YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information

Chien-Yao Wang[1,2], I-Hau Yeh[2], and Hong-Yuan Mark Liao[1,2,3]
[1]Institute of Information Science, Academia Sinica, Taiwan
[2]National Taipei University of Technology, Taiwan
[3]Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan
kinyiu@iis.sinica.edu.tw, ihyeh@emc.com.tw, and liao@iis.sinica.edu.tw

Abstract

Today's deep learning methods focus on how to design the most appropriate objective functions so that the prediction results of the model can be closest to the ground truth. Meanwhile, an appropriate architecture that can facilitate acquisition of enough information for prediction has to be designed. Existing methods ignore a fact that when input data undergoes layer-by-layer feature extraction and spatial transformation, large amount of information will be lost. This paper will delve into the important issues of data loss when data is transmitted through deep networks, namely information bottleneck and reversible functions. We proposed the concept of programmable gradient information (PGI) to cope with the various changes required by deep networks to achieve multiple objectives. PGI can provide complete input information for the target task to calculate objective function, so that reliable gradient information can be obtained to update network weights. In addition, a new lightweight network architecture – Generalized Efficient Layer Aggregation Network (GELAN), based on gradient path planning is designed. GELAN's architecture confirms that PGI has gained superior results on lightweight models. We verified the proposed GELAN and PGI on MS COCO dataset based object detection. The results show that GELAN only uses conventional convolution operators to achieve better parameter utilization than the state-of-the-art methods developed based on depth-wise convolution. PGI can be used for variety of models from lightweight to large. It can be used to obtain complete information, so that train-from-scratch models can achieve better results than state-of-the-art models pre-trained using large datasets, the comparison results are shown in Figure 1. The source codes are at: https://github.com/WongKinYiu/yolov9.

1. Introduction

Deep learning-based models have demonstrated far better performance than past artificial intelligence systems in various fields, such as computer vision, language processing, and speech recognition. In recent years, researchers

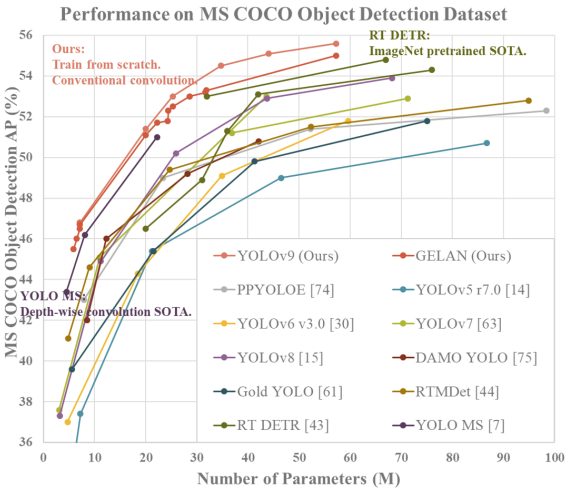
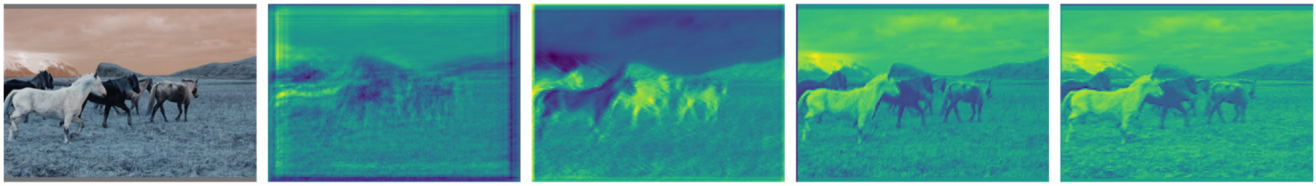


Figure 1. Comparisons of the real-time object detecors on MS COCO dataset. The GELAN and PGI-based object detection method surpassed all previous train-from-scratch methods in terms of object detection performance. In terms of accuracy, the new method outperforms RT DETR [43] pre-trained with a large dataset, and it also outperforms depth-wise convolution-based design YOLO MS [7] in terms of parameters utilization.

in the field of deep learning have mainly focused on how to develop more powerful system architectures and learning methods, such as CNNs [21–23, 42, 55, 71, 72], Transformers [8, 9, 40, 41, 60, 69, 70], Perceivers [26, 26, 32, 52, 56, 81, 81], and Mambas [17, 38, 80]. In addition, some researchers have tried to develop more general objective functions, such as loss function [5, 45, 46, 50, 77, 78], label assignment [10, 12, 33, 67, 79] and auxiliary supervision [18, 20, 24, 28, 29, 51, 54, 68, 76]. The above studies all try to precisely find the mapping between input and target tasks. However, most past approaches have ignored that input data may have a non-negligible amount of information loss during the feedforward process. This loss of information can lead to biased gradient flows, which are subsequently used to update the model. The above problems can result in deep networks to establish incorrect associations between targets and inputs, causing the trained model to produce incorrect predictions.



(a) Input Image **(b) PlainNet** **(c) ResNet** **(d) CSPNet** **(e) GELAN**

Figure 2. Visualization results of random initial weight output feature maps for different network architectures: (a) input image, (b) PlainNet, (c) ResNet, (d) CSPNet, and (e) proposed GELAN. From the figure, we can see that in different architectures, the information provided to the objective function to calculate the loss is lost to varying degrees, and our architecture can retain the most complete information and provide the most reliable gradient information for calculating the objective function.

In deep networks, the phenomenon of input data losing information during the feedforward process is commonly known as information bottleneck [59], and its schematic diagram is as shown in Figure 2. At present, the main methods that can alleviate this phenomenon are as follows: (1) The use of reversible architectures [3, 16, 19]: this method mainly uses repeated input data and maintains the information of the input data in an explicit way; (2) The use of masked modeling [1, 6, 9, 27, 71, 73]: it mainly uses reconstruction loss and adopts an implicit way to maximize the extracted features and retain the input information; and (3) Introduction of the deep supervision concept [28,51,54,68]: it uses shallow features that have not lost too much important information to pre-establish a mapping from features to targets to ensure that important information can be transferred to deeper layers. However, the above methods have different drawbacks in the training process and inference process. For example, a reversible architecture requires additional layers to combine repeatedly fed input data, which will significantly increase the inference cost. In addition, since the input data layer to the output layer cannot have a too deep path, this limitation will make it difficult to model high-order semantic information during the training process. As for masked modeling, its reconstruction loss sometimes conflicts with the target loss. In addition, most mask mechanisms also produce incorrect associations with data. For the deep supervision mechanism, it will produce error accumulation, and if the shallow supervision loses information during the training process, the subsequent layers will not be able to retrieve the required information. The above phenomenon will be more significant on difficult tasks and small models.

To address the above-mentioned issues, we propose a new concept, which is programmable gradient information (PGI). The concept is to generate reliable gradients through auxiliary reversible branch, so that the deep features can still maintain key characteristics for executing target task. The design of auxiliary reversible branch can avoid the semantic loss that may be caused by a traditional deep supervision process that integrates multi-path features. In other words, we are programming gradient information propagation at different semantic levels, and thereby achieving the best training results. The reversible architecture of PGI is built on auxiliary branch, so there is no additional cost. Since PGI can freely select loss function suitable for the target task, it also overcomes the problems encountered by mask modeling. The proposed PGI mechanism can be applied to deep neural networks of various sizes and is more general than the deep supervision mechanism, which is only suitable for very deep neural networks.

In this paper, we also designed generalized ELAN (GELAN) based on ELAN [65], the design of GELAN simultaneously takes into account the number of parameters, computational complexity, accuracy and inference speed. This design allows users to arbitrarily choose appropriate computational blocks for different inference devices. We combined the proposed PGI and GELAN, and then designed a new generation of YOLO series object detection system, which we call YOLOv9. We used the MS COCO dataset to conduct experiments, and the experimental results verified that our proposed YOLOv9 achieved the top performance in all comparisons.

We summarize the contributions of this paper as follows:

1. We theoretically analyzed the existing deep neural network architecture from the perspective of reversible function, and through this process we successfully explained many phenomena that were difficult to explain in the past. We also designed PGI and auxiliary reversible branch based on this analysis and achieved excellent results.
2. The PGI we designed solves the problem that deep supervision can only be used for extremely deep neural network architectures, and therefore allows new lightweight architectures to be truly applied in daily life.
3. The GELAN we designed only uses conventional convolution to achieve a higher parameter usage than the depth-wise convolution design that based on the most advanced technology, while showing great advantages of being light, fast, and accurate.
4. Combining the proposed PGI and GELAN, the object detection performance of the YOLOv9 on MS COCO dataset greatly surpasses the existing real-time object detectors in all aspects.

2. Related work

2.1. Real-time Object Detectors

The current mainstream real-time object detectors are the YOLO series [2, 7, 13–15, 25, 30, 31, 47–49, 61–63, 74, 75], and most of these models use CSPNet [64] or ELAN [65] and their variants as the main computing units. In terms of feature integration, improved PAN [37] or FPN [35] is often used as a tool, and then improved YOLOv3 head [49] or FCOS head [57, 58] is used as prediction head. Recently some real-time object detectors, such as RT DETR [43], which puts its fundation on DETR [4], have also been proposed. However, since it is extremely difficult for DETR series object detector to be applied to new domains without a corresponding domain pre-trained model, the most widely used real-time object detector at present is still YOLO series. This paper chooses YOLOv7 [63], which has been proven effective in a variety of computer vision tasks and various scenarios, as a base to develop the proposed method. We use GELAN to improve the architecture and the training process with the proposed PGI. The above novel approach makes the proposed YOLOv9 the top real-time object detector of the new generation.

2.2. Reversible Architectures

The operation unit of reversible architectures [3, 16, 19] must maintain the characteristics of reversible conversion, so it can be ensured that the output feature map of each layer of operation unit can retain complete original information. Before, RevCol [3] generalizes traditional reversible unit to multiple levels, and in doing so can expand the semantic levels expressed by different layer units. Through a literature review of various neural network architectures, we found that there are many high-performing architectures with varying degree of reversible properties. For example, Res2Net module [11] combines different input partitions with the next partition in a hierarchical manner, and concatenates all converted partitions before passing them backwards. CBNet [34, 39] re-introduces the original input data through composite backbone to obtain complete original information, and obtains different levels of multi-level reversible information through various composition methods. These network architectures generally have excellent parameter utilization, but the extra composite layers cause slow inference speeds. DynamicDet [36] combines CBNet [34] and the high-efficiency real-time object detector YOLOv7 [63] to achieve a very good trade-off among speed, number of parameters, and accuracy. This paper introduces the DynamicDet architecture as the basis for designing reversible branches. In addition, reversible information is further introduced into the proposed PGI. The proposed new architecture does not require additional connections during the inference process, so it can fully retain the advantages of speed, parameter amount, and accuracy.

2.3. Auxiliary Supervision

Deep supervision [28, 54, 68] is the most common auxiliary supervision method, which performs training by inserting additional prediction layers in the middle layers. Especially the application of multi-layer decoders introduced in the transformer-based methods is the most common one. Another common auxiliary supervision method is to utilize the relevant meta information to guide the feature maps produced by the intermediate layers and make them have the properties required by the target tasks [18, 20, 24, 29, 76]. Examples of this type include using segmentation loss or depth loss to enhance the accuracy of object detectors. Recently, there are many reports in the literature [53, 67, 82] that use different label assignment methods to generate different auxiliary supervision mechanisms to speed up the convergence speed of the model and improve the robustness at the same time. However, the auxiliary supervision mechanism is usually only applicable to large models, so when it is applied to lightweight models, it is easy to cause an under parameterization phenomenon, which makes the performance worse. The PGI we proposed designed a way to reprogram multi-level semantic information, and this design allows lightweight models to also benefit from the auxiliary supervision mechanism.

3. Problem Statement

Usually, people attribute the difficulty of deep neural network convergence problem due to factors such as gradient vanish or gradient saturation, and these phenomena do exist in traditional deep neural networks. However, modern deep neural networks have already fundamentally solved the above problem by designing various normalization and activation functions. Nevertheless, deep neural networks still have the problem of slow convergence or poor convergence results.

In this paper, we explore the nature of the above issue further. Through in-depth analysis of information bottleneck, we deduced that the root cause of this problem is that the initial gradient originally coming from a very deep network has lost a lot of information needed to achieve the goal soon after it is transmitted. In order to confirm this inference, we feedforward deep networks of different architectures with initial weights, and then visualize and illustrate them in Figure 2. Obviously, PlainNet has lost a lot of important information required for object detection in deep layers. As for the proportion of important information that ResNet, CSPNet, and GELAN can retain, it is indeed positively related to the accuracy that can be obtained after training. We further design reversible network-based methods to solve the causes of the above problems. In this section we shall elaborate our analysis of information bottleneck principle and reversible functions.

3.1. Information Bottleneck Principle

According to information bottleneck principle, we know that data X may cause information loss when going through transformation, as shown in Eq. 1 below:

$$I(X, X) \geq I(X, f_\theta(X)) \geq I(X, g_\phi(f_\theta(X))), \quad (1)$$

where I indicates mutual information, f and g are transformation functions, and θ and ϕ are parameters of f and g, respectively.

In deep neural networks, $f_\theta(\cdot)$ and $g_\phi(\cdot)$ respectively represent the operations of two consecutive layers in deep neural network. From Eq. 1, we can predict that as the number of network layer becomes deeper, the original data will be more likely to be lost. However, the parameters of the deep neural network are based on the output of the network as well as the given target, and then update the network after generating new gradients by calculating the loss function. As one can imagine, the output of a deeper neural network is less able to retain complete information about the prediction target. This will make it possible to use incomplete information during network training, resulting in unreliable gradients and poor convergence.

One way to solve the above problem is to directly increase the size of the model. When we use a large number of parameters to construct a model, it is more capable of performing a more complete transformation of the data. The above approach allows even if information is lost during the data feedforward process, there is still a chance to retain enough information to perform the mapping to the target. The above phenomenon explains why the width is more important than the depth in most modern models. However, the above conclusion cannot fundamentally solve the problem of unreliable gradients in very deep neural network. Below, we will introduce how to use reversible functions to solve problems and conduct relative analysis.

3.2. Reversible Functions

When a function r has an inverse transformation function v, we call this function reversible function, as shown in Eq. 2.

$$X = v_\zeta(r_\psi(X)), \quad (2)$$

where ψ and ζ are parameters of r and v, respectively. Data X is converted by reversible function without losing information, as shown in Eq. 3.

$$I(X, X) = I(X, r_\psi(X)) = I(X, v_\zeta(r_\psi(X))). \quad (3)$$

When the network's transformation function is composed of reversible functions, more reliable gradients can be obtained to update the model. Almost all of today's popular deep learning methods are architectures that conform to the reversible property, such as Eq. 4.

$$X^{l+1} = X^l + f_\theta^{l+1}(X^l), \quad (4)$$

where l indicates the l-th layer of a PreAct ResNet and f is the transformation function of the l-th layer. PreAct ResNet [22] repeatedly passes the original data X to subsequent layers in an explicit way. Although such a design can make a deep neural network with more than a thousand layers converge very well, it destroys an important reason why we need deep neural networks. That is, for difficult problems, it is difficult for us to directly find simple mapping functions to map data to targets. This also explains why PreAct ResNet performs worse than ResNet [21] when the number of layers is small.

In addition, we tried to use masked modeling that allowed the transformer model to achieve significant breakthroughs. We use approximation methods, such as Eq. 5, to try to find the inverse transformation v of r, so that the transformed features can retain enough information using sparse features. The form of Eq. 5 is as follows:

$$X = v_\zeta(r_\psi(X) \cdot M), \quad (5)$$

where M is a dynamic binary mask. Other methods that are commonly used to perform the above tasks are diffusion model and variational autoencoder, and they both have the function of finding the inverse function. However, when we apply the above approach to a lightweight model, there will be defects because the lightweight model will be under parameterized to a large amount of raw data. Because of the above reason, important information $I(Y, X)$ that maps data X to target Y will also face the same problem. For this issue, we will explore it using the concept of information bottleneck [59]. The formula for information bottleneck is as follows:

$$I(X, X) \geq I(Y, X) \geq I(Y, f_\theta(X)) \geq ... \geq I(Y, \hat{Y}). \quad (6)$$

Generally speaking, $I(Y, X)$ will only occupy a very small part of $I(X, X)$. However, it is critical to the target mission. Therefore, even if the amount of information lost in the feedforward stage is not significant, as long as $I(Y, X)$ is covered, the training effect will be greatly affected. The lightweight model itself is in an under parameterized state, so it is easy to lose a lot of important information in the feedforward stage. Therefore, our goal for the lightweight model is how to accurately filter $I(Y, X)$ from $I(X, X)$. As for fully preserving the information of X, that is difficult to achieve. Based on the above analysis, we hope to propose a new deep neural network training method that can not only generate reliable gradients to update the model, but also be suitable for shallow and lightweight neural networks.

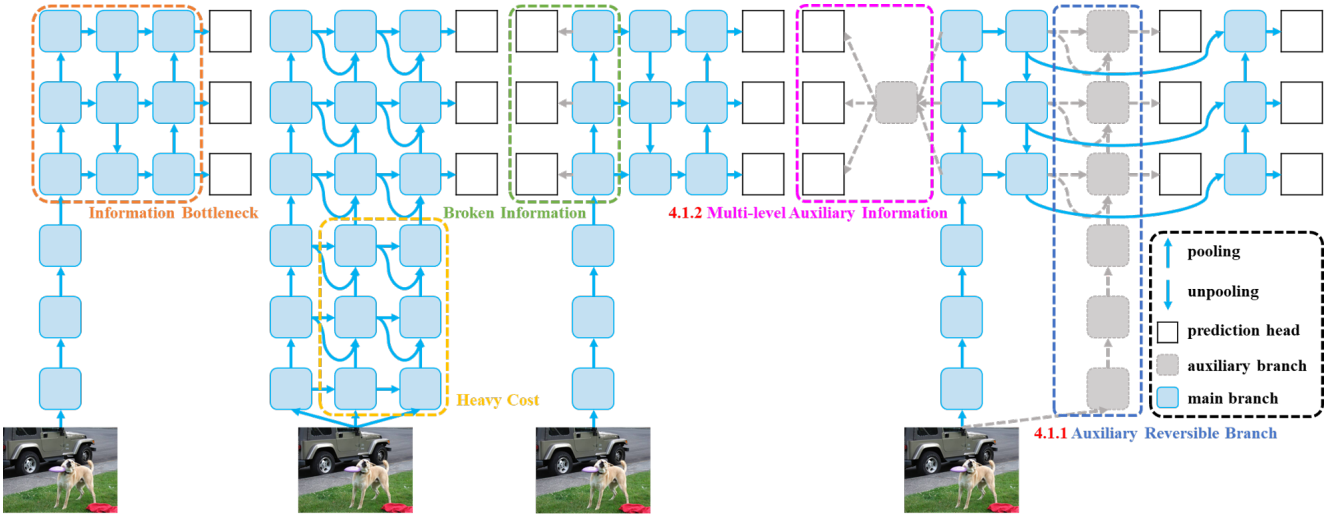


Figure 3. PGI and related network architectures and methods. (a) Path Aggregation Network (PAN)) [37], (b) Reversible Columns (RevCol) [3], (c) conventional deep supervision, and (d) our proposed Programmable Gradient Information (PGI). PGI is mainly composed of three components: (1) main branch: architecture used for inference, (2) auxiliary reversible branch: generate reliable gradients to supply main branch for backward transmission, and (3) multi-level auxiliary information: control main branch learning plannable multi-level of semantic information.

4. Methodology

4.1. Programmable Gradient Information

In order to solve the aforementioned problems, we propose a new auxiliary supervision framework called Programmable Gradient Information (PGI), as shown in Figure 3 (d). PGI mainly includes three components, namely (1) main branch, (2) auxiliary reversible branch, and (3) multi-level auxiliary information. From Figure 3 (d) we see that the inference process of PGI only uses main branch and therefore does not require any additional inference cost. As for the other two components, they are used to solve or slow down several important issues in deep learning methods. Among them, auxiliary reversible branch is designed to deal with the problems caused by the deepening of neural networks. Network deepening will cause information bottleneck, which will make the loss function unable to generate reliable gradients. As for multi-level auxiliary information, it is designed to handle the error accumulation problem caused by deep supervision, especially for the architecture and lightweight model of multiple prediction branch. Next, we will introduce these two components step by step.

4.1.1 Auxiliary Reversible Branch

In PGI, we propose auxiliary reversible branch to generate reliable gradients and update network parameters. By providing information that maps from data to targets, the loss function can provide guidance and avoid the possibility of finding false correlations from incomplete feedforward features that are less relevant to the target. We propose the maintenance of complete information by introducing reversible architecture, but adding main branch to reversible architecture will consume a lot of inference costs. We analyzed the architecture of Figure 3 (b) and found that when additional connections from deep to shallow layers are added, the inference time will increase by 20%. When we repeatedly add the input data to the high-resolution computing layer of the network (yellow box), the inference time even exceeds twice the time.

Since our goal is to use reversible architecture to obtain reliable gradients, "reversible" is not the only necessary condition in the inference stage. In view of this, we regard reversible branch as an expansion of deep supervision branch, and then design auxiliary reversible branch, as shown in Figure 3 (d). As for the main branch deep features that would have lost important information due to information bottleneck, they will be able to receive reliable gradient information from the auxiliary reversible branch. These gradient information will drive parameter learning to assist in extracting correct and important information, and the above actions can enable the main branch to obtain features that are more effective for the target task. Moreover, the reversible architecture performs worse on shallow networks than on general networks because complex tasks require conversion in deeper networks. Our proposed method does not force the main branch to retain complete original information but updates it by generating useful gradient through the auxiliary supervision mechanism. The advantage of this design is that the proposed method can also be applied to shallower networks.

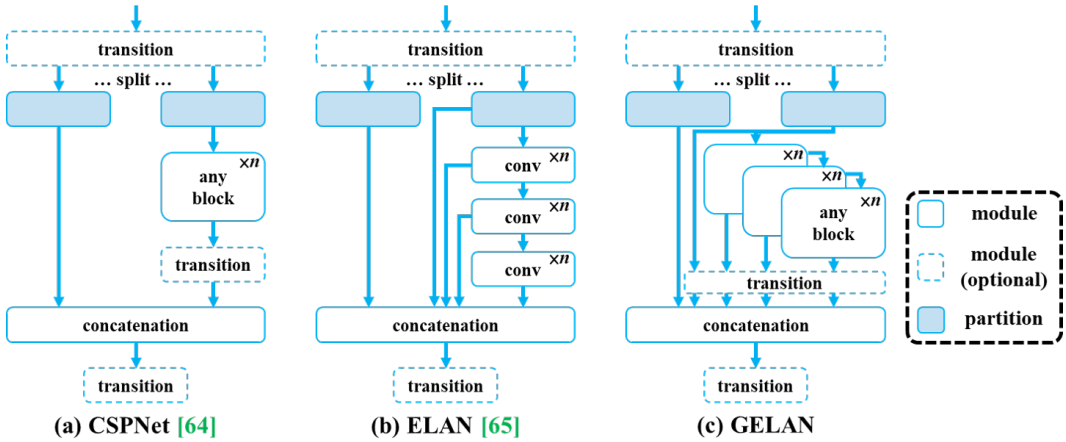


Figure 4. The architecture of GELAN: (a) CSPNet [64], (b) ELAN [65], and (c) proposed GELAN. We imitate CSPNet and extend ELAN into GELAN that can support any computational blocks.

Finally, since auxiliary reversible branch can be removed during the inference phase, the inference capabilities of the original network can be retained. We can also choose any reversible architectures in PGI to play the role of auxiliary reversible branch.

4.1.2 Multi-level Auxiliary Information

In this section we will discuss how multi-level auxiliary information works. The deep supervision architecture including multiple prediction branch is shown in Figure 3 (c). For object detection, different feature pyramids can be used to perform different tasks, for example together they can detect objects of different sizes. Therefore, after connecting to the deep supervision branch, the shallow features will be guided to learn the features required for small object detection, and at this time the system will regard the positions of objects of other sizes as the background. However, the above deed will cause the deep feature pyramids to lose a lot of information needed to predict the target object. Regarding this issue, we believe that each feature pyramid needs to receive information about all target objects so that subsequent main branch can retain complete information to learn predictions for various targets.

The concept of multi-level auxiliary information is to insert an integration network between the feature pyramid hierarchy layers of auxiliary supervision and the main branch, and then uses it to combine returned gradients from different prediction heads, as shown in Figure 3 (d). Multi-level auxiliary information is then to aggregate the gradient information containing all target objects, and pass it to the main branch and then update parameters. At this time, the characteristics of the main branch's feature pyramid hierarchy will not be dominated by some specific object's information. As a result, our method can alleviate the broken information problem in deep supervision. In addition, any integrated network can be used in multi-level auxiliary information. Therefore, we can plan the required semantic levels to guide the learning of network architectures of different sizes.

4.2. Generalized ELAN

In this Section we describe the proposed new network architecture – GELAN. By combining two neural network architectures, CSPNet [64] and ELAN [65], which are designed with gradient path planning, we designed generalized efficient layer aggregation network (GELAN) that takes into account lighweight, inference speed, and accuracy. Its overall architecture is shown in Figure 4. We generalized the capability of ELAN [65], which originally only used stacking of convolutional layers, to a new architecture that can use any computational blocks.

5. Experiments

5.1. Experimental Setup

We verify the proposed method with MS COCO dataset. All experimental setups follow YOLOv7 AF [63], while the dataset is MS COCO 2017 splitting. All models we mentioned are trained using the train-from-scratch strategy, and the total number of training times is 500 epochs. In setting the learning rate, we use linear warm-up in the first three epochs, and the subsequent epochs set the corresponding decay manner according to the model scale. As for the last 15 epochs, we turn mosaic data augmentation off. For more settings, please refer to Appendix.

5.2. Implimentation Details

We built general and extended version of YOLOv9 based on YOLOv7 [63] and Dynamic YOLOv7 [36] respectively. In the design of the network architecture, we replaced ELAN [65] with GELAN using CSPNet blocks [64] with planned RepConv [63] as computational blocks. We also simplified downsampling module and optimized anchor-free prediction head. As for the auxiliary loss part of PGI, we completely follow YOLOv7's auxiliary head setting. Please see Appendix for more details.

Table 1. Comparison of state-of-the-art real-time object detectors.

Model	#Param. (M)	FLOPs (G)	$AP^{val}_{50:95}$ (%)	AP^{val}_{50} (%)	AP^{val}_{75} (%)	AP^{val}_{S} (%)	AP^{val}_{M} (%)	AP^{val}_{L} (%)
YOLOv5-N r7.0 [14]	1.9	4.5	28.0	45.7	–	–	–	–
YOLOv5-S r7.0 [14]	7.2	16.5	37.4	56.8	–	–	–	–
YOLOv5-M r7.0 [14]	21.2	49.0	45.4	64.1	–	–	–	–
YOLOv5-L r7.0 [14]	46.5	109.1	49.0	67.3	–	–	–	–
YOLOv5-X r7.0 [14]	86.7	205.7	50.7	68.9	–	–	–	–
YOLOv6-N v3.0 [30]	4.7	11.4	37.0	52.7	–	–	–	–
YOLOv6-S v3.0 [30]	18.5	45.3	44.3	61.2	–	–	–	–
YOLOv6-M v3.0 [30]	34.9	85.8	49.1	66.1	–	–	–	–
YOLOv6-L v3.0 [30]	59.6	150.7	51.8	69.2	–	–	–	–
YOLOv7 [63]	36.9	104.7	51.2	69.7	55.9	31.8	55.5	65.0
YOLOv7-X [63]	71.3	189.9	52.9	71.1	51.4	36.9	57.7	68.6
YOLOv7-N AF [63]	3.1	8.7	37.6	53.3	40.6	18.7	41.7	52.8
YOLOv7-S AF [63]	11.0	28.1	45.1	61.8	48.9	25.7	50.2	61.2
YOLOv7 AF [63]	43.6	130.5	53.0	70.2	57.5	35.8	58.7	68.9
YOLOv8-N [15]	3.2	8.7	37.3	52.6	–	–	–	–
YOLOv8-S [15]	11.2	28.6	44.9	61.8	–	–	–	–
YOLOv8-M [15]	25.9	78.9	50.2	67.2	–	–	–	–
YOLOv8-L [15]	43.7	165.2	52.9	69.8	57.5	35.3	58.3	69.8
YOLOv8-X [15]	68.2	257.8	53.9	71.0	58.7	35.7	59.3	70.7
DAMO YOLO-T [75]	8.5	18.1	42.0	58.0	45.2	23.0	46.1	58.5
DAMO YOLO-S [75]	12.3	37.8	46.0	61.9	49.5	25.9	50.6	62.5
DAMO YOLO-M [75]	28.2	61.8	49.2	65.5	53.0	29.7	53.1	66.1
DAMO YOLO-L [75]	42.1	97.3	50.8	67.5	55.5	33.2	55.7	66.6
Gold YOLO-N [61]	5.6	12.1	39.6	55.7	–	19.7	44.1	57.0
Gold YOLO-S [61]	21.5	46.0	45.4	62.5	–	25.3	50.2	62.6
Gold YOLO-M [61]	41.3	87.5	49.8	67.0	–	32.3	55.3	66.3
Gold YOLO-L [61]	75.1	151.7	51.8	68.9	–	34.1	57.4	68.2
YOLO MS-N [7]	4.5	17.4	43.4	60.4	47.6	23.7	48.3	60.3
YOLO MS-S [7]	8.1	31.2	46.2	63.7	50.5	26.9	50.5	63.0
YOLO MS [7]	22.2	80.2	51.0	68.6	55.7	33.1	56.1	66.5
GELAN-S (Ours)	7.1	26.4	46.7	63.0	50.7	25.9	51.5	64.0
GELAN-M (Ours)	20.0	76.3	51.1	67.9	55.7	33.6	56.4	67.3
GELAN-C (Ours)	25.3	102.1	52.5	69.5	57.3	35.8	57.6	69.4
GELAN-E (Ours)	57.3	189.0	55.0	71.9	60.0	38.0	60.6	70.9
YOLOv9-S (Ours)	7.1	26.4	46.8	63.4	50.7	26.6	56.0	64.5
YOLOv9-M (Ours)	20.0	76.3	51.4	68.1	56.1	33.6	57.0	68.0
YOLOv9-C (Ours)	25.3	102.1	53.0	70.2	57.8	36.2	58.5	69.3
YOLOv9-E (Ours)	57.3	189.0	55.6	72.8	60.6	40.2	61.0	71.4

5.3. Comparison with state-of-the-arts

Table 1 lists comparison of our proposed YOLOv9 with other train-from-scratch real-time object detectors. Overall, the best performing methods among existing methods are YOLO MS-S [7] for lightweight models, YOLO MS [7] for medium models, YOLOv7 AF [63] for general models, and YOLOv8-X [15] for large models. Compared with lightweight and medium model YOLO MS [7], YOLOv9 has about 10% less parameters and 5∼15% less calculations, but still has a 0.4∼0.6% improvement in AP. Compared with YOLOv7 AF, YOLOv9-C has 42% less parameters and 22% less calculations, but achieves the same AP (53%). Compared with YOLOv8-X, YOLOv9-E has 16% less parameters, 27% less calculations, and has significant improvement of 1.7% AP. The above comparison results show that our proposed YOLOv9 has significantly improved in all aspects compared with existing methods.

On the other hand, we also include ImageNet pretrained model in the comparison, and the results are shown in Figure 5. We compare them based on the parameters and the amount of computation respectively. In terms of the number of parameters, the best performing large model is RT DETR [43]. From Figure 5, we can see that YOLOv9 using conventional convolution is even better than YOLO MS using depth-wise convolution in parameter utilization. As for the parameter utilization of large models, it also greatly surpasses RT DETR using ImageNet pretrained model. Even better is that in the deep model, YOLOv9 shows the huge advantages of using PGI. By accurately retaining and extracting the information needed to map the data to the target, our method requires only 66% of the parameters while maintaining the accuracy as RT DETR-X.

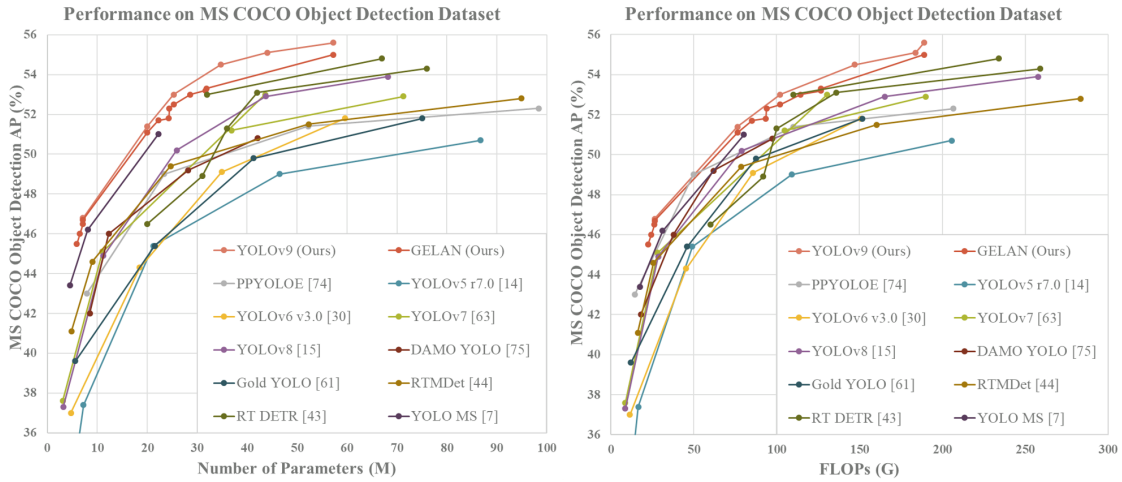


Figure 5. Comparison of state-of-the-art real-time object detectors. The methods participating in the comparison all use ImageNet as pre-trained weights, including RT DETR [43], RTMDet [44], and PP-YOLOE [74], etc. The YOLOv9 that uses train-from-scratch method clearly surpasses the performance of other methods.

As for the amount of computation, the best existing models from the smallest to the largest are YOLO MS [7], PP YOLOE [74], and RT DETR [43]. From Figure 5, we can see that YOLOv9 is far superior to the train-from-scratch methods in terms of computational complexity. In addition, if compared with those based on depth-wise convolution and ImageNet-based pretrained models, YOLOv9 is also very competitive.

5.4. Ablation Studies

5.4.1 Generalized ELAN

For GELAN, we first do ablation studies for computational blocks. We used Res blocks [21], Dark blocks [49], and CSP blocks [64] to conduct experiments, respectively. Table 2 shows that after replacing convolutional layers in ELAN with different computational blocks, the system can maintain good performance. Users are indeed free to replace computational blocks and use them on their respective inference devices. Among different computational block replacements, CSP blocks perform particularly well. They not only reduce the amount of parameters and computation, but also improve AP by 0.7%. Therefore, we choose CSP-ELAN as the component unit of GELAN in YOLOv9.

Table 2. Ablation study on various computational blocks.

Model	CB type	#Param.	FLOPs	$AP^{val}_{50:95}$
GELAN-S	Conv	6.2M	23.5G	44.8%
GELAN-S	Res [21]	5.4M	21.0G	44.3%
GELAN-S	Dark [49]	5.7M	21.8G	44.5%
GELAN-S	CSP [64]	5.9M	22.4G	45.5%

[1] CB type nedotes as computational block type.
[2] -S nedotes small size model.

Next, we conduct ELAN block-depth and CSP block-depth experiments on GELAN of different sizes, and display the results in Table 3. We can see that when the depth of ELAN is increased from 1 to 2, the accuracy is significantly improved. But when the depth is greater than or equal to 2, no matter it is improving the ELAN depth or the CSP depth, the number of parameters, the amount of computation, and the accuracy will always show a linear relationship. This means GELAN is not sensitive to the depth. In other words, users can arbitrarily combine the components in GELAN to design the network architecture, and have a model with stable performance without special design. In Table 3, for YOLOv9-{S,M,C}, we set the pairing of the ELAN depth and the CSP depth to {{2, 3}, {2, 1}, {2, 1}}.

Table 3. Ablation study on ELAN and CSP depth.

Model	$\mathbf{D}_{ELAN}$	$\mathbf{D}_{CSP}$	#Param.	FLOPs	$AP^{val}_{50:95}$
GELAN-S	2	1	5.9M	22.4G	45.5%
GELAN-S	2	2	6.5M	24.4G	46.0%
GELAN-S	3	1	7.1M	26.3G	46.5%
GELAN-S	2	3	7.1M	26.4G	46.7%
GELAN-M	2	1	20.0M	76.3G	51.1%
GELAN-M	2	2	22.2M	85.1G	51.7%
GELAN-M	3	1	24.3M	93.5G	51.8%
GELAN-M	2	3	24.4M	94.0G	52.3%
GELAN-C	1	1	18.9M	77.5G	50.7%
GELAN-C	2	1	25.3M	102.1G	52.5%
GELAN-C	2	2	28.6M	114.4G	53.0%
GELAN-C	3	1	31.7M	126.8G	53.2%
GELAN-C	2	3	31.9M	126.7G	53.3%

[1] $\mathbf{D}_{ELAN}$ and $\mathbf{D}_{CSP}$ respectively nedotes depth of ELAN and CSP.
[2] -{S, M, C} indicate small, medium, and compact models.

5.4.2 Programmable Gradient Information

In terms of PGI, we performed ablation studies on auxiliary reversible branch and multi-level auxiliary information on the backbone and neck, respectively. We designed auxiliary reversible branch ICN to use DHLC [34] linkage to obtain multi-level reversible information. As for multi-level auxiliary information, we use FPN and PAN for ablation studies and the role of PFH is equivalent to the traditional deep supervision. The results of all experiments are listed in Table 4. From Table 4, we can see that PFH is only effective in deep models, while our proposed PGI can improve accuracy under different combinations. Especially when using ICN, we get stable and better results. We also tried to apply the lead-head guided assignment proposed in YOLOv7 [63] to the PGI's auxiliary supervision, and achieved much better performance.

Table 4. Ablation study on PGI of backbone and neck.

Model	$G_{backbone}$	G_{neck}	$AP^{val}_{50:95}$	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
GELAN-C	–	–	52.5%	35.8%	57.6%	**69.4%**
GELAN-C	PFH	–	52.5%	35.3%	58.1%	68.9%
GELAN-C	FPN	–	52.6%	35.3%	58.1%	68.9%
GELAN-C	–	ICN	52.7%	35.3%	58.4%	68.9%
GELAN-C	FPN	ICN	52.8%	35.8%	58.2%	69.1%
GELAN-C	ICN	–	**52.9%**	35.2%	**58.7%**	68.6%
GELAN-C	LHG-ICN	–	**53.0%**	**36.3%**	58.5%	69.1%
GELAN-E	–	–	55.0%	38.0%	60.6%	70.9%
GELAN-E	PFH	–	55.3%	38.3%	60.3%	71.6%
GELAN-E	FPN	–	**55.6%**	**40.2%**	61.0%	71.4%
GELAN-E	PAN	–	55.5%	39.0%	**61.1%**	71.5%
GELAN-E	FPN	ICN	**55.6%**	39.8%	60.9%	**71.9%**

[1] $\mathbf{D}_{ELAN}$ and $\mathbf{D}_{CSP}$ respectively nedotes depth of ELAN and CSP.
[2] LHG indicates lead head guided training proposed by YOLOv7 [63].

We further implemented the concepts of PGI and deep supervision on models of various sizes and compared the results, these results are shown in Table 5. As analyzed at the beginning, introduction of deep supervision will cause a loss of accuracy for shallow models. As for general models, introducing deep supervision will cause unstable performance, and the design concept of deep supervision can only bring gains in extremely deep models. The proposed PGI can effectively handle problems such as information bottleneck and information broken, and can comprehensively improve the accuracy of models of different sizes. The concept of PGI brings two valuable contributions. The first one is to make the auxiliary supervision method applicable to shallow models, while the second one is to make the deep model training process obtain more reliable gradients. These gradients enable deep models to use more accurate information to establish correct correlations between data and targets.

Table 5. Ablation study on PGI.

Model	$AP^{val}_{50:95}$		AP^{val}_{50}		AP^{val}_{75}	
GELAN-S	46.7%		63.0%		**50.7%**	
+ DS	46.5%	-0.2	62.9%	-0.1	50.5%	-0.2
+ PGI	**46.8%**	+0.1	**63.4%**	+0.4	**50.7%**	=
GELAN-M	51.1%		67.9%		55.7%	
+ DS	51.2%	+0.1	**68.2%**	+0.3	55.7%	=
+ PGI	**51.4%**	+0.3	68.1%	+0.2	**56.1%**	+0.4
GELAN-C	52.5%		69.5%		57.3%	
+ DS	52.5%	=	69.9%	+0.4	57.1%	-0.2
+ PGI	**53.0%**	+0.5	**70.3%**	+0.8	**57.8%**	+0.5
GELAN-E	55.0%		71.9%		60.0%	
+ DS	55.3%	+0.3	72.3%	+0.4	60.2%	+0.2
+ PGI	**55.6%**	+0.6	**72.8%**	+0.9	**60.6%**	+0.6

[1] DS indicates deep supervision.
[2] -{S, M, C, E} indicate small, medium, compact, and extended models.

Finally, we show in the table the results of gradually increasing components from baseline YOLOv7 to YOLOv9-E. The GELAN and PGI we proposed have brought all-round improvement to the model.

Table 6. Ablation study on GELAN and PGI.

Model	#Param.	FLOPs	$AP^{val}_{50:95}$	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv7 [63]	36.9	104.7	51.2%	31.8%	55.5%	65.0%
+ AF [63]	43.6	130.5	53.0%	35.8%	58.7%	68.9%
+ GELAN	41.2	126.4	53.2%	36.2%	58.5%	69.9%
+ DHLC [34]	57.3	189.0	55.0%	38.0%	60.6%	70.9%
+ PGI	57.3	189.0	55.6%	40.2%	61.0%	71.4%

5.5. Visualization

This section will explore the information bottleneck issues and visualize them. In addition, we will also visualize how the proposed PGI uses reliable gradients to find the correct correlations between data and targets. In Figure 6 we show the visualization results of feature maps obtained by using random initial weights as feedforward under different architectures. We can see that as the number of layers increases, the original information of all architectures gradually decreases. For example, at the 50^{th} layer of the PlainNet, it is difficult to see the location of objects, and all distinguishable features will be lost at the 100^{th} layer. As for ResNet, although the position of object can still be seen at the 50^{th} layer, the boundary information has been lost. When the depth reached to the 100^{th} layer, the whole image becomes blurry. Both CSPNet and the proposed GELAN perform very well, and they both can maintain features that support clear identification of objects until the 200^{th} layer. Among the comparisons, GELAN has more stable results and clearer boundary information.

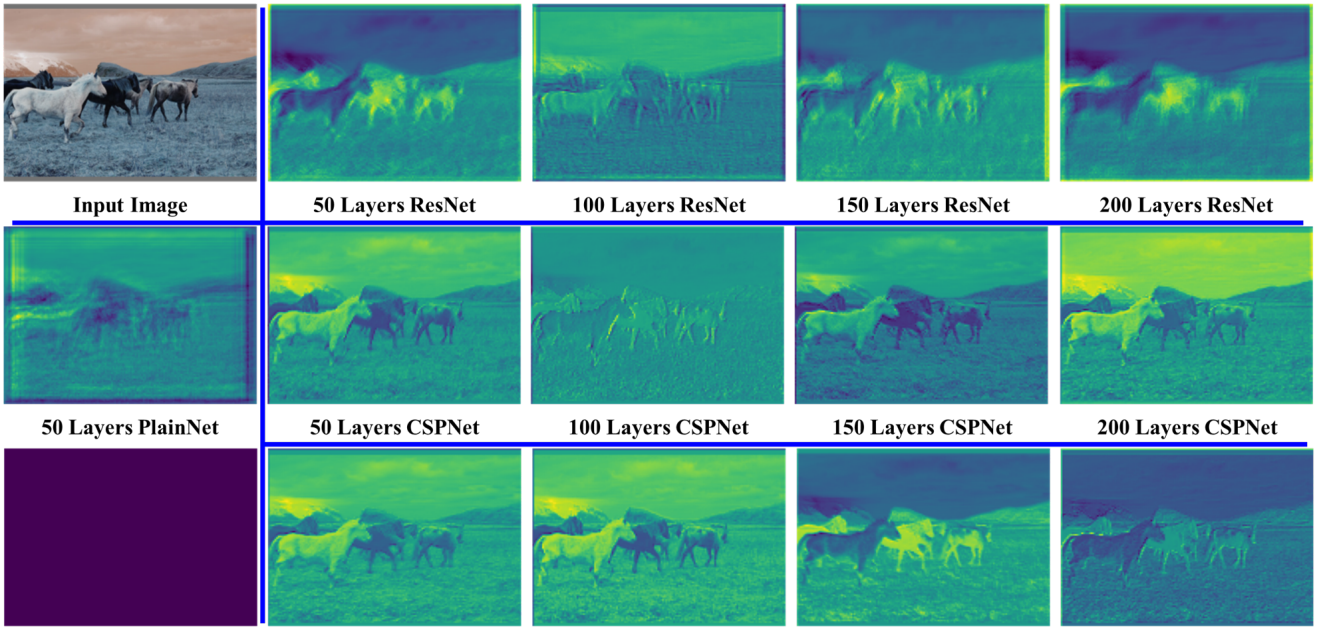


Figure 6. Feature maps (visualization results) output by random initial weights of PlainNet, ResNet, CSPNet, and GELAN at different depths. After 100 layers, ResNet begins to produce feedforward output that is enough to obfuscate object information. Our proposed GELAN can still retain quite complete information up to the 150^{th} layer, and is still sufficiently discriminative up to the 200^{th} layer.

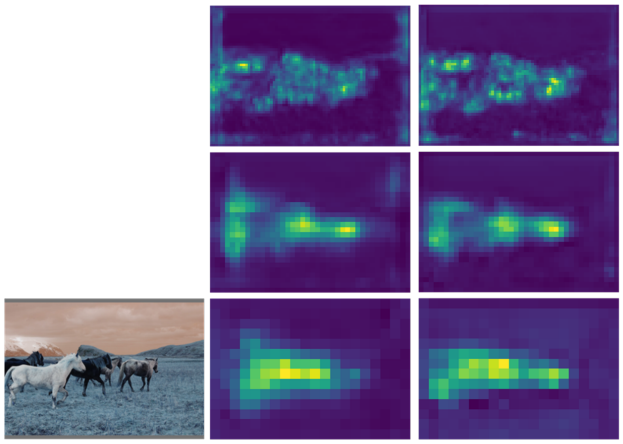


Figure 7. PAN feature maps (visualization results) of GELAN and YOLOv9 (GELAN + PGI) after one epoch of bias warm-up. GELAN originally had some divergence, but after adding PGI's reversible branch, it is more capable of focusing on the target object.

Figure 7 is used to show whether PGI can provide more reliable gradients during the training process, so that the parameters used for updating can effectively capture the relationship between the input data and the target. Figure 7 shows the visualization results of the feature map of GELAN and YOLOv9 (GELAN + PGI) in PAN bias warm-up. From the comparison of Figure 7(b) and (c), we can clearly see that PGI accurately and concisely captures the area containing objects. As for GELAN that does not use PGI, we found that it had divergence when detecting object boundaries, and it also produced unexpected responses in some background areas. This experiment confirms that PGI can indeed provide better gradients to update parameters and enable the feedforward stage of the main branch to retain more important features.

6. Conclusions

In this paper, we propose to use PGI to solve the information bottleneck problem and the problem that the deep supervision mechanism is not suitable for lightweight neural networks. We designed GELAN, a highly efficient and lightweight neural network. In terms of object detection, GELAN has strong and stable performance at different computational blocks and depth settings. It can indeed be widely expanded into a model suitable for various inference devices. For the above two issues, the introduction of PGI allows both lightweight models and deep models to achieve significant improvements in accuracy. The YOLOv9, designed by combining PGI and GELAN, has shown strong competitiveness. Its excellent design allows the deep model to reduce the number of parameters by 49% and the amount of calculations by 43% compared with YOLOv8, but it still has a 0.6% AP improvement on MS COCO dataset.

7. Acknowledgements

The authors wish to thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 3

[3] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 3, 5

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2020. 3

[5] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. AP-loss for accurate one-stage object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(11):3782–3798, 2020. 1

[6] Yabo Chen, Yuchen Liu, Dongsheng Jiang, Xiaopeng Zhang, Wenrui Dai, Hongkai Xiong, and Qi Tian. SdAE: Self-distillated masked autoencoder. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 108–124, 2022. 2

[7] Yuming Chen, Xinbin Yuan, Ruiqi Wu, Jiabao Wang, Qibin Hou, and Ming-Ming Cheng. YOLO-MS: rethinking multi-scale representation learning for real-time object detection. *arXiv preprint arXiv:2308.05480*, 2023. 1, 3, 7, 8

[8] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. DaVIT: Dual attention vision transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 74–92, 2022. 1

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2

[10] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. TOOD: Task-aligned one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499, 2021. 1

[11] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2Net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(2):652–662, 2019. 3

[12] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. OTA: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 303–312, 2021. 1

[13] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 3

[14] Jocher Glenn. YOLOv5 release v7.0. https://github.com/ultralytics/yolov5/releases/tag/v7.0, 2022. 3, 7

[15] Jocher Glenn. YOLOv8 release v8.1.0. https://github.com/ultralytics/ultralytics/releases/tag/v8.1.0, 2024. 3, 7

[16] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 3

[17] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1

[18] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. AugFPN: Improving multi-scale feature learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12595–12604, 2020. 1, 3

[19] Qi Han, Yuxuan Cai, and Xiangyu Zhang. RevColV2: Exploring disentangled representations in masked image modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2, 3

[20] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. Boundary-aware instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5696–5704, 2017. 1, 3

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 4, 8

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016. 1, 4

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. 1

[24] Kuan-Chih Huang, Tsung-Han Wu, Hung-Ting Su, and Winston H Hsu. MonoDTR: Monocular 3D object detection with depth-aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4012–4021, 2022. 1, 3

[25] Lin Huang, Weisheng Li, Linlin Shen, Haojie Fu, Xue Xiao, and Suihan Xiao. YOLOCS: Object detection based on dense channel compression for feature spatial solidification. *arXiv preprint arXiv:2305.04170*, 2023. 3

[26] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, pages 4651–4664, 2021. 1

[27] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, volume 1, page 2, 2019. 2

[28] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015. 1, 2, 3

[29] Alex Levinshtein, Alborz Rezazadeh Sereshkeh, and Konstantinos Derpanis. DATNet: Dense auxiliary tasks for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1419–1427, 2020. 1, 3

[30] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. YOLOv6 v3.0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*, 2023. 3, 7, 2, 4

[31] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022. 3

[32] Hao Li, Jinguo Zhu, Xiaohu Jiang, Xizhou Zhu, Hongsheng Li, Chun Yuan, Xiaohua Wang, Yu Qiao, Xiaogang Wang, Wenhai Wang, et al. Uni-perceiver v2: A generalist model for large-scale vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2700, 2023. 1

[33] Shuai Li, Chenhang He, Ruihuang Li, and Lei Zhang. A dual weighting label assignment scheme for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9387–9396, 2022. 1

[34] Tingting Liang, Xiaojie Chu, Yudong Liu, Yongtao Wang, Zhi Tang, Wei Chu, Jingdong Chen, and Haibin Ling. CBNet: A composite backbone network architecture for object detection. *IEEE Transactions on Image Processing (TIP)*, 2022. 3, 9

[35] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017. 3

[36] Zhihao Lin, Yongtao Wang, Jinhe Zhang, and Xiaojie Chu. DynamicDet: A unified dynamic architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6282–6291, 2023. 3, 6, 2, 4

[37] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018. 3, 5

[38] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024. 1

[39] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. CBNet: A novel composite backbone network architecture for object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11653–11660, 2020. 3

[40] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[41] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 1

[42] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022. 1

[43] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. DETRs beat YOLOs on real-time object detection. *arXiv preprint arXiv:2304.08069*, 2023. 1, 3, 7, 8, 2, 4

[44] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. RTMDet: An empirical study of designing real-time object detectors. *arXiv preprint arXiv:2212.07784*, 2022. 8, 2, 3, 4

[45] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:15534–15545, 2020. 1

[46] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Rank & sort loss for object detection and instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3009–3018, 2021. 1

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 3

[48] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017. 3

[49] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 3, 8

[50] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019. 1

[51] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Object detection from scratch with deep supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(2):398–412, 2019. 1, 2

[52] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation.

In *Conference on Robot Learning (CoRL)*, pages 785–799, 2023. 1

[53] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang, and Ping Luo. What makes for end-to-end object detection? In *International Conference on Machine Learning (ICML)*, pages 9934–9944, 2021. 3

[54] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1, 2, 3

[55] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 1

[56] Zineng Tang, Jaemin Cho, Jie Lei, and Mohit Bansal. Perceiver-VL: Efficient vision-and-language modeling with iterative latent attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4410–4420, 2023. 1

[57] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9627–9636, 2019. 3

[58] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(4):1922–1933, 2022. 3

[59] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015. 2, 4

[60] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxVIT: Multi-axis vision transformer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 459–479, 2022. 1

[61] Chengcheng Wang, Wei He, Ying Nie, Jianyuan Guo, Chuanjian Liu, Kai Han, and Yunhe Wang. Gold-YOLO: Efficient object detector via gather-and-distribute mechanism. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3, 7, 2, 4

[62] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, 2021. 3

[63] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7475, 2023. 3, 6, 7, 9, 1

[64] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 390–391, 2020. 3, 6, 8

[65] Chien-Yao Wang, Hong-Yuan Mark Liao, and I-Hau Yeh. Designing network design strategies through gradient path analysis. *Journal of Information Science and Engineering (JISE)*, 39(4):975–995, 2023. 2, 3, 6

[66] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *Journal of Information Science & Engineering (JISE)*, 39(3):691–709, 2023. 2, 3, 4

[67] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15849–15858, 2021. 1, 3

[68] Liwei Wang, Chen-Yu Lee, Zhuowen Tu, and Svetlana Lazebnik. Training deeper convolutional networks with deep supervision. *arXiv preprint arXiv:1505.02496*, 2015. 1, 2, 3

[69] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, 2021. 1

[70] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022. 1

[71] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16133–16142, 2023. 1, 2

[72] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017. 1

[73] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9653–9663, 2022. 2

[74] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. PP-YOLOE: An evolved version of YOLO. *arXiv preprint arXiv:2203.16250*, 2022. 3, 8, 2, 4

[75] Xianzhe Xu, Yiqi Jiang, Weihua Chen, Yilun Huang, Yuan Zhang, and Xiuyu Sun. DAMO-YOLO: A report on real-time object detection design. *arXiv preprint arXiv:2211.15444*, 2022. 3, 7, 2, 4

[76] Renrui Zhang, Han Qiu, Tai Wang, Ziyu Guo, Ziteng Cui, Yu Qiao, Hongsheng Li, and Peng Gao. MonoDETR: Depth-guided transformer for monocular 3D object detection. In

Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 9155–9166, 2023. 1, 3

[77] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 12993–13000, 2020. 1

[78] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. IoU loss for 2D/3D object detection. In *International Conference on 3D Vision (3DV)*, pages 85–94, 2019. 1

[79] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. AutoAssign: Differentiable label assignment for dense object detection. *arXiv preprint arXiv:2007.03496*, 2020. 1

[80] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 1

[81] Xizhou Zhu, Jinguo Zhu, Hao Li, Xiaoshi Wu, Hongsheng Li, Xiaohua Wang, and Jifeng Dai. Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16804–16815, 2022. 1

[82] Zhuofan Zong, Guanglu Song, and Yu Liu. DETRs with collaborative hybrid assignments training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6748–6758, 2023. 3

Appendix

A. Implementation Details

Table 1. Hyper parameter settings of YOLOv9.

hyper parameter	value
epochs	500
optimizer	SGD
initial learning rate	0.01
finish learning rate	0.0001
learning rate decay	linear
momentum	0.937
weight decay	0.0005
warm-up epochs	3
warm-up momentum	0.8
warm-up bias learning rate	0.1
box loss gain	7.5
class loss gain	0.5
DFL loss gain	1.5
HSV saturation augmentation	0.7
HSV value augmentation	0.4
translation augmentation	0.1
scale augmentation	0.9
mosaic augmentation	1.0
MixUp augmentation	0.15
copy & paste augmentation	0.3
close mosaic epochs	15

The training parameters of YOLOv9 are shown in Table 1. We fully follow the settings of YOLOv7 AF [63], which is to use SGD optimizer to train 500 epochs. We first warm-up for 3 epochs and only update the bias during the warm-up stage. Next we step down from the initial learning rate 0.01 to 0.0001 in linear decay manner, and the data augmentation settings are listed in the bottom part of Table 1. We shut down mosaic data augmentation operations on the last 15 epochs.

Table 2. Network configurations of YOLOv9.

Index	Module	Route	Filters	Depth	Size	Stride
0	Conv	–	64	–	3	2
1	Conv	0	128	–	3	2
2	CSP-ELAN	1	256, 128, 64	2, 1	–	1
3	DOWN	2	256	–	3	2
4	CSP-ELAN	3	512, 256, 128	2, 1	–	1
5	DOWN	4	512	–	3	2
6	CSP-ELAN	5	512, 512, 256	2, 1	–	1
7	DOWN	6	512	–	3	2
8	CSP-ELAN	7	512, 512, 256	2, 1	–	1
9	SPP-ELAN	8	512, 256, 256	3, 1	–	1
10	Up	9	512	–	–	2
11	Concat	10, 6	1024	–	–	1
12	CSP-ELAN	11	512, 512, 256	2, 1	–	1
13	Up	12	512	–	–	2
14	Concat	13, 4	1024	–	–	1
15	CSP-ELAN	14	256, 256, 128	2, 1	–	1
16	DOWN	15	256	–	3	2
17	Concat	16, 12	768	–	–	1
18	CSP-ELAN	17	512, 512, 256	2, 1	–	1
19	DOWN	18	512	–	3	2
20	Concat	19, 9	1024	–	–	1
21	CSP-ELAN	20	512, 512, 256	2, 1	–	1
22	Predict	15, 18, 21	–	–	–	–

The network topology of YOLOv9 completely follows YOLOv7 AF [63], that is, we replace ELAN with the proposed CSP-ELAN block. As listed in Table 2, the depth parameters of CSP-ELAN are represented as ELAN depth and CSP depth, respectively. As for the parameters of CSP-ELAN filters, they are represented as ELAN output filter, CSP output filter, and CSP inside filter. In the downsampling module part, we simplify CSP-DOWN module to DOWN module. DOWN module is composed of a pooling layer with size 2 and stride 1, and a Conv layer with size 3 and stride 2. Finally, we optimized the prediction layer and replaced top, left, bottom, and right in the regression branch with decoupled branch.

Table 3. Comparison of state-of-the-art object detectors with different training settings.

	Model	#Param. (M)	FLOPs (G)	$AP_{50:95}$ (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
Train-from-scratch	**Dy-YOLOv7 [36]**	–	181.7	53.9	72.2	58.7	35.3	57.6	66.4
	Dy-YOLOv7-X [36]	–	307.9	55.0	73.2	60.0	36.6	58.7	68.5
	YOLOv9-S (Ours)	7.1	26.4	46.8	63.4	50.7	26.6	56.0	64.5
	YOLOv9-M (Ours)	20.0	76.3	51.4	68.1	56.1	33.6	57.0	68.0
	YOLOv9-C (Ours)	25.3	102.1	53.0	70.2	57.8	36.2	58.5	69.3
	YOLOv9-E (Ours)	34.7	147.1	54.5	71.7	59.2	38.1	59.9	70.3
	YOLOv9-E (Ours)	44.0	183.9	55.1	72.3	60.7	38.7	60.6	71.4
	YOLOv9-E (Ours)	57.3	189.0	55.6	72.8	60.6	40.2	61.0	71.4
ImageNet Pretrained	**RTMDet-T [44]**	4.8	12.6	41.1	57.9	–	–	–	–
	RTMDet-S [44]	9.0	25.6	44.6	61.9	–	–	–	–
	RTMDet-M [44]	24.7	78.6	49.4	66.8	–	–	–	–
	RTMDet-L [44]	52.3	160.4	51.5	68.8	–	–	–	–
	RTMDet-X [44]	94.9	283.4	52.8	70.4	–	–	–	–
	PPYOLOE-S [74]	7.9	14.4	43.0	60.5	46.6	23.2	46.4	56.9
	PPYOLOE-M [74]	23.4	49.9	49.0	66.5	53.0	28.6	52.9	63.8
	PPYOLOE-L [74]	52.2	110.1	51.4	68.9	55.6	31.4	55.3	66.1
	PPYOLOE-X [74]	98.4	206.6	52.3	69.5	56.8	35.1	57.0	68.6
	RT DETR-L [43]	32	110	53.0	71.6	57.3	34.6	57.3	71.2
	RT DETR-X [43]	67	234	54.8	73.1	59.4	35.7	59.6	72.9
	RT DETR-R18 [43]	20	60	46.5	63.8	–	–	–	–
	RT DETR-R34 [43]	31	92	48.9	66.8	–	–	–	–
	RT DETR-R50M [43]	36	100	51.3	69.6	–	–	–	–
	RT DETR-R50 [43]	42	136	53.1	71.3	57.7	34.8	58.0	70.0
	RT DETR-R101 [43]	76	259	54.3	72.7	58.6	36.0	58.8	72.1
	Gold YOLO-S [61]	21.5	46.0	45.5	62.2	–	–	–	–
	Gold YOLO-M [61]	41.3	57.5	50.2	67.5	–	–	–	–
	Gold YOLO-L [61]	75.1	151.7	52.3	69.6	–	–	–	–
Knowledge Distillation	**YOLOv6-N v3.0 [30]**	4.7	11.4	37.5	53.1	–	–	–	–
	YOLOv6-S v3.0 [30]	18.5	45.3	45.0	61.8	–	–	–	–
	YOLOv6-M v3.0 [30]	34.9	85.8	50.0	66.9	–	–	–	–
	YOLOv6-L v3.0 [30]	59.6	150.7	52.8	70.3	–	–	–	–
	DAMO YOLO-T [75]	8.5	18.1	43.6	59.4	46.6	23.3	47.4	61.0
	DAMO YOLO-S [75]	16.3	37.8	47.7	63.5	51.1	26.9	51.7	64.9
	DAMO YOLO-M [75]	28.2	61.8	50.4	67.2	55.1	31.6	55.3	67.1
	DAMO YOLO-L [75]	42.1	97.3	51.9	68.5	56.7	33.3	57.0	67.6
	Gold YOLO-N [61]	5.6	12.1	39.9	55.9	–	–	–	–
	Gold YOLO-S [61]	21.5	46.0	46.1	63.3	–	–	–	–
	Gold YOLO-M [61]	41.3	57.5	50.9	68.2	–	–	–	–
	Gold YOLO-L [61]	75.1	151.7	53.2	70.5	–	–	–	–
Complex Setting	**Gold YOLO-S [61]**	21.5	46.0	46.4	63.4	–	–	–	–
	Gold YOLO-M [61]	41.3	57.5	51.1	68.5	–	–	–	–
	Gold YOLO-L [61]	75.1	151.7	53.3	70.9	–	–	–	–
	YOLOR-CSP [66]	52.9	120.4	52.8	71.2	57.6	–	–	–
	YOLOR-CSP-X [66]	96.9	226.8	54.8	73.1	59.7	–	–	–
	PPYOLOE+-S [74]	7.9	14.4	43.7	60.6	47.9	23.2	46.4	56.9
	PPYOLOE+-M [74]	23.4	49.9	49.8	67.1	54.5	31.8	53.9	66.2
	PPYOLOE+-L [74]	52.2	110.1	52.9	70.1	57.9	35.2	57.5	69.1
	PPYOLOE+-X [74]	98.4	206.6	54.7	72.0	59.9	37.9	59.3	70.4

B. More Comparison

We compare YOLOv9 to state-of-the-art real-time object detectors trained with different methods. It mainly includes four different training methods: (1) train-from-scratch: we have completed most of the comparisons in the text. Here are only list of additional data of DynamicDet [36] for comparisons; (2) Pretrained by ImageNet: this includes two methods of using ImageNet for supervised pretrain and self-supervised pretrain; (3) knowledge distillation: a method to perform additional self-distillation after training is completed; and (4) a more complex training process: a combination of steps including pretrained by ImageNet, knowledge distillation, DAMO-YOLO and even additional pretrained large object detection dataset. We show the results in Table 3. From this table, we can see that our proposed YOLOv9 performed better than all other methods. Compared with PPYOLOE+-X trained using ImageNet and Objects365, our method still reduces the number of parameters by 55% and the amount of computation by 11%, and improving 0.4% AP.

Table 4. Comparison of state-of-the-art object detectors with different training settings (sorted by number of parameters).

Model	#Param. (M)	FLOPs (G)	$AP^{val}_{50:95}$ (%)	AP^{val}_{50} (%)	AP^{val}_{75} (%)	AP^{val}_{S} (%)	AP^{val}_{M} (%)	AP^{val}_{L} (%)
YOLOv6-N v3.0 [30] (D)	4.7	11.4	**37.5**	53.1	–	–	–	–
RTMDet-T [44] (I)	4.8	12.6	**41.1**	57.9	–	–	–	–
Gold YOLO-N [61] (D)	5.6	12.1	39.9	55.9	–	–	–	–
YOLOv9-S (S)	7.1	26.4	**46.8**	63.4	50.7	26.6	56.0	64.5
PPYOLOE+-S [74] (C)	7.9	14.4	43.7	60.6	47.9	23.2	46.4	56.9
PPYOLOE-S [74] (I)	7.9	14.4	43.0	60.5	46.6	23.2	46.4	56.9
DAMO YOLO-T [75] (D)	8.5	18.1	43.6	59.4	46.6	23.3	47.4	61.0
RTMDet-S [44] (I)	9.0	25.6	44.6	61.9	–	–	–	–
DAMO YOLO-S [75] (D)	16.3	37.8	**47.7**	63.5	51.1	26.9	51.7	64.9
YOLOv6-S v3.0 [30] (D)	18.5	45.3	45.0	61.8	–	–	–	–
RT DETR-R18 [43] (I)	20	60	46.5	63.8	–	–	–	–
YOLOv9-M (S)	20.0	76.3	**51.4**	68.1	56.1	33.6	57.0	68.0
Gold YOLO-S [61] (C)	21.5	46.0	46.4	63.4	–	–	–	–
Gold YOLO-S [61] (D)	21.5	46.0	46.1	63.3	–	–	–	–
Gold YOLO-S [61] (I)	21.5	46.0	45.5	62.2	–	–	–	–
PPYOLOE+-M [74] (C)	23.4	49.9	49.8	67.1	54.5	31.8	53.9	66.2
PPYOLOE-M [74] (I)	23.4	49.9	49.0	66.5	53.0	28.6	52.9	63.8
RTMDet-M [44] (I)	24.7	78.6	49.4	66.8	–	–	–	–
YOLOv9-C (S)	25.3	102.1	**53.0**	70.2	57.8	36.2	58.5	69.3
DAMO YOLO-M [75] (D)	28.2	61.8	50.4	67.2	55.1	31.6	55.3	67.1
RT DETR-R34 [43] (I)	31	92	48.9	66.8	–	–	–	–
RT DETR-L [43] (I)	32	110	53.0	71.6	57.3	34.6	57.3	71.2
YOLOv9-E (S)	34.7	147.1	**54.5**	71.7	59.2	38.1	59.9	70.3
YOLOv6-M v3.0 [30] (D)	34.9	85.8	50.0	66.9	–	–	–	–
RT DETR-R50M [43] (I)	36	100	51.3	69.6	–	–	–	–
Gold YOLO-M [61] (C)	41.3	57.5	51.1	68.5	–	–	–	–
Gold YOLO-M [61] (D)	41.3	57.5	50.9	68.2	–	–	–	–
Gold YOLO-M [61] (I)	41.3	57.5	50.2	67.5	–	–	–	–
RT DETR-R50 [43] (I)	42	136	53.1	71.3	57.7	34.8	58.0	70.0
DAMO YOLO-L [75] (D)	42.1	97.3	51.9	68.5	56.7	33.3	57.0	67.6
YOLOv9-E (S)	44.0	183.9	**55.1**	72.3	60.7	38.7	60.6	71.4
PPYOLOE+-L [74] (C)	52.2	110.1	52.9	70.1	57.9	35.2	57.5	69.1
PPYOLOE-L [74] (I)	52.2	110.1	51.4	68.9	55.6	31.4	55.3	66.1
RTMDet-L [44] (I)	52.3	160.4	51.5	68.8	–	–	–	–
YOLOR-CSP [66] (C)	52.9	120.4	52.8	71.2	57.6	–	–	–
YOLOv9-E (S)	57.3	189.0	**55.6**	72.8	60.6	40.2	61.0	71.4
YOLOv6-L v3.0 [30] (D)	59.6	150.7	52.8	70.3	–	–	–	–
RT DETR-X [43] (I)	67	234	54.8	73.1	59.4	35.7	59.6	72.9
Gold YOLO-L [61] (C)	75.1	151.7	53.3	70.9	–	–	–	–
Gold YOLO-L [61] (D)	75.1	151.7	53.2	70.5	–	–	–	–
Gold YOLO-L [61] (I)	75.1	151.7	52.3	69.6	–	–	–	–
RT DETR-R101 [43] (I)	76	259	54.3	72.7	58.6	36.0	58.8	72.1
RTMDet-X [44] (I)	94.9	283.4	52.8	70.4	–	–	–	–
YOLOR-CSP-X [66] (C)	96.9	226.8	54.8	73.1	59.7	–	–	–
PPYOLOE+-X [74] (C)	98.4	206.6	54.7	72.0	59.9	37.9	59.3	70.4
PPYOLOE-X [74] (I)	98.4	206.6	52.3	69.5	56.8	35.1	57.0	68.6

[1] (S), (I), (D), (C) indicate train-from-scratch, ImageNet pretrained, knowledge distillation, and complex setting, respectively.

Table 4 shows the performance of all models sorted by parameter size. Our proposed YOLOv9 is Pareto optimal in all models of different sizes. Among them, we found no other method for Pareto optimal in models with more than 20M parameters. The above experimental data shows that our YOLOv9 has excellent parameter usage efficiency.

Shown in Table 5 is the performance of all participating models sorted by the amount of computation. Our proposed YOLOv9 is Pareto optimal in all models with different scales. Among models with more than 60 GFLOPs, only ELAN-based DAMO-YOLO and DETR-based RT DETR can rival the proposed YOLOv9. The above comparison results show that YOLOv9 has the most outstanding performance in the trade-off between computation complexity and accuracy.

Table 5. Comparison of state-of-the-art object detectors with different training settings (sorted by amount of computation).

Model	#Param. (M)	FLOPs (G)	$AP^{val}_{50:95}$ (%)	AP^{val}_{50} (%)	AP^{val}_{75} (%)	AP^{val}_{S} (%)	AP^{val}_{M} (%)	AP^{val}_{L} (%)
YOLOv6-N v3.0 [30] (D)	4.7	11.4	**37.5**	53.1	–	–	–	–
Gold YOLO-N [61] (D)	5.6	12.1	**39.9**	55.9	–	–	–	–
RTMDet-T [44] (I)	4.8	12.6	**41.1**	57.9	–	–	–	–
PPYOLOE+-S [74] (C)	7.9	14.4	**43.7**	60.6	47.9	23.2	46.4	56.9
PPYOLOE-S [74] (I)	7.9	14.4	43.0	60.5	46.6	23.2	46.4	56.9
DAMO YOLO-T [75] (D)	8.5	18.1	43.6	59.4	46.6	23.3	47.4	61.0
RTMDet-S [44] (I)	9.0	25.6	**44.6**	61.9	–	–	–	–
YOLOv9-S (S)	7.1	26.4	**46.8**	63.4	50.7	26.6	56.0	64.5
DAMO YOLO-S [75] (D)	16.3	37.8	**47.7**	63.5	51.1	26.9	51.7	64.9
YOLOv6-S v3.0 [30] (D)	18.5	45.3	45.0	61.8	–	–	–	–
Gold YOLO-S [61] (C)	21.5	46.0	46.4	63.4	–	–	–	–
Gold YOLO-S [61] (D)	21.5	46.0	46.1	63.3	–	–	–	–
Gold YOLO-S [61] (I)	21.5	46.0	45.5	62.2	–	–	–	–
PPYOLOE+-M [74] (C)	23.4	49.9	**49.8**	67.1	54.5	31.8	53.9	66.2
PPYOLOE-M [74] (I)	23.4	49.9	49.0	66.5	53.0	28.6	52.9	63.8
Gold YOLO-M [61] (C)	41.3	57.5	**51.1**	68.5	–	–	–	–
Gold YOLO-M [61] (D)	41.3	57.5	50.9	68.2	–	–	–	–
Gold YOLO-M [61] (I)	41.3	57.5	50.2	67.5	–	–	–	–
RT DETR-R18 [43] (I)	20	60	46.5	63.8	–	–	–	–
DAMO YOLO-M [75] (D)	28.2	61.8	50.4	67.2	55.1	31.6	55.3	67.1
YOLOv9-M (S)	20.0	76.3	**51.4**	68.1	56.1	33.6	57.0	68.0
RTMDet-M [44] (I)	24.7	78.6	49.4	66.8	–	–	–	–
YOLOv6-M v3.0 [30] (D)	34.9	85.8	50.0	66.9	–	–	–	–
RT DETR-R34 [43] (I)	31	92	48.9	66.8	–	–	–	–
DAMO YOLO-L [75] (D)	42.1	97.3	**51.9**	68.5	56.7	33.3	57.0	67.6
RT DETR-R50M [43] (I)	36	100	51.3	69.6	–	–	–	–
YOLOv9-C (S)	25.3	102.1	**53.0**	70.2	57.8	36.2	58.5	69.3
RT DETR-L [43] (I)	32	110	53.0	71.6	57.3	34.6	57.3	71.2
PPYOLOE+-L [74] (C)	52.2	110.1	52.9	70.1	57.9	35.2	57.5	69.1
PPYOLOE-L [74] (I)	52.2	110.1	51.4	68.9	55.6	31.4	55.3	66.1
YOLOR-CSP [66] (C)	52.9	120.4	52.8	71.2	57.6	–	–	–
RT DETR-R50 [43] (I)	42	136	**53.1**	71.3	57.7	34.8	58.0	70.0
YOLOv9-E (S)	34.7	147.1	**54.5**	71.7	59.2	38.1	59.9	70.3
YOLOv6-L v3.0 [30] (D)	59.6	150.7	52.8	70.3	–	–	–	–
Gold YOLO-L [61] (C)	75.1	151.7	53.3	70.9	–	–	–	–
Gold YOLO-L [61] (D)	75.1	151.7	53.2	70.5	–	–	–	–
Gold YOLO-L [61] (I)	75.1	151.7	52.3	69.6	–	–	–	–
RTMDet-L [44] (I)	52.3	160.4	51.5	68.8	–	–	–	–
Dy-YOLOv7 [36] (S)	–	181.7	53.9	72.2	58.7	35.3	57.6	66.4
YOLOv9-E (S)	44.0	183.9	**55.1**	72.3	60.7	38.7	60.6	71.4
YOLOv9-E (S)	57.3	189.0	**55.6**	72.8	60.6	40.2	61.0	71.4
PPYOLOE+-X [74] (C)	98.4	206.6	54.7	72.0	59.9	37.9	59.3	70.4
PPYOLOE-X [74] (I)	98.4	206.6	52.3	69.5	56.8	35.1	57.0	68.6
YOLOR-CSP-X [66] (C)	96.9	226.8	54.8	73.1	59.7	–	–	–
RT DETR-X [43] (I)	67	234	54.8	73.1	59.4	35.7	59.6	72.9
RT DETR-R101 [43] (I)	76	259	54.3	72.7	58.6	36.0	58.8	72.1
RTMDet-X [44] (I)	94.9	283.4	52.8	70.4	–	–	–	–
Dy-YOLOv7-X [36] (S)	–	307.9	55.0	73.2	60.0	36.6	58.7	68.5

[1] (S), (I), (D), (C) indicate train-from-scratch, ImageNet pretrained, knowledge distillation, and complex setting, respectively.