feature map, a max pool operation and an average pool operation are implemented simultaneously. Max pooling facilitates the extraction of salient features, while average pooling retains background information. The collaborative use of max pooling and average pooling ensures an accurate and seamless fusion of global features. The kernel size and padding for each pooling operation are set at 3 (in contrast to 5 in Spatial Pyramid Pooling with Fusion [SPPF]), as smaller kernels enhance the detection of smaller objects, thereby improving overall detection performance [29]. The stride values for the two pooling operations on each input feature map are 4, 2, and 1, respectively. Meanwhile, CBS blocks that are used in SPPF are removed for simplification.

3.5 BiFPN Feature Aggregation

Based on the PANet [21] architecture in YOLOv5, there are 3 different scale features (P3, P4 and P5). Multi-scale feature fusion aims to aggregate features at different resolutions [37]. To fuse more features without adding much cost, we can implement BiFPN architecture by adding an extra jump connection edge from the original input node to the output node if they are at the same level, thus reusing the feature in the original input node [37]. Specifically, in YOLOv5, P4 output in the backbone can be concatenated in the top-down process in the neck with a jump connection edge. As shown in Fig. 4, the red edge represents the additional connection in BiFPN.

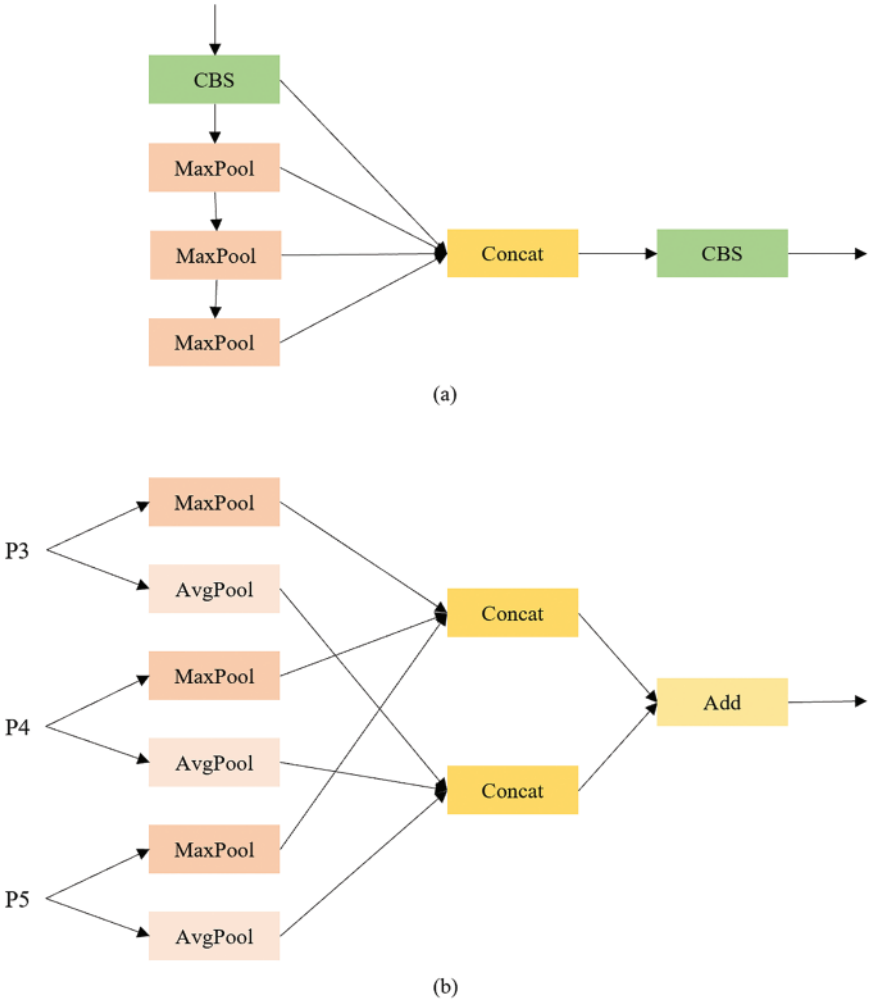

Figure 3: (a) The SPPF module used in YOLOv5; (b) The proposed SPPG module

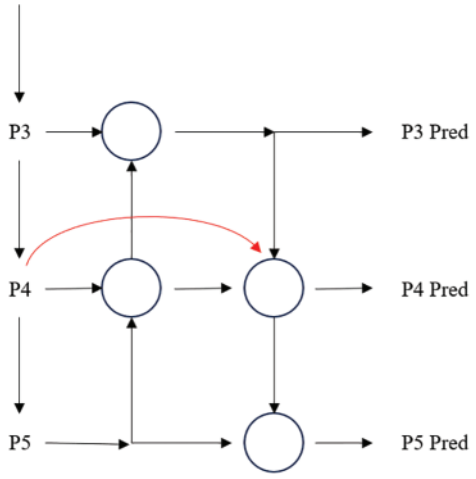

Figure 4: YOLOv5 with BiPFN feature aggregation

4 Experiments

4.1 Data Set

We decide to use a large image dataset to train the model to achieve better precision and a video dataset to test the scene text detection performance. Therefore, in our experiments, SynthText dataset [9] and RoadText dataset [10] are used. The SynthText dataset is an image dataset, and the RoadText dataset is a video dataset. SynthText is used as train data, because as a synthetic dataset, it has notable advantages due to its abundant data volume and precise annotations, making it highly suitable for training. RoadText serves as test data because it includes annotated ground truth information concerning the texts found on the road, which demonstrates a genuine working environment of the model.

SynthText has synthetic text in diverse scene shapes and fonts, having 858,750 images, 7,266,866 word instances, and 28,917,487 characters in total. The images are in various resolutions but mostly 600 × 450 pixels. The synthetic dataset has notable advantages due to its abundant data volume and precise annotations, making it highly suitable for training. Some examples are shown in Fig. 5.

Figure 5: Some examples of SynthText dataset

The RoadText dataset comprises 500 real-world driving videos. Each video is 30 FPS and is approximately 10 s long, featuring a resolution of 1280 × 720 pixels. The dataset includes annotated ground truth information concerning the texts found on the road. It demonstrates a genuine working environment of the model, thus we chose this dataset for testing. Some examples are shown in Fig. 6.

Figure 6: Some examples of RoadText dataset

RoadText dataset has text data in various complex application scenarios, which the model may encounter in real applications, as shown in Fig. 7.

Figure 7: Some examples of RoadText datasets in different scenarios. (a) Light (normal) environment; (b) Dark environment; (c) Raining environment

4.2 Implementation Details

Our detection network implementation is based on YOLOv5s v6.0 and the PyTorch framework. The trains run 11 epochs with a batch size of 64, a learning rate of 0.01, a learning rate momentum of 0.937, and use SGD optimizer with decay of 0.0005. We implement ablation study to verify the validity of each modification, as listed in Table 1. We also implement a series of lightweight detection networks with the same settings, as listed in Table 2 for comparison. These tests are run on an NVIDIA GeForce RTX 2060 GPU.

Table 1: Ablation study results

Modification				Params (M)	FLOPs (G)	mAP@0.5	mAP@0.5: 0.95	Inference FPS
DWConv	C2seq	SPPG	BiFPN					
				7.01	15.8	0.925	0.706	107.53
√				5.29↓	11.8↓	0.924↓	0.707↑	109.89 ↑
	√			5.37↓	13.4↓	0.927↑	0.709↑	120.48↑
		√		7.37↑	15.3↓	0.927↑	0.717↑	102.04↓
			√	7.08↑	16.0↑	0.932↑	0.723↑	105.26↓
√	√			4.04↓	9.6↓	0.917↓	0.685↓	126.58↑
√	√	√		3.38↓	9.1↓	0.911↓	0.677↓	131.41↑
√	√		√	4.04↓	9.6↓	0.917↓	0.686↓	125.00↑
√	√	√	√	4.30↓	9.1↓	0.910↓	0.676↓	**131.58**↑

Table 2: Detection performance comparison

Model	Backbone	Params (M)	FLOPs (G)	mAP@0.5	mAP@0.5: 0.95	Inference FPS
YOLOv5s [2]	YOLOv5 backbone	7.01	15.8	0.925	0.706	107.53
YOLOv5s-GhostNet [23]	GhostNet	5.08	10.5	0.918	0.693	101.01
YOLOv5s-MobileNetV3 [24]	MobileNetV3	1.03	2.0	0.860	0.585	88.50
YOLOv5s-ShuffleNetV2 [25]	ShuffleNetV2	**0.84**	**1.8**	0.821	0.531	123.46
YOLOv5s-CBAM [26]	YOLOv5 backbone	7.03	15.8	0.926	0.706	86.21
YOLOv5s-GSConv [27]	YOLOv5 backbone	6.57	15.2	0.933	**0.724**	98.04
YOLOv5ST	**YOLOv5ST backbone**	4.30	9.1	0.910	0.676	**131.58**

4.3 Ablation Study

In this section, we present the impact of various modifications applied to our proposed model. The experimental environment and datasets remain consistent throughout. The performance of the original YOLOv5s v6.0 model is listed in the first row of Table 1, and models with different modifications are listed in other rows.

For DWConv and C2seq, the number of parameters (params) and Floating Point Operations (FLOPs) decrease significantly, leading to a substantial increase in their inference speed. Particularly noteworthy is the case of C2seq, which exhibits a slight increase in mean Average Precision (mAP). In the case of SPPG and BiFPN, there is a notable increase in the number of parameters (params), FLOPs,

and mAP. However, the drop in inference speed is insignificant. It can be seen that these improvements have had the intended effect.

4.4 Scene Text Detection

To compare the performance of scene text detection, we compare our proposed YOLOv5ST model with other YOLO-based object detection models, the results are listed in Table 2.

From Table 2, it can be seen that among those listed models, our proposed YOLOv5ST model has the best inference speed at 135.58 FPS, which increases by 26.09% compared to YOLOv5. As a relatively small and light model, its number of parameters and FLOPS reduced by 38.66% and 42.41% compared to YOLOv5s, respectively. Although there are other models like GhostNet, MobileNet v3, and ShufflNet v2 that have fewer parameters and FLOPs than ours, they do not achieve an inference speed as fast as ours. At the same time, our proposed YOLOv5ST model keeps a good mAP accuracy, with an acceptable decrease by only 1.63% and 4.19% in mAP@0.5 and mAP@0.5:0.95 compared to YOLOv5s, respectively. In conclusion, our proposed YOLOv5ST model has better overall performance in scene text detection. The performance comparison is also shown in Figs. 8 and 9. Since YOLOv5ST is the closest to the upper left corner of Fig. 9, it has the best overall performance.

In addition, to implement a concrete application of an end-to-end scene text detection and recognition system, YOLOv5ST integrates the CRNN [11] model to recognize the detected text. To demonstrate the joint performance of our scene text detection model and CRNN scene text recognition model in a practical application, some examples are shown in Fig. 10. The models can detect and recognize most of the texts in the dataset, thus a lot of applications can be implemented based on our models to achieve good performance, especially in performance-limited devices like drones, closed-circuit television cameras, and other embedded devices.

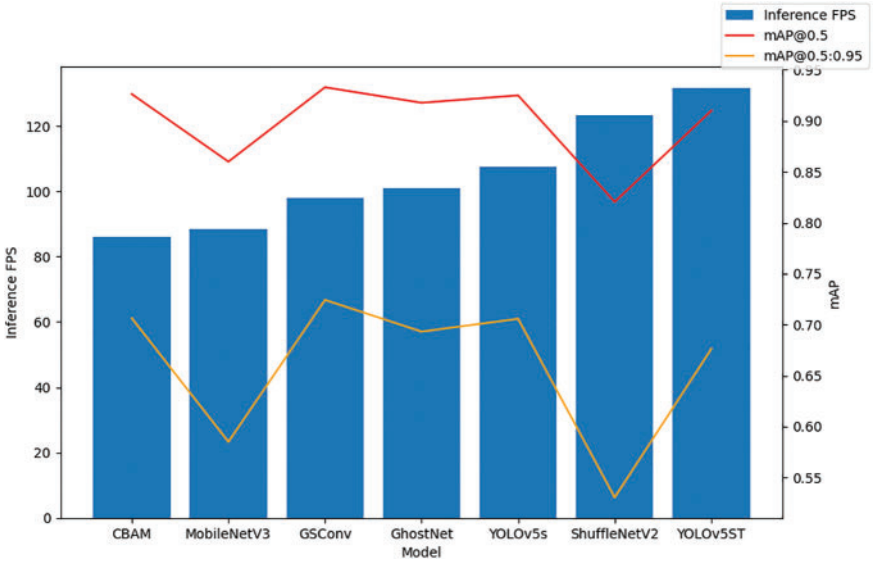

Figure 8: Detection performance comparison

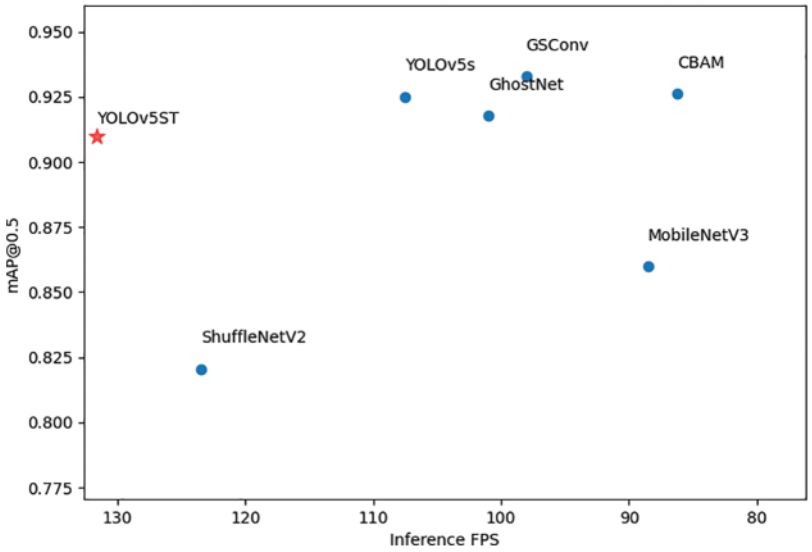

Figure 9: Detection performance comparison

Figure 10: Some examples of scene text detection and recognition

Compared to the original YOLOv5, YOLOv5ST has fewer false detections, as shown in Fig. 11. This is attributed to our model that optimizes the feature extraction and increases the feature aggregation, thus leading to better accuracy.

Also, compared to the original YOLOv5, YOLOv5ST has better performance in handling various complex scenarios. Fig. 7 shows different scenario examples in the RoadText dataset. In those scenarios, precision and recall are shown in Table 3. In most scenarios, YOLOv5ST has better recall and precision than YOLOv5s.

Figure 11: Some examples of YOLOv5 and YOLOv5ST

Table 3: Detection performance comparison

Model		Dark	Light	Raining weather
YOLOv5s	Precise (%)	45.9	55.7	50.6
	Recall (%)	31.3	33.3	45.0
YOLOv5ST	Precise (%)	**62.5**	**66.7**	50.6
	Recall (%)	**35.4**	**39.0**	**45.7**

However, the model still has some false detections, as shown in Fig. 12. For instance, the model occasionally misclassifies crosswalks and other objects that have visual patterns similar to text as textual elements. These misclassification cases show that the complexity of real-world scenes still presents a challenge for the model. To solve this, we need a more precise approach to training that specifically handles complex environments, by discovering deeper intrinsic connections of visual features and using more training data. Furthermore, the current model still has the potential for optimization and refinement through additional training.

Figure 12: Some examples of false detections

5 Conclusion and Future Work

Fast real-time scene text detection is an important task in object detection. In this paper, we propose a lightweight and fast scene text detection model based on the YOLOv5 v6.0 network. Our proposed YOLOv5ST model implements four key modifications on the YOLOv5s to reduce the computational cost, increase inference speed, and keep detection accuracy as much as possible. The proposed model uses DWConv and C2seq to simplify and lightweight the network and uses SPPG and BiFPN to enhance feature aggregation. We conduct experiments to demonstrate the feasibility of these modifications, and compare YOLOv5ST with other YOLO-based lightweight object detection models, the results show that we achieved a significant increase in inference speed of over 26% with less than 5% mAP decrease, thus proving the validity of our model. We also combine the model with a CRNN model to implement an end-to-end scene text detection and recognition system, so that practical applications can be developed based on our models, especially in performance-limited devices like drones, closed-circuit television cameras, and other embedded devices.

In summary, this paper makes improvements to the YOLOv5s network architecture to reduce the burden on the original backbone network improve feature aggregation, and prove the performance by experiments.

We believe our research will produce positive results and influence the object detection field. For a long time, in various lightweight object detection tasks, how to better balance the accuracy and inference speed is the key to research. This research shows a possibility to achieve both fast and accurate text detection, and these optimizations can also be combined with other object detection algorithms such as newer versions of YOLO, or applied to other object detection tasks.

In future work, we will attempt to further enhance both the accuracy and efficiency of our model. One key direction is to consider more data scenarios. For example, considering different text types and occlusion situations, as well as more kinds of weather conditions like foggy and snowy

scenarios, to improve the robustness of the model in complex scenes. Another key direction is to incorporate our approach with the latest versions of YOLO, building lighter models by optimizing the network architecture, fine-tuning hyperparameters, and redesigning the backbone networks or feature extractors to discover better performance of YOLO. Additionally, we will conduct more experiments to investigate more suitable optimization algorithms and loss functions that best suit our specific problem domain.

Acknowledgement: The authors would like to express their gratitude for the valuable feedback and suggestions provided by all the anonymous reviewers and the editorial team.

Funding Statement: This work is supported in part by the National Natural Science Foundation of PR China (42075130) and Nari Technology Co., Ltd. (4561655965).

Author Contributions: Study conceptualization, Yiwei Liu, Yi Chen and Yingnan Zhao; Data collection, Yiwei Liu and Yi Chen; Investigation, Yiwei Liu, Yi Chen, Zheng Hu and Min Xia; Methodology, Yiwei Liu, Yi Chen and Yingnan Zhao; Model implementation and training, Yiwei Liu and Yi Chen; Data analysis, Yiwei Liu and Yi Chen; Manuscript draft, Yiwei Liu and Yingnan Zhao; Manuscript revision, Yiwei Liu and Yingnan Zhao. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The code is available at https://github.com/lyw02/YOLOv5ST.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1480–1500, Jul. 2015. doi: 10.1109/TPAMI.2014.2366765.

[2] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A fast text detector with a single deep neural network," in *Proc. 2017 AAAI Conf. Artif. Intell.*, San Francisco, California, USA, Feb. 4–9, 2017, vol. 31, no. 1. doi: 10.1609/aaai.v31i1.11196.

[3] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, Honolulu, Hawaii, USA, 2017, pp. 7263–7271.

[4] A. Howard *et al.*, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Seoul, Korea, 2019, pp. 1314–1324.

[5] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sept. 8–14, 2018, pp. 116–131.

[6] K. Han, Y. Wang, Q. Tian, J. Guo, C. J. Xu and C. Xu, "Ghostnet: More features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn.*, 2020, pp. 1580–1589.

[7] F. Furlán, E. Rubio, H. Sossa, and V. Ponce, "CNN based detectors on planetary environments: A performance evaluation," *Front Neurorobot.*, vol. 14, pp. 85, 2020. doi: 10.3389/fnbot.2020.590371.

[8] "Yolov5," Accessed: Jun. 25, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[9] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, Las Vegas, Nevada, USA, 2016, pp. 2315–2324.

[10] S. Reddy, M. Mathew, L. Gomez, M. Rusinol, D. Karatzas and C. V. Jawahar, "Roadtext-1k: Text detection & recognition dataset for driving videos," in *2020 IEEE Int. Conf. Robot. Autom. (ICRA)*, Paris, France, 2020, pp. 11074–11080. doi: 10.1109/ICRA40945.2020.9196577.

[11] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Analy. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 1 Nov. 2017. doi: 10.1109/TPAMI.2016.2646371.
[12] F. Naiemi, V. Ghods, and H. Khalesi, "Scene text detection and recognition: A survey," *Multimed. Tools Appl.*, vol. 81, no. 14, pp. 20255–20290, 2022. doi: 10.1007/s11042-022-12693-7.
[13] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn.*, San Francisco, CA, USA, 2010, pp. 2963–2970. doi: 10.1109/CVPR.2010.5540041.
[14] W. Huang, Z. Lin, J. Yang, and J. Wang, "Text localization in natural images using stroke feature transform and text covariance descriptors," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sydney, Australia, 2013, pp. 1241–1248.
[15] L. Neumann and J. Matas, "A method for text localization and recognition in real-world images," in *Comput. Vis.*, Berlin Heidelberg, Springer, 2011, pp. 770–783.
[16] H. Cho, M. Sung, and B. Jun, "Canny text detector: Fast and robust scene text localization algorithm," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, Las Vegas, Nevada, 2016, pp. 3566–3573.
[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, Columbus, Ohio, USA, 2014, pp. 580–587.
[18] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 1440–1448.
[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Adv. Neural Inf. Process. Syst.*, Montreal, Canada, 2015, pp. 28.
[20] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, 2017, pp. 2961–2969.
[21] Y. Jiang *et al.*, "R2CNN: Rotational region CNN for orientation robust scene text detection," 2017. doi: 10.48550/arXiv.1706.09579.
[22] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Lecture Notes in Comput. Sci.*, Cham, Springer, 2016, vol. 9905. doi: 10.1007/978-3-319-46448-0_2.
[23] M. Liao, B. Shi, and X. Bai, "TextBoxes++: A single-shot oriented scene text detector," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3676–3690, Aug. 2018. doi: 10.1109/TIP.2018.2825107.
[24] J. Redmon, and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, Honolulu, Hawaii, USA, 2017, pp. 7263–7271.
[25] J. Redmon, and A. Farhadi, "Yolov3: An incremental improvement," 2018. doi: 10.48550/arXiv.1804.0276.
[26] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, Honolulu, Hawaii, USA, 2017, pp. 2117–2125.
[27] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020. doi: 10.48550/arXiv.2004.10934.
[28] C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn. Works.*, 2020, pp. 390–391.
[29] D. Qi, W. Tan, Q. Yao, and J. Liu, "YOLO5Face: Why reinventing a face detector," in *ECCV 2022, Tel-Aviv. Lecture Notes in Comput. Sci.*, Cham, Springer, 2022, vol. 13805. doi: 10.1007/978-3-031-25072-9_15.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Analy. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sept. 2015. doi: 10.1109/TPAMI.2015.2389824.
[31] T. Khan, R. Sarkar, and A. F. Mollah, "Deep learning approaches to scene text detection: A comprehensive review," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3239–3298, 2021. doi: 10.1007/s10462-020-09930-6.
[32] Y. Zhao, Y. Cai, W. Wu, and W. Wang, "Explore faster localization learning for scene text detection," in *2023 IEEE Int. Conf. Multimed. Expo (ICME)*, Brisbane, Australia, 2023, pp. 156–161. doi: 10.1109/ICME55011.2023.00035.

[33] A. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. doi: 10.48550/arXiv.1704.04861.
[34] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with YOLOv8," 2023. doi: 10.48550/arXiv.2305.09972.
[35] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, Salt Lake City, Utah, USA, 2018, pp. 8759–8768.
[36] R. Yan, L. Peng, S. Xiao, and G. Yao, "Primitive representation learning for scene text recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn.*, 2021, pp. 284–293.
[37] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn.*, 2020, pp. 10781–10790.