

CBNet: A Composite Backbone Network Architecture for Object Detection

Tingting Liang*, Xiaojie Chu*, Yudong Liu*, Yongtao Wang, Zhi Tang, Wei Chu, Jingdong Chen, Haibin Ling

Abstract—Modern top-performing object detectors depend heavily on backbone networks, whose advances bring consistent performance gains through exploring more effective network structures. In this paper, we propose a novel and flexible backbone framework, namely *CBNet*, to construct high-performance detectors using *existing* open-source pre-trained backbones under the pre-training fine-tuning paradigm. In particular, CBNet architecture groups multiple identical backbones, which are connected through composite connections. Specifically, it integrates the high- and low-level features of multiple identical backbone networks and gradually expands the receptive field to more effectively perform object detection. We also propose a better training strategy with *auxiliary supervision* for CBNet-based detectors. CBNet has strong generalization capabilities for different backbones and head designs of the detector architecture. Without additional pre-training of the composite backbone, CBNet can be adapted to various backbones (*i.e.*, CNN-based vs. Transformer-based) and head designs of most mainstream detectors (*i.e.*, one-stage vs. two-stage, anchor-based vs. anchor-free-based). Experiments provide strong evidence that, compared with simply increasing the depth and width of the network, CBNet introduces a more efficient, effective, and resource-friendly way to build high-performance backbone networks. Particularly, our CB-Swin-L achieves 59.4% box AP and 51.6% mask AP on COCO test-dev under the single-model and single-scale testing protocol, which are significantly better than the state-of-the-art results (*i.e.*, 57.7% box AP and 50.2% mask AP) achieved by Swin-L, while reducing the training time by 6×. With multi-scale testing, we push the current best single model result to a new record of 60.1% box AP and 52.3% mask AP without using extra training data. Code is available at <https://github.com/VDIGPKU/CBNetV2>.

Index Terms—Deep Learning, Object Detection, Backbone Networks, Composite Architectures.

I. INTRODUCTION

OBJECT detection aims to locate each object instance from a predefined set of classes in an arbitrary image. It serves a wide range of applications such as autonomous driving, intelligent video surveillance, remote sensing, *etc.* In recent years, great progresses have been made for object detection thanks to the booming development of deep convolutional networks [2], and excellent detectors have been proposed, *e.g.*,

SSD [3], YOLO [4], Faster R-CNN [5], RetinaNet [6], ATSS [7], Mask R-CNN [8], Cascade R-CNN [9], *etc.*

Typically, in a Neural Network (NN)-based detector, a backbone network is used to extract basic features for detecting objects, and in most cases designed originally for image classification and pre-trained on ImageNet [10]. Intuitively, the more representative features extracted by the backbone, the better the performance of its host detector. To obtain higher accuracy, deeper and wider backbones have been exploited by mainstream detectors (*i.e.*, from mobile-size models [11], [12] and ResNet [13], to ResNeXt [14] and Res2Net [15]). Recently, Transformer [16], [17] based backbones have shown very promising performance. Overall, advances in large backbone pre-training demonstrate a trend towards more effective multi-scale representations in object detection.

Encouraged by the results achieved by pre-trained large backbone-based detectors, we seek further improvement to construct high-performance detectors by exploiting existing well-designed backbone architectures and their pre-trained weights. Though one may design a new improved backbone, the expertise and computing resources overhead can be expensive. On the one hand, designing a new architecture of backbone requires expert experience and a lot of trials and errors. On the other hand, pre-training a new backbone (especially for large models) on ImageNet requires a large number of computational resources, which makes it costly to obtain better detection performance following the pre-training and fine-tuning paradigm. Alternatively, training detectors from scratch saves the cost of pre-training but requires even more computing resources and training skills [18].

In this paper, we present a simple and novel composition approach to use existing pre-trained backbones under the pre-training fine-tuning paradigm. Unlike most previous methods that focus on modular crafting and require pre-training on ImageNet to strengthen the representation, we improve the existing backbone representation ability without additional pre-training. As shown in Fig. 1, our solution, named *Composite Backbone Network* (CBNet), groups multiple identical backbones together. Specifically, parallel backbones (named assisting backbones and lead backbone) are connected via *composite connections*. From left to right in Fig. 1, the output of each stage in an assisting backbone flows to the parallel and lower-level stages of its succeeding sibling. Finally, the features of the lead backbone are fed to the neck and detection head for bounding box regression and classification. Contrary to simple network deepening or widening, CBNet integrates the high- and low-level features of multiple backbone networks and progressively expands the receptive field for more effective

Corresponding author: Yongtao Wang. * indicates equal contribution.

T. Liang, X. Chu, Y. Liu, Y. Wang, Z. Tang are with the Wangxuan Institute of Computer Technology, Peking University, Beijing 100080, China. E-mail: {tingtingliang, bahuangliuhe, wyt, tangzhi}@pku.edu.cn, chuxiaojie@stu.pku.edu.cn

W. Chu, J. Cheng are with Ant Group of Alibaba, China. E-mail: weichu.cw@alibaba-inc.com, jingdongchen.cjd@antfin.com

H. Ling is with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA. E-mail: hling@cs.stonybrook.edu.

A preliminary version of this manuscript was published in [1].

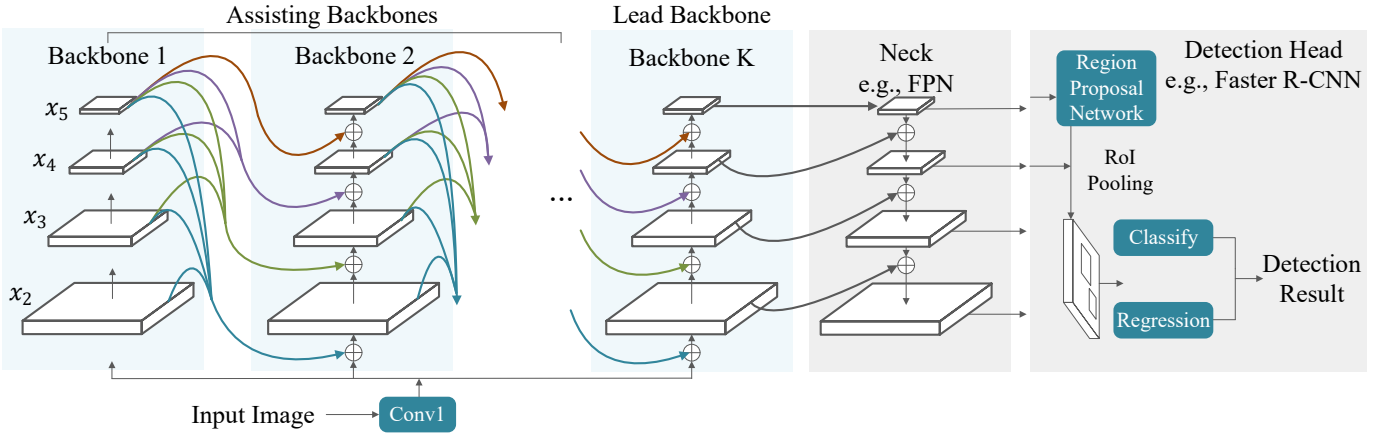


Fig. 1: Illustration of the proposed Composite Backbone Network (CBNet) architecture for object detection.

object detection. Notably, each composed backbone of CBNet is initialized by the weights of an existing open-source pre-trained individual backbone (e.g., CB-ResNet50 is initialized by the weights of ResNet50 [13], which are available in the open-source community). In addition, to further exploit the potential of CBNet, we propose an effective training strategy with supervision for assisting backbones, achieving higher detection accuracy while sacrificing no inference speed. In particular, we propose a pruning strategy to reduce the model complexity while not sacrificing accuracy.

We present two versions of CBNet. The first, named CBNetV1 [1], connects only the adjacent stages of parallel backbones, providing a simple implementation of our composite backbone that is easy to follow. The other one, CBNetV2, combines the dense higher-level composition strategy, the auxiliary supervision, and a special pruning strategy, to fully explore the potential of CBNet for object detection. We empirically demonstrate the superiority of CBNetV2 over CBNetV1.

We demonstrate the effectiveness of our framework by conducting experiments on the challenging MS COCO benchmark [19]. Experiments show that CBNet has strong generalization capabilities for different backbones and head designs of the detector architecture, which enables us to train detectors that significantly outperform detectors based on larger backbones. Specifically, CBNet can be applied to various backbones, from convolution-based [13], [14], [15] to Transformer-based [20]. Compared to the original backbones, CBNet boosts their performances by 3.4%~3.5% AP, demonstrating the effectiveness of the proposed CBNet. At comparable model complexity, our CBNet still improves by 1.1% ~ 2.1% AP, indicating that the composed backbone is more efficient than the pre-trained wider and deeper networks. Moreover, CBNet can be flexibly plugged into mainstream detectors (e.g., RetinaNet [6], ATSS [7], Faster R-CNN [5], Mask R-CNN [8], Cascade R-CNN and Cascade Mask R-CNN [9]), and consistently improve the performances of these detectors by 3%~3.8% AP, demonstrating its strong adaptability to various head designs of detectors. Besides, CBNet is compatible with feature enhancing networks [21], [22] and model ensemble method [23]. Remarkably, it presents a general and resource-friendly framework to drive the accuracy ceiling of high-

performance detectors. Without bells and whistles, our CB-Swin-L achieves unparalleled single-model single-scale result of 59.4% box AP and 51.6% mask AP on COCO *test-dev*, surpassing the state-of-the-art result (i.e., 57.7% box AP and 50.2% mask AP obtained by Swin-L), while reducing the training schedule by 6 \times . With multi-scale testing, we push the current best single-model result to a new record of 60.1% box AP and 52.3% mask AP.

The main contributions of this paper are listed as follows:

- We propose a general, efficient and effective framework, CBNet (Composite Backbone Network), to construct high-performance backbone networks for object detection without additional pre-training.
- We propose a Dense Higher-Level Composition (DHLC) strategy, auxiliary supervision, and a pruning strategy to efficiently use existing pre-trained weights for object detection under the pre-training fine-tuning paradigm.
- Our CB-Swin-L achieves a new record of single-model single-scale result on COCO at a shorter (by 6 \times) training schedule than Swin-L. With multi-scale testing, our method achieves the best-known result without extra training data.

II. RELATED WORK

A. Object Detection

Object detection aims to locate each object instance from a predefined set of classes in an input image. With the rapid development of convolutional neural networks (CNNs), there is a popular paradigm for deep learning-based object detectors: the backbone network (typically designed for classification and pre-trained on ImageNet) extracts basic features from the input image, and then the neck (e.g., feature pyramid network [25]) enhances the multi-scale features from the backbone, after which the detection head predicts the object bounding boxes with position and classification information. Based on detection heads, the cutting-edge methods for generic object detection can be briefly categorized into two major branches. The first branch contains one-stage detectors such as YOLO [4], SSD [3], RetinaNet [6], NAS-FPN [26], EfficientDet [27], and [28]. The other branch contains two-stage methods such as Faster R-CNN [5], FPN [25], Mask

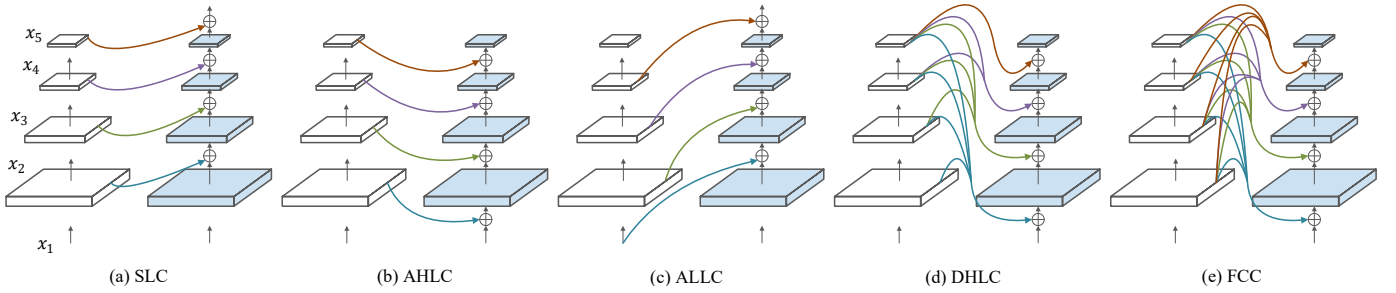


Fig. 2: Five kinds of composite strategies for Composite Backbone architecture when $K = 2$. (a) Same Level Composition (SLC). (b) Adjacent Higher-Level Composition (AHLC). (c) Adjacent Lower-Level Composition (ALLC). (d) Dense Higher-Level Composition (DHLC). (e) Full-connected Composition (FCC). The composite connections are colored lines representing some operations such as element-wise operation, scaling, 1×1 Conv layer, and BN layer.

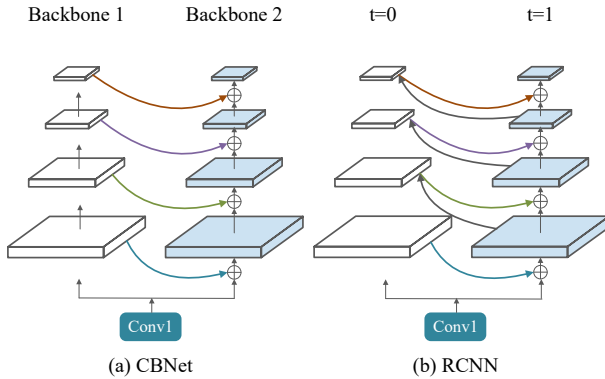


Fig. 3: Comparison between our proposed CBNet architecture ($K = 2$) and the unrolled architecture of RCNN [$T = 2$].

R-CNN [8], Cascade R-CNN [9], and Libra R-CNN [29]. Recently, academic attention has been geared toward anchor-free detectors due partly to the emergence of FPN [25] and focal Loss [6], where more elegant end-to-end detectors have been proposed. On the one hand, FSAF [30], FCOS [31], ATSS [7] and GFL [32] improve RetinaNet with center-based anchor-free methods. On the other hand, CornerNet [33], CenterNet [34], and FoveaBox [35] detect object bounding boxes with a keypoint-based method. In addition to the above CNN-based detectors, Transformer [16] has also been utilized for detection. DETR [36] proposes a fully end-to-end detector by combining CNN and Transformer encoder-decoders.

More recently, Neural Architecture Search (NAS) is applied to automatically search the architecture for a specific detector. NAS-FPN [26], NAS-FCOS [37] and SpineNet [38] use reinforcement learning to control the architecture sampling and obtain promising results. SM-NAS [39] uses the evolutionary algorithm and partial order pruning method to search the optimal combination of different parts of the detector. Auto-FPN [40] uses the gradient-based method to search for the best detector. OPANAS [41] uses the one-shot method to search for an efficient neck for object detection.

B. Backbones for Object Detection

Starting from AlexNet [2], deeper and wider backbones have been exploited by mainstream detectors, such as VGG [42], ResNet [13], DenseNet [43], ResNeXt [14], and Res2Net

[15]. Since the backbone network is usually designed for classification, whether it is pre-trained on ImageNet and fine-tuned on a given detection dataset or trained from scratch on the detection dataset, it requires many computational resources and is difficult to optimize. Recently, two non-trivially designed backbones, *i.e.*, DetNet [44] and FishNet [45], are specifically designed for the detection task. However, they still require pre-training for the classification task before fine-tuning for the detection task. Res2Net [15] achieves impressive results in object detection by representing multi-scale features at the granular level. HRNet [21] maintains high-resolution representations and achieves promising results in human pose estimation, semantic segmentation, and object detection. In addition to manually designing the backbone architecture, DetNAS [46] and Joint-DetNAS [47] use NAS to search for a better backbone for object detection, thereby reducing the cost of manual design. Swin Transformer [20] and PVT [17] utilize Transformer modular to build the backbone and achieve impressive results, despite the need for expensive pre-training.

It is well known that designing and pre-training a new and robust backbone requires significant computational costs. Alternatively, we propose a more economical and efficient solution to build a more powerful object detection backbone, by grouping multiple identical existing backbones (*e.g.*, ResNet [13], ResNeXt [14], Res2Net [15], HRNet [21], and Swin Transformer [20]).

C. Recurrent Convolution Neural Network

Different from the feed-forward architecture of CNN, Recurrent CNN (RCNN) [24] incorporates recurrent connections into each convolution layer to enhance the contextual information integration ability of the model. As shown in Fig. 3, our proposed Composite Backbone Network shares some similarities with the unfolded RCNN [24], but they are very different. First, the connections between the parallel stages in CBNet are unidirectional, while they are bidirectional in RCNN. Second, in RCNN, the parallel stages at different time steps share parameter weights, while in the proposed CBNet, the parallel stages of backbones are independent of each other. Moreover, we need to pre-train RCNN on ImageNet if we use it as the backbone of the detector. By contrast, CBNet does not require additional pre-training because it directly uses existing

pre-trained weights.

D. Model Ensemble

It is well known that a combination of many different predictors can lead to more accurate predictions, *e.g.*, ensemble methods are considered as the state-of-the-art solution for many machine learning challenges. The model ensemble improves the prediction performance of a single model by training multiple different models and combining their prediction results through post processing [48], [49].

There are two key characteristics for model ensemble: *model diversity* and *voting*. Model diversity means that the models with different architectures or training techniques are trained separately, and its importance for the model ensemble is well established [50], [51], [52], [53]. Most ensemble methods need voting strategies to compare the outputs of different models and refine the final predictions [23]. In terms of the above two characteristics, our CBNet is very different from the model ensemble. In fact, CBNet benefits from the *identical* backbones grouping, the recurrent style feature enhancing by *jointly* training. Furthermore, the output of the lead backbone is used directly for the final prediction without the need to be assembled with other backbones. More practical analysis can be found in Sec. IV-E2.

In practice, leading approaches to the challenge object detection benchmarks like MS COCO [19] or OpenImage [54] are based on the usage of model ensemble [55], [56], [57], [58], [59], [60]. For example, [60] separately trains 28 models of different architectures, heads, data splits, class sampling strategies, augmentation strategies and supervisions and aggregate these detector's outputs by ensembling method. [23] proposes the Probabilistic Ranking Aware Ensemble (PRAE) that refines the confidence of bounding boxes from different detectors. Our CBNet is compatible with such model ensemble methods, as are other conventional backbones. More details can be found in Sec. IV-F6.

E. Our Approach

Our network groups multiple identical backbones in parallel. It integrates the high- and low-level features of multiple identical backbones and gradually expands the receptive field to more efficiently perform object detection. This paper represents a very substantial extension of our previous conference paper [1] with results under recently developed start-of-the-art object detection frameworks. The main technical novelties compared with [1] lie in three aspects. (1) We extend the network (named as CBNetV1) proposed in [1], with three modifications: a specialized training method, a better composite strategy and a pruning strategy, which respectively optimizes the training process, more efficiently enhances feature representation and reduces the model complexity of CBNetV2. (2) We show the strong generalization capabilities of CBNetV2 for various backbones and head designs of detector architecture. (3) We show the superiority of CBNetV2 over CBNetV1 and present the state-of-art result of CBNetV2 in object detection.

III. PROPOSED METHOD

This section elaborates the proposed CBNet in details. In Sec. III-A and Sec. III-B, we describe its basic architecture and variants, respectively. In Sec. III-C, we propose a training

strategy for CBNet-based detectors. In Sec. III-D, we briefly introduce the pruning strategy. In Sec. III-E, we summarize the detection framework of CBNet.

A. Architecture of CBNet

The proposed CBNet consists of K identical backbones ($K \geq 2$). In particular, we call the case $K = n$ as CB-Backbone- Kn , where '- Kn ' is omitted when $K = 2$.

As in Fig. 1, the CBNet architecture includes two types of backbones: lead backbone B_K and assisting backbones B_1, B_2, \dots, B_{K-1} . Each backbone comprises L stages (usually $L = 5$), and each stage consists of several convolutional layers with feature maps of the same size. The l -th stage of the backbone implements the non-linear transformation $F^l(\cdot)$ ($l = 1, 2, \dots, L$).

Most conventional convolutional networks follow the design of encoding the input images into intermediate features with monotonically decreased resolution. In particular, the l -th stage takes the output (denoted as x^{l-1}) of the previous $(l-1)$ -th stage as input, which can be expressed as follows:

$$x^l = F^l(x^{l-1}), l \geq 2. \quad (1)$$

Differently, we adopt assisting backbones B_1, B_2, \dots, B_{K-1} to improve the representative ability of lead backbone B_K . We iterate the features of a backbone to its successor in a stage-by-stage fashion. Thus, Equation (1) can be rewritten as:

$$x_k^l = F_k^l(x_k^{l-1} + g^{l-1}(x_{k-1})), \quad l \geq 2, k = 2, 3, \dots, K, \quad (2)$$

where $g^{l-1}(\cdot)$ represents the composite connection, which takes features (denoted as $x_{k-1} = \{x_{k-1}^i | i = 1, 2, \dots, L\}$) from assisting backbone B_{k-1} as input and takes the features of the same size as x_k^{l-1} as output. Therefore, the output features of B_{k-1} are transformed and contribute to the input of each stage in B_k . Note that $x_1^1, x_2^1, \dots, x_K^1$ are weight sharing.

For the object detection task, only the output features of the lead backbone $\{x_K^i, i = 2, 3, \dots, L\}$ are fed into the neck and then the RPN/detection head, while the outputs of the assisting backbone are forwarded to its succeeding siblings. It is worth noting that B_1, B_2, \dots, B_{K-1} can be used for various backbone architectures (*e.g.*, ResNet [13], ResNeXt [14], Res2Net [15], and Swin Transformer [20]) and initialized directly from the pre-trained weights of a single backbone.

B. Possible Composite Strategies

For composite connection $g^l(x)$ which takes $x = \{x^i | i = 1, 2, \dots, L\}$ from an assisting backbone as input and outputs a feature of the same size of x^l (omitting k for simplicity), we propose the following five different composite strategies.

1) *Same Level Composition (SLC)*: An intuitive and simple way of compositing is to fuse the output features from the same stage of backbones. As shown in Fig. 2.a, the operation of SLC can be formulated as:

$$g^l(x) = \mathbf{w}(x^l), l \geq 2, \quad (3)$$

where \mathbf{w} represents a 1×1 convolution layer and a batch normalization layer.

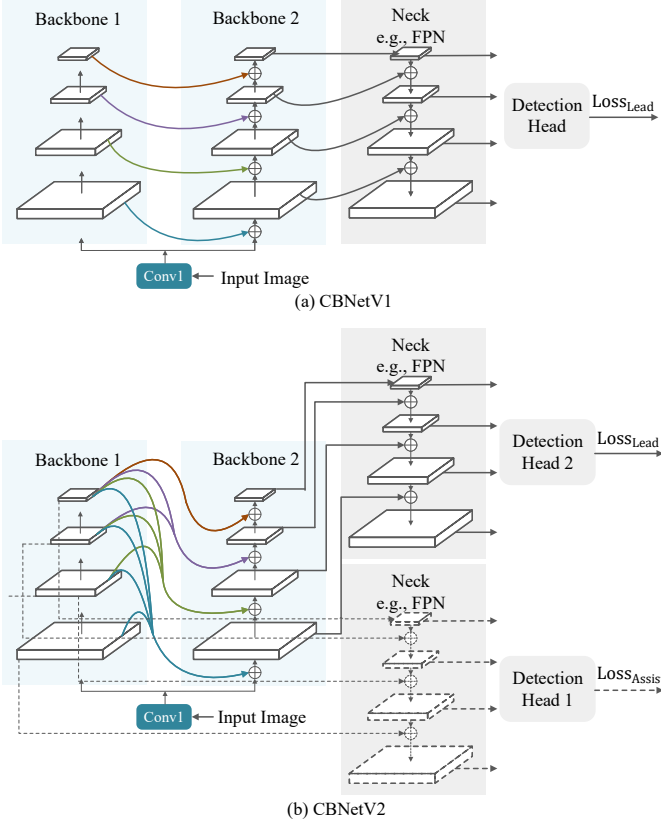


Fig. 4: (a) CBNetV1 [1] ($K = 2$). (b) CBNetV2 ($K = 2$) with auxiliary supervision. The two FPNs and detection heads share the same weights. It is worth noting that the main differences between CBNetV2 and CBNetV1 are in composite strategies and training strategies.

2) *Adjacent Higher-Level Composition (AHLC)*: Motivated by Feature Pyramid Networks [25], the top-down pathway introduces the spatially coarser, but semantically stronger, higher-level features to enhance the lower-level features of the bottom-up pathway, we introduce AHLC to feed the output of the adjacent higher-level stage of the previous backbone to the subsequent one (from left to right in Fig. 2.b):

$$g^l(x) = \mathbf{U}(\mathbf{w}(x^{l+1})), l \geq 1, \quad (4)$$

where $\mathbf{U}(\cdot)$ indicates the up-sampling operation.

3) *Adjacent Lower-Level Composition (ALLC)*: Contrary to AHLC, we introduce a bottom-up pathway to feed the output of the adjacent lower-level stage of the previous backbone to the succeeding one. We show ALLC in Fig. 2.c, which is formulated as:

$$g^l(x) = \mathbf{D}(\mathbf{w}(x^{l-1})), l \geq 2, \quad (5)$$

where $\mathbf{D}(\cdot)$ denotes the down-sample operation.

4) *Dense Higher-Level Composition (DHLC)*: In DenseNet [43], each layer is connected to all subsequent layers to build comprehensive features. Inspired by this,

we utilize dense composite connections in our CBNet architecture. The operation of DHLC is expressed as follows:

$$g^l(x) = \sum_{i=l+1}^L \mathbf{U}(\mathbf{w}_i(x^i)), l \geq 1. \quad (6)$$

As shown in Fig. 2.d, when $K = 2$, we compose the features from all the higher-level stages in the previous backbone and add them to the lower-level stages in the latter one.

5) *Full-connected Composition (FCC)*: As shown in Fig. 2.e, we compose features from all the stages in the previous backbones and feed them to each stage in the following one. Compared to DHLC, we add connections in low-high-level case. The operation of FCC can be expressed as:

$$g^l(x) = \sum_{i=2}^L \mathbf{I}(\mathbf{w}_i(x^i)), l \geq 1, \quad (7)$$

where $\mathbf{I}(\cdot)$ denotes scale-resizing, $\mathbf{I}(\cdot) = \mathbf{D}(\cdot)$ when $i > l$, and $\mathbf{I}(\cdot) = \mathbf{U}(\cdot)$ when $i < l$.

C. Auxiliary Supervision

Although increasing the depth usually leads to performance improvement [13], it may introduce additional optimization difficulties, as in the case of image classification [61]. The studies in [62], [63] introduce the auxiliary classifiers of intermediate layers to improve the convergence of very deep networks. In original CBNet, although the composite backbones are parallel, the latter backbone (e.g., lead backbone in Fig. 4.a) deepen the network through adjacent connections between the previous backbone (e.g., assisting backbone in Fig. 4.a). To better train the CBNet-based detector, We propose to generate initial results of assisting backbones by supervision with the auxiliary neck and detection head to provide additional regularization.

An example of our supervised CBNet when $K=2$ is illustrated in Fig. 4.b. Apart from the original loss that uses the lead backbone feature to train the detection head 1, another detection head 2 takes assisting backbone features as input, producing *auxiliary supervision*. Note that detection head 1 and detection head 2 are weight sharing, as are the two necks. The auxiliary supervision helps to optimize the learning process, while the original loss for the lead backbone takes the greatest responsibility. We add weights to balance the auxiliary supervision, where the total loss is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{Lead}} + \sum_{i=1}^{K-1} (\lambda_i \cdot \mathcal{L}_{\text{Assist}}^i). \quad (8)$$

where $\mathcal{L}_{\text{Lead}}$ is the loss of lead backbone, $\mathcal{L}_{\text{Assist}}$ is the loss of assisting backbones, and λ_i is the loss weight for the i -th assisting backbone.

During the inference phase, we abandon the auxiliary supervision branch and only utilize the output features of the lead backbone in CBNet (Fig. 4.b). Consequently, auxiliary supervision does not affect the inference speed.

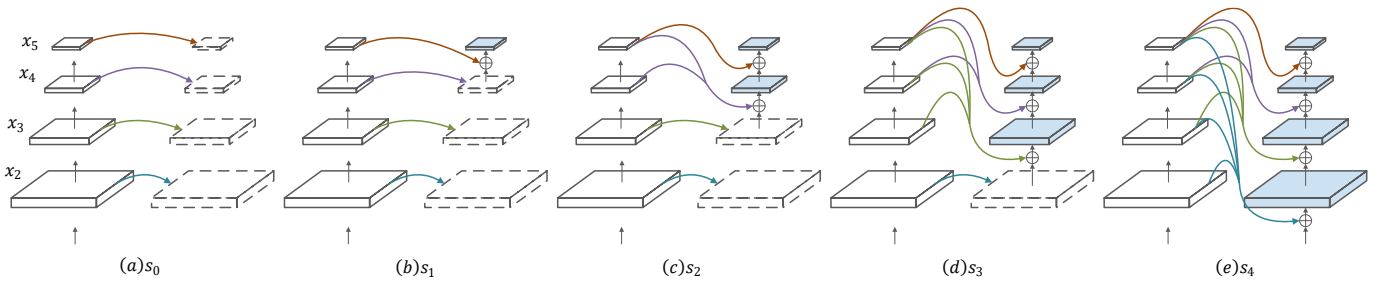


Fig. 5: Illustration of different pruning strategies for composite backbone when $K = 2$. s_i indicates there are i stages $\{x_j | j \geq 6 - i \text{ and } j \leq 5, i = 0, 1, 2, 3, 4\}$ in the 2, 3, ..., K -th backbone and the pruned stages are filled by the features of the same stages in the first backbone.

D. Pruning Strategy for CBNet

To reduce the model complexity of CBNet, we explore the possibility of pruning the different number of stages in 2, 3, ..., K -th backbones instead of composing the backbones in a holistic manner. For simplicity, we show five pruning methods when $K = 2$ in Fig. 5. s_i indicates there are i stages $\{x_j | j \geq 6 - i \text{ and } j \leq 5, i = 0, 1, 2, 3, 4\}$ in the 2, 3, ..., K -th backbone and the pruned stages are filled by the features of the same stages in the first backbone.

E. Architecture of Detection Network with CBNet

CBNet can be applied to various off-the-shelf detectors without additional modifications to network architectures. In practice, we attach the lead backbone with functional networks, e.g., FPN [25] and detection head. The inference phase of CBNet is shown in Fig. 1. Note that we present two versions of CBNet. The first one, named CBNetV1 [1], uses only AHLIC composition strategy, providing a simple implementation of the composite backbone that is easy to follow. The other one, CBNetV2, combines DHLIC composition strategy, the auxiliary supervision, and a special pruning strategy, to fully explore the potential of CBNet for object detection. We empirically demonstrate the superior of CBNetV2 over CBNetV1 in the following Sec. IV. In this paper, CBNet denotes CBNetV2 in the following experiments if not specified.

IV. EXPERIMENTS

In this section, we evaluate our CBNet through extensive experiments. In Sec. IV-A, we detail the experimental setup. In Sec. IV-B, we compare CBNet with state-of-the-art detection methods. In Sec. IV-C, we demonstrate the generality of our method over different backbones and detectors. In Sec. IV-E, we show the compatibility of CBNet with DCN and model ensemble. In Sec. IV-F, we conduct an extensive ablation study to investigate individual components of our framework.

A. Implementation details

1) *Datasets and Evaluation Criteria*: We conduct experiments on the COCO [19] benchmark. The training is conducted on the 118k training images, and ablation studies on the 5k `minival` images. We also report the results on the 20k images in `test-dev` for comparison with the state-of-the-art

(SOTA) methods. For evaluation, we adopt the metrics from the COCO detection evaluation criteria, including the mean Average Precision (AP) across IoU thresholds ranging from 0.5 to 0.95 at different scales.

2) *Training and Inference Details*: Our experiments are based on the open-source detection toolbox MMDetection [71]. For ablation studies and simple comparisons, we resize the input size to 800×500 during training and inference if not specified. We choose Faster R-CNN (ResNet50 [13]) with FPN [25] as the baseline. We use the SGD optimizer with an initial learning rate of 0.02, the momentum of 0.9, and 10^{-4} as weight decay. We train detectors for 12 epochs with a learning rate decreased by $10 \times$ at epoch 8 and 11. We use only random flip for data augmentation and set the batch size to 16. Note that experiments related to special backbones, e.g., Swin Transformer [20], [72], HRNet [21], PVT [17], PVTv2 [73], and DetectorRS [74], are not highlighted specifically following the hyper-parameters of the original papers. The inference speed FPS (frames per second) for the detector is measured on a machine with 1 V100 GPU.

To compare with state-of-the-art detectors, we utilize multi-scale training [75] (the short side resized to $400 \sim 1400$ and the long side is at most 1600) and a longer training schedule (details can be found in Sec. IV-B). During the inference phase, we use Soft-NMS [76] with a threshold of 0.001, and the input size is set to 1600×1400 . All other hyper-parameters in this paper follow MMDetection if not specified.

B. Comparison with State-of-the-Art

We compare our methods with cutting-edge detectors. We divide the results into object detection (Table I) and instance segmentation (Table II) according to whether or not the instance segmentation annotations are used during training. Following [20], we improve the detector heads of Cascade R-CNN, Cascade Mask R-CNN, and HTC in the above two tables by adding four convolution layers [77] in each bounding box head and using GIoU loss [78] instead of Smooth L1 [79].

1) *Object Detection*: For detectors trained with only bounding box annotations, we summarize them into two categories: anchor-based, and anchor-free-based in Table I. We select ATSS [7] as the anchor-free representative, and Cascade R-CNN as the anchor-based representative.

Anchor-free. CB-Res2Net101-DCN equipped with ATSS is trained for 20 epochs, where the learning rate is decayed

TABLE I: Comparison with the state-of-the-art results on COCO *test-dev*. Our CB-Res2Net101-DCN achieves higher bbox AP over previous anchor-free and anchor-based detectors while using comparable or fewer training epochs.

| | Method | AP ^{box} | AP ₅₀ ^{box} | AP ₇₅ ^{box} | AP _S ^{box} | AP _M ^{box} | AP _L ^{box} | Params | Epochs |
|--------------------------|--|-------------------|---------------------------------|---------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------|-----------|
| <i>anchor-free-based</i> | FSAF [30] | 42.9 | 63.8 | 46.3 | 26.6 | 46.2 | 52.7 | 94 M | 24 |
| | FCOS [31] | 44.7 | 64.1 | 48.4 | 27.6 | 47.5 | 55.6 | 90 M | 24 |
| | DETR-DC5 [36] | 44.9 | 64.7 | 47.7 | 23.7 | 49.5 | 62.3 | 60 M | 500 |
| | NAS-FCOS [37] | 46.1 | - | - | - | - | - | 89 M | 24 |
| | ATSS [7] | 47.7 | 66.5 | 51.9 | 29.7 | 50.8 | 59.4 | 95 M | 24 |
| | GFL [32] | 48.2 | 67.4 | 52.6 | 29.2 | 51.7 | 60.2 | 53 M | 24 |
| | Deformable DETR [64] | 50.1 | 69.7 | 54.6 | 30.6 | 52.8 | 64.7 | - | 50 |
| | PAA [65] | 50.8 | 69.7 | 55.1 | 31.4 | 54.7 | 65.2 | 129 M | 24 |
| | CB-Res2Net101-DCN (ATSS) | 52.7 | 71.1 | 57.7 | 35.7 | 56.6 | 62.7 | 107 M | 20 |
| <i>anchor-based</i> | Auto-FPN [40] | 44.3 | - | - | - | - | - | 90 M | 24 |
| | SM-NAS:E5 [39] | 45.9 | 64.6 | 49.6 | 27.1 | 49.0 | 58.0 | - | 24 |
| | NAS-FPN [26] | 48.3 | - | - | - | - | - | - | 150 |
| | SP-NAS [66] | 49.1 | 67.1 | 53.5 | 31 | 52.6 | 63.7 | - | 50 |
| | Joint-DetNAS [47] | 50.7 | 69.6 | 55.4 | 31.3 | 53.8 | 64.0 | - | 16 |
| | SpineNet-190 [38] | 52.1 | 71.8 | 56.5 | 35.4 | 55 | 63.6 | 164 M | 250 |
| | OPANAS [41] | 52.2 | 71.3 | 57.3 | 33.3 | 55.6 | 65.4 | 83 M | 24 |
| | EfficientDet-D7x [27] | 55.1 | 74.3 | 59.9 | - | - | - | 77 M | 600 |
| | YOLOv4-P7 [67] | 55.5 | 73.4 | 60.8 | 38.4 | 59.4 | 67.7 | 288 M | 450 |
| | CB-Res2Net101-DCN (Cascade R-CNN) | 55.6 | 73.7 | 60.8 | 37.4 | 59.0 | 67.6 | 146 M | 32 |

TABLE II: Comparison with the state-of-the-art object detection and instance segmentation results on COCO. In collaboration with Swin Transformer, our CBNetV2 achieves the state-of-the-art box AP and mask AP while using fewer training epochs.

| Pre-trained on | Method | mini-val | | test-dev | | Params | Epochs |
|----------------|---------------------------------------|-------------------|--------------------|-------------------|--------------------|--------------|-----------|
| | | AP ^{box} | AP ^{mask} | AP ^{box} | AP ^{mask} | | |
| ImageNet-1K | GCNet* [68] | 51.8 | 44.7 | 52.3 | 45.4 | - | 36 |
| | Swin-B [20] | 51.9 | 45.0 | - | - | 145 M | 36 |
| | ResNeSt-200* [69] | 52.5 | - | 53.3 | 47.1 | - | 36 |
| | CopyPaste [70] [†] | 55.9 | 47.2 | 56.0 | 47.4 | 185 M | 96 |
| | CB-Swin-S (Cascade Mask R-CNN) | 56.3 | 48.6 | 56.9 | 49.1 | 156 M | 36 |
| ImageNet-22K | Swin-L (HTC++) [20] | 57.1 | 49.5 | 57.7 | 50.2 | 284 M | 72 |
| | Swin-L (HTC++) [20]* | 58.0 | 50.4 | 58.7 | 51.1 | 284 M | 72 |
| | CB-Swin-B (HTC) | 58.4 | 50.7 | 58.7 | 51.1 | 235 M | 20 |
| | CB-Swin-B (HTC)* | 58.9 | 51.3 | 59.3 | 51.8 | 235 M | 20 |
| | CB-Swin-L (HTC) | 59.1 | 51.0 | 59.4 | 51.6 | 453 M | 12 |
| | CB-Swin-L (HTC)* | 59.6 | 51.8 | 60.1 | 52.3 | 453 M | 12 |

¹ * indicates method with multi-scale testing.

² † indicates additional unlabeled data are used to pretrain backbone.

by 10× in the 16th and 19th epochs. Notably, our CB-Res2Net101-DCN achieves 52.8% AP, outperforming previous anchor-free methods [7], [30], [31], [32], [36], [37], [64] under single-scale testing protocol.

Anchor-based. Our CB-Res2Net101-DCN achieves 55.6% AP, surpassing other anchor-based detectors [26], [27], [38], [39], [40], [41], [66], [80]. It is worth noting that our CBNet trains only for 32 epochs (the first 20 epochs are regular training and the remaining 12 epochs are trained with Stochastic Weights Averaging [81]), being 16× and 12× shorter than EfficientDet and YOLOv4, respectively.

2) *Instance Segmentation:* We further compare our method with state-of-the-art results [20], [68], [69], [70] using both bounding box and instance segmentation annotations in Table II. Following [20], we provide results with the backbone pre-trained on regular ImageNet-1K and ImageNet-22K to show the high capacity of CBNet.

Results with regular ImageNet-1K pre-train. Following [20], 3x schedule (36 epochs with the learning rate decayed by 10× at epochs 27 and 33) is used for CB-Swin-S. Using Cascade Mask R-CNN, our CB-Swin-S achieves 56.3% box AP and 48.6% mask AP on COCO *minival* in terms of the bounding box and instance segmentation, showing significant gains of +4.4% box AP and +3.6% mask AP to Swin-B with similar model size and the same training protocol. In addition, CB-Swin-S achieves 56.9% box AP and 49.1% mask AP on COCO *dev*, outperforming other ImageNet-1K pre-trained backbone-based detectors.

Results with ImageNet-22K pre-train. Our CB-Swin-B achieves single-scale result of 58.4% box AP and 50.7% mask AP on COCO *minival*, which is 1.3% box AP and 1.2% mask AP higher than that of Swin-L (HTC++) [20] while the number of parameters is decreased by 17% and the training schedule is reduced by 3.6×. Especially, with only 12 epochs

TABLE III: Comparison between CBNet and conventional backbones in terms of different architectures. Our CBNet boosts the performance of CNN-based backbones by over 3.4% AP.

| Backbone | AP ^{box} | | Δ AP |
|-----------------|-------------------|-------------|-------------|
| | Origin | CB- | |
| ResNet50 | 34.6 | 38.0 | +3.4 |
| ResNeXt50-32x4d | 36.3 | 39.8 | +3.5 |
| Res2Net50 | 37.7 | 41.2 | +3.5 |

TABLE IV: Comparison between CB-Backbone and deeper and wider single backbones in terms of different architectures. The backbones in each group are sorted by FLOPs for efficiency comparison. Backbones armed with our proposed method are more efficient than their own wider and deeper version.

| Backbone | AP ^{box} | Params | FLOPs | FPS |
|---------------------------|-------------------|---------------|--------------|-------------|
| ResNet101 | 36.3 | 60.5 M | 121 G | 25.8 |
| CB-ResNet50 | 38.0 | 69.4 M | 121 G | 23.3 |
| ResNet152 | 37.8 | 76.2 M | 151 G | 21.3 |
| ResNeXt101-32x4d | 37.7 | 60.2 M | 122 G | 20.8 |
| CB-ResNeXt50-32x4d | 39.8 | 68.4 M | 123 G | 19.3 |
| ResNeXt101-64x4d | 38.7 | 99.3 M | 183 G | 15.8 |
| Res2Net101 | 40.1 | 61.2 M | 125 G | 20.7 |
| CB-Res2Net50 | 41.2 | 69.7 M | 125 G | 20.2 |
| Res2Net200 | 41.7 | 92.2 M | 185 G | 14.8 |

training (which is $6\times$ shorter than Swin-L), our CB-Swin-L achieves 59.4% box AP and 51.6% mask AP on COCO test-dev, outperforming prior arts. We can push the current best result to a new record of 60.1% box AP and 52.3% mask AP through multi-scale testing. The results demonstrate that CBNet proposes an efficient, effective, and resource-friendly framework to build high-performance detectors.

C. Generalization Capability of CBNet

CBNet expands the receptive field by combining the backbones in parallel rather than simply increasing the depth of the network. To demonstrate the generality of our design strategy, we perform experiments on various backbones and different head designs of the detector architecture.

1) Generality for Main-stream Backbone Architectures:

Effectiveness. To demonstrate the effectiveness of CBNet, we conduct experiments on Faster R-CNN with different backbone architectures. As shown in Table III, for CNN-based backbones (e.g., ResNet, ResNeXt-32x4d, and Res2Net), our method can boost baseline by over 3.4% AP.

Efficiency. Note that the number of parameters in CBNet has increased compared to the baseline. To better demonstrate the efficiency of the composite architecture, we compare CBNet with deeper and wider backbone networks. As shown in Table IV, with comparable number of number and inference speed, CBNet improves ResNet101, ResNeXt101-32x4d, Res2Net101 by 1.7%, 2.1%, and 1.1% AP, respectively. Additionally, CB-ResNeXt50-32x4d is 1.1% AP higher than that of ResNeXt101-64x4d, while the number of parameters is only 70%. The results demonstrate that our composite

TABLE V: Comparison between CB-Swin-T and the Swin equipped with Cascade Mask R-CNN. CBNet is more effective and efficient than the wider and deeper version of Swin-Transformer.

| Backbone | AP ^{box} | AP ^{mask} | Params | FLOPs | FPS |
|------------------|----------------------------|----------------------------|----------------|--------------|------------|
| Swin-T | 50.5 | 43.7 | 85.6 M | 742 G | 7.8 |
| Swin-S | 51.8 ^{+1.3} | 44.7 ^{+1.0} | 107.0 M | 832 G | 7.0 |
| Swin-B | 51.9 ^{+1.4} | 45.0 ^{+1.3} | 145.0 M | 975 G | 5.9 |
| CB-Swin-T | 53.6^{+3.1} | 46.2^{+2.5} | 113.8 M | 836 G | 6.5 |

TABLE VI: Generalization capability of CBNet over various special backbones.

| Detector | Backbone | mAP | Params | FLOPs |
|-------------|------------------------|-------------|---------------|---------------|
| YOLOV3 | MobileNetV2 | 22.2 | 3.74 M | 1.69 G |
| | CB-MobileNetV2 | 25.4 | 5.20 M | 2.27 G |
| | MobileNetV2 (1.4x) | 24.4 | 5.04 M | 2.21 G |
| Faster-RCNN | HRNetv2p_w32 | 40.2 | 47.3 M | 285 G |
| | CB-HRNetv2p_w32 | 42.6 | 76.8 M | 449 G |
| | HRNetv2p_w48 | 42.0 | 83.4 M | 460 G |
| RetinaNet | PVT-Small | 40.4 | 34.2 M | 214 G |
| | CB-PVT-Small | 43.4 | 58.9 M | 278 G |
| | PVT-Large | 42.6 | 71.1 M | 309 G |
| | PVTv2-B2 | 44.6 | 35.1 M | 218 G |
| | CB-PVTv2-B2 | 47.7 | 60.7 M | 287 G |
| | PVTv2-B5 | 46.1 | 91.7 M | 335 G |

backbone architecture is more efficient and effective than simply increasing the depth and width of the network.

2) *Generality for Swin Transformer:* Transformer is notable for the use of attention to model long-range dependencies in data, and Swin Transformer [20] is one of the most representative recent arts. We conduct experiments on Swin Transformer to show the model generality of CBNet. For a fair comparison, we follow the same training strategy as [20] with multi-scale training (the short side resized to $480 \sim 800$ and the long side at most 1333), AdamW optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs). As shown in Table V, the accuracy of the model slowly increases as the Swin Transformer is deepened and widened, and saturates at Swin-S. Swin-B is only 0.1% AP higher than that of Swin-S, but the amount of parameters increases by 38M. When using CB-Swin-T, we achieve 53.6% box AP and 46.2% mask AP by improving Swin-T 3.1% box AP and 2.5% mask AP. Surprisingly, our CB-Swin-T is 1.7% box AP and 1.2% mask AP higher than that of the deeper and wider Swin-B while the model complexity is lower (e.g., FLOPs 836G vs. 975G, Params 113.8M vs. 145.0M, FPS 6.5 vs. 5.9). These results prove that CBNet can also improve non-pure convolutional architectures. They also demonstrate that CBNet pushes the upper limit of accuracy for high-performance detectors more effectively than simply increasing the depth and width of the network.

3) *Generality for special backbones:* To further show the generality of CBNet for various backbones, we conduct experiments on CBNet equipped with different backbones including MobileNetV2 [72], HRNet [21], PVT [17], and PVTv2 [73].

TABLE VII: Comparison of ResNet50 and our CB-ResNet50. CB-ResNet50 significantly boosts all popular object detectors by 3.0% ~ 3.8% bbox AP and 2.9% mask AP. 'R50' is short for 'ResNet50'.

| Detector | AP ^{box} | | Δ AP | AP ^{mask} | | Δ AP |
|---------------|-------------------|-------------|-------------|--------------------|-------------|-------------|
| | R50 | CB-R50 | | R50 | CB-R50 | |
| RetinaNet | 33.2 | 36.2 | +3.0 | - | - | - |
| ATSS | 36.9 | 39.9 | +3.0 | - | - | - |
| Faster R-CNN | 34.6 | 38.0 | +3.4 | - | - | - |
| Mask R-CNN | 35.2 | 39.0 | +3.8 | 31.8 | 34.7 | +2.9 |
| Cascade R-CNN | 38.2 | 41.2 | +3.0 | - | - | - |

TABLE VIII: Comparison of CBNet with YOLOX and Joint-DetNAS.

| Backbone | Method | mAP | Params | FLOPs |
|--------------------------|--------------|-------------|---------------|--------------|
| CSPNet-L | | 49.4 | 54.2 M | 78 G |
| CSPNet-X | YOLOX | 50.9 | 99.1 M | 141 G |
| CB-CSPNet-L | | 52.0 | 83.8 M | 118 G |
| Searched X101-FPN | Joint-DetNAS | 45.7 | - | 266 G |

For a fair comparison, we choose the publicly available pre-trained backbones and all experiment settings (*e.g.*, choice of detectors, training, and inference details) are following their settings in MMDetection [82]. Results are shown in Table VI. For Mobile settings with YOLOV3 [83], our CB-MobileNetV2 improves MobileNetV2 by 3.1% AP and is 1% AP higher than MobileNetV2(1.4x) with comparable model complexity. For backbones with high-resolution representations, our CB-HRNetv2p_w32 improves HRNetv2p_w32 by 2.4% AP and is 0.6% AP higher than HRNetv2p_w48 with less model complexity. For global transformer backbones, we choose RetinaNet as detector follow the original paper [17], [73]. Our CB-PVT-Small improves PVT-Small by 3% AP and is 0.8% AP higher than PVT-Large with only 83% number of parameters. Furthermore, our CB-PVTv2-B2 improves PVTv2-B2 by 3.1% AP and is 1.6% AP higher than PVTv2-B5 with only 66% number of parameters. The results show that our CBNet improves a wide variety of backbones and achieves better accuracy under comparable or less parameters and FLOPs, which verify the effectiveness and efficiency of CBNet.

4) *Model Adaptability for Mainstream Detectors*: We evaluate the adaptability of CBNet by plugging it into mainstream detectors such as RetinaNet, ATSS, Faster R-CNN, Mask R-CNN, and Cascade R-CNN. These methods present a variety of detector head designs (*e.g.*, two-stage vs. one-stage, anchor-based vs. anchor-free). As shown in Table VII, our CBNet significantly boosts all popular object detectors by over 3% AP. The instance segmentation accuracy of Mask R-CNN is also improved by 2.9% AP. These results demonstrate the robust adaptability of CBNet to various head designs of detectors.

D. Comparison with Relevant Works.

There are several relevant detectors, such as DetecoRS [74] that composites both backbone and FPN and Joint-DetNAS [47] searches for the model scaling strategy. We conduct comparisons between CBNet and these two methods.

TABLE IX: Comparison between CBNet and DetecoRS based on Faster-RCNN with 1 \times training. CBNet achieves on par or better accuracy-efficiency trade-offs.

| Backbone | Method | mAP | Params | FLOPs |
|------------------|--------------|-------------|----------------|--------------|
| ResNet101 | CBNet | 42.7 | 107.4 M | 436 G |
| | DetecoRS | 42.8 | 104.0 M | 512 G |
| ResNeXt101-32x4d | CBNet | 44.2 | 106.7 M | 444 G |
| | DetecoRS | 44.0 | 103.5 M | 519 G |
| Res2Net101 | CBNet | 45.8 | 108.7 M | 452 G |
| | DetecoRS | 45.7 | 105.4 M | 533 G |
| Swin-Tiny | CBNet | 46.7 | 74.1 M | 307 G |
| | DetecoRS | 45.9 | 73.4 M | 366 G |

Joint-DetNAS [47] integrates neural architecture search (NAS), pruning and knowledge distillation for optimizing detectors. Similarly, our CBNet also uses pruning strategy but focus more on scaling backbones using composite strategy. Thanks to the strong generalization ability, our CBNet can boost the performance of advanced high-performance detectors (*e.g.*, YOLOX [84]). As shown in Table VIII, our CB-CSPNet-L improves CSPNet-L [84] by 2.6% AP and is 1.1% AP higher than CSPNet-X with only 85% number of parameters. We further compare our CBNet using an existing hand-designed detector (*i.e.*, YOLOX) with Joint-DetNAS which uses an advanced knowledge distillation training strategy. Our CBNet achieves 52% AP with 118 GFLOPs, superior to that of Joint-DetNAS (X101-FPN based) at 45.7% AP with 266 GFLOPs. Note that it is hard to have a fair comparison because our CBNet focuses on the architecture design of the backbone while Joint-DetNAS focuses on the joint optimization of the architecture and training for the entire detector.

DetecoRS [74] conducts a similar design as CBNet while DetecoRS composites both backbone and FPN. We compare CBNetV2 and DetecoRS with different backbones on Faster R-CNN in Table IX. Under the same training strategy of DetecoRS with 1333 \times 800 as input size, CBNet achieves comparable or higher AP with fewer FLOPs. Specifically, with advanced backbone Swin-Tiny, our CBNet outperforms DetecoRS by 0.8% AP with only 84% FLOPs.

E. Compatibility of CBNet

1) *Compatibility with Deformable Convolution*: Deformable convolution [22] enhances the transformation modeling capability of CNNs and is widely used for accurate object detectors (*e.g.*, simply adding DCN improves Faster R-CNN ResNet50 from 34.6% to 37.4% AP). To show the compatibility of CBNet architecture with deformable convolution, we perform experiments on ResNet and ResNeXt equipped with Faster R-CNN. As shown in Table X, DCN is still effective on CBNet with 2.3% AP~2.7% AP improvement. This improvement is greater than the 2.0% AP and 1.3% AP increments on ResNet152 and ResNeXt101-64x4d. On the other hand, CB-ResNet50-DCN increases the AP of ResNet50-DCN and the deeper ResNet152-DCN by 3.0% and 0.6%, respectively. In addition, CB-ResNet50-32x4d-DCN increases the AP of ResNet50-32x4d-DCN and the deeper and wider ResNeXt101-64x4d-DCN by 3.7% and 1.3%, respectively.

TABLE X: Experimental results on the compatibility of CBNet and deformable convolution. CBNet and deformable convolution can be superimposed on each other without conflict.

| Backbone | AP ^{box} | | Δ AP |
|--------------------------|-------------------|-------------|-------------|
| | w/o DCN | w/ DCN | |
| ResNet50 | 34.6 | 37.4 | +2.8 |
| ResNet152 | 37.8 | 39.8 | +2.0 |
| CB-ResNet50 | 38.0 | 40.4 | +2.4 |
| ResNeXt50-32x4d | 36.3 | 38.6 | +2.3 |
| ResNeXt101-64x4 | 38.7 | 41.0 | +2.3 |
| CB-ResNeXt50-32x4 | 39.8 | 42.5 | +2.7 |

TABLE XI: Compatibility of CBNet and Model Ensemble Method. They can be superimposed on each other without conflict. 'R50', 'X50', 'R2-50', 'R101', 'X101', 'R2-101' are short for Faster R-CNN equipped with ResNet50, ResNeXt50-32x4d, Res2Net50, ResNet101, ResNeXt101-64x4, Res2Net101 respectively. 'Single' denotes single model best AP.

| Ensemble Models | AP ^{box} | | Δ AP |
|--|-------------------|-------------|-------------|
| | Single | Ensem. | |
| R50(34.6) & X50(36.3) | 36.3 | 37.3 | 1.0 |
| R101(36.3) & X101(38.7) | 38.7 | 39.7 | 1.0 |
| CB-R50(38.0) & CB-X50(39.8) | 39.8 | 40.8 | 1.0 |
| R50(34.6) & R2-50(37.7) | 37.7 | 38.8 | 1.1 |
| R101(36.3) & R2-101(40.1) | 40.1 | 40.9 | 0.8 |
| CB-R50(38.0) & CB-R2-50(41.2) | 41.2 | 42.2 | 1.0 |
| X50(36.3) & R2-50(37.7) | 37.7 | 39.4 | 1.7 |
| X101(38.7) & R2-101(40.1) | 40.1 | 41.9 | 1.8 |
| CB-X50(39.8) & CB-R2-50(41.2) | 41.2 | 42.9 | 1.7 |

The results show that the effects of CBNet and deformable convolution can be superimposed without conflicting with each other.

2) *Compatibility with Model Ensemble*: The model ensemble improves the prediction performance of a single model by training multiple different models and combining their prediction results through post-processing [48], [49]. Probabilistic Ranking Aware Ensemble (PRAE) [23] refines the confidence of bounding boxes from different detectors and outperforms other ensemble learning methods for object detection by significant margins (e.g., assembling Faster R-CNN ResNet50 and Faster R-CNN ResNeXt50 improves the single model best AP from 36.3% to 37.3%). Note that assembling two same detectors (i.e., two Faster R-CNN ResNeXt50) does not improve the performance (same as the single detector 36.3% AP). To show the compatibility of our CBNet architecture with the model ensemble method PRAE, we perform experiments on traditional backbones (i.e., ResNet, ResNeXt, Res2Net) and their Composite Backbones equipped with Faster R-CNN. As shown in Table XI, PRAE is still effective for assembling detectors with CBNet, with 0.8% \sim 1.7% AP improvement, which is consistent with the case of assembling detectors with traditional backbones. In addition, CBNet is more effective than the model ensembling method PRAE, e.g., Faster R-CNN CB-R50 achieves 38.0% AP, superior to the 37.3% AP of assembling Faster R-CNN ResNet50 and Faster R-CNN ResNeXt50. The results show that the effects of CBNet and

TABLE XII: Comparison between different composite strategies. Obviously, DHLC achieves the best FLOPs-accuracy and Params-accuracy trade-offs.

| Composite Strategy | AP ^{box} | Params | FLOPs | FPS |
|--------------------|-------------------|--------|-------|------|
| - | 34.6 | 41.5 M | 90 G | 30.0 |
| SLC | 35.0 | 64.8 M | 123 G | 22.6 |
| AHLC | 36.0 | 67.6 M | 126 G | 22.4 |
| ALLC | 32.4 | 67.6 M | 133 G | 22.5 |
| DHLC | 37.3 | 69.7 M | 127 G | 21.4 |
| FCC | 37.4 | 72.0 M | 145 G | 19.9 |

TABLE XIII: Comparison between AHLC and DHLC on different backbones.

| Backbone | Composite | AP ^{box} | Params | FLOPs |
|----------------------|-------------|-------------------|---------|-------|
| CB-ResNet101 | AHLC | 37.9 | 105.6 M | 186 G |
| | DHLC | 38.7 | 107.6 M | 188 G |
| CB-ResNeXt101 | AHLC | 39.9 | 183.0 M | 312 G |
| | DHLC | 41.0 | 185.1 M | 313 G |

model ensemble can be superimposed without conflicting with each other, suggesting that the detector equipped with CBNet should be considered as a single detector/model despite having multiple identical backbones compositions.

F. Ablation Studies

We ablate various design choices for our proposed CBNet. For simplicity, all accuracy results here are on the COCO validation set with 800×500 input size if not specified.

1) *Effectiveness of Different Composite Strategies*: We conduct experiments to compare the proposed composite strategies in Fig. 2, including SLC, AHLC, ALLC, DHLC and FCC. All these experiments are conducted based on the Faster R-CNN CB-ResNet50 architecture. Results are shown in Table XII.

SLC gets a slightly improves accuracy of the single-backbone baseline (35% vs. 34.6% AP). The features extracted by the same stage of both backbones are similar, and thus SLC can only learn slightly more semantic information than a single backbone does.

AHLC raises the baseline by 1.4% AP, which verifies our motivation in Sec. III-B2, i.e., the semantic information higher-level features of the former backbone enhances the representation ability of the latter backbone.

ALLC degrades the performance of the baseline by 2.2% AP. We infer that directly adding the lower-level features of the assisting backbone to the higher-level ones of the lead backbone impair the representation ability of the latter.

DHLC improves the performance of the baseline by a large margin (from 34.6% AP to 37.3% AP by 2.7% AP). More composite connections of the high-low cases enrich the representation ability of features to some extent.

FCC achieves the best performance of 37.4% AP while being 7% slower than DHLC (19.9 vs. 21.4 FPS).

In summary, FCC and DHLC achieve the two best results. Considering the computational simplicity, we recommend using DHLC for CBNet. All the above composite strategies have a similar amount of parameters, but the accuracy varies greatly. The results prove that simply increasing the number of

TABLE XIV: Ablation study of loss weights for assistant supervision.

| Backbone | λ_1 | λ_2 | AP ^{bbox} |
|----------------|-------------|-------------|--------------------|
| | 0 | - | 37.3 |
| | 0.125 | - | 37.6 |
| CB-ResNet50 | 0.25 | - | 37.8 |
| | 0.5 | - | 38.1 |
| | 1.0 | - | 37.9 |
| | 0 | 0 | 37.4 |
| | 0.25 | 0.25 | 37.9 |
| CB-ResNet50-K3 | 0.25 | 0.5 | 38.9 |
| | 0.5 | 0.5 | 38.6 |
| | 0.5 | 1.0 | 39.2 |

parameters or adding a backbone network does not guarantee a better result while the composite connection plays a crucial part. These results show that the suggested DHLC composite strategy is effective and nontrivial.

Note that there is a minor performance gap between the original CBNetV1 in [1] and this paper for DHLC and AHLC. The reason is that CBNetV1 and this paper are performed under different deep learning platforms (CAFFE vs. PyTorch), which use different model initialization strategies and result in different model performances. We compare DHLC and AHLC on different backbones in Table XIII. DHLC outperforms AHLC by 0.8% AP and 1.1% AP on ResNet101 and ResNeXt101-64x4d, respectively, showing the generality of DHLC for different backbones.

We conduct a grid search by proxy task to search for better composite strategies. To reduce the search cost, we simplify the search space by only searching the connections including x_3, x_4, x_5 stages in composite backbones, and design a proxy task with 1/5 of the COCO training set with input size set to 800×500 . In this way, we only need to train $(2^3)^3 = 512$ detectors for 205 GPU days. The best-searched strategy is a simplified DHLC(s_3) without the connection between x_4 of the former backbone to the input of x_3 of the latter one. The searched strategy achieves 37.3% AP with 69.1 M, 126 GFLOPs, and performs un-par with our designed DHLC (37.3% AP with 69.7 M, 127 GFLOPs), further validating the necessity of high-to-low connections in our handcraft design.

2) *Weights for Auxiliary Supervision*: Experimental results related to weighting the auxiliary supervision are presented in Table XIV. For simplicity, we perform DHLC composite strategy on CBNet. The first setting is the Faster R-CNN CB-ResNet50 baseline and the second is the CB-ResNet50-K3 ($K = 3$ in CBNet) baseline, where the λ for assisting backbone in Equation (8) is set to zero. For the case $K = 2$, the baseline can be improved by 0.8% AP by setting λ_1 to 0.5. For the case $K = 3$, the baseline can be improved by 1.8% AP by setting $\{\lambda_1, \lambda_2\}$ to $\{0.5, 1.0\}$. The experimental results verify that the auxiliary supervision forms an effective training strategy that improves the performance of CBNet.

3) *Efficiency of Pruning Strategy*: As shown in Fig. 6a, with the pruning strategy, our CB-ResNet50 family and CB-ResNet50-K3 family achieve better FLOPs-accuracy trade-offs than ResNet family. This also illustrates the efficiency of our pruning strategy. In particular, the number of FLOPs in s_3 is

TABLE XV: Comparison between CBNetV1 [1] and CBNetV2. CBNetV2 is superior to CBNetV1 in terms of accuracy and complexity.

| Backbone | DHLC | Sup. | Prun. | AP ^{bbox} | Params | FLOPs | FPS |
|----------|------|------|-------|--------------------|--------|--------------|------|
| ResNet50 | | | | 34.6 | 41.5 M | 90 G | 30.0 |
| CBNetV1 | | | | 36.0 | 67.6 M | 126 G | 22.4 |
| CBNetV1 | | | ✓ | 35.6 | 66.2 M | 111 G | 26.6 |
| | | ✓ | | 36.9 | 67.6 M | 126 G | 22.4 |
| | ✓ | | | 37.3 | 69.7 M | 127 G | 21.4 |
| | ✓ | ✓ | | 38.1 | 69.7 M | 127 G | 21.4 |
| CBNetV2 | ✓ | ✓ | ✓ | 38.0 | 69.4 M | 121 G | 23.3 |

reduced by 10% compared to s_4 , but the accuracy is decreased by only 0.1%. This is because the weights of the pruned stage are fixed during the detector training [71] so pruning this stage does not sacrifice detection accuracy. Hence, when speed and memory cost need to be prioritized, we suggest pruning the fixed stages in 2, 3, ..., K -th backbones in CBNet.

4) *Number of Backbones in CBNet*: To further explore the ability to construct high-performance detectors of CBNet, we evaluate the efficiency of our CBNet by controlling the number of backbones. As shown in Fig. 6b, we vary the number of backbones (e.g., $K = 1, 2, 3, 4, 5$) and compare their accuracy and efficiency (GFLOPs) with the ResNet family. Note that the accuracy continues to increase as the complexity of the model increases. Compared with ResNet152, our method obtains higher accuracy at $K=2$ while computation cost is lower. Meanwhile, the accuracy can be further improved for $K=3, 4, 5$. CBNet provides an effective and efficient alternative to improve the model performance rather than simply increasing the depth or width of the backbone.

5) *Comparison of CBNetV1 and CBNetV2*: To fairly compare CBNetV1 [1] and CBNetV2, we progressively apply the DHLC composite strategy, auxiliary supervision, and pruning strategy to CBNetV1, where AHLC is the default composite strategy in Table XV. As in the 1st and 2nd rows of Table XV, the composite backbone structure CBNetV1 [1] improves the Faster R-CNN ResNet50 baseline by 1.4% AP. As in the 2nd and 3rd row, the accelerated version of CBNetV1 (s_2 pruning version in Fig. 5) improves the inference speed from 22.4 FPS to 26.6 FPS while decreasing the accuracy by 0.4% AP. As in the 2nd and 4th rows, the auxiliary supervision brings a 0.9% AP increment to CBNetV1, thanks to the better training strategy that improves the representative ability of the lead backbone. Note that the auxiliary supervision does not introduce extra parameters during the inference phase. As in the 2nd and 5th rows, DHLC composite strategy improves the detection performance of CBNetV1 by 1.3% AP with higher model complexity. The results confirm that DHLC enables a larger receptive field, with features at each level obtaining rich semantic information from all higher-level features. As in the 1st and 6th rows, when combining the DHLC and the auxiliary supervision, there is a significant improvement of 2.1% AP over the baseline. As in the 2nd and last row, when we perform our default pruning strategy (s_3 version in Fig. 5), CBNetV2 is faster (23.3 vs. 22.4 FPS) and much more accurate (38.0% vs. 36.0% AP) than CBNetV1 [1]. DHLC slows down the

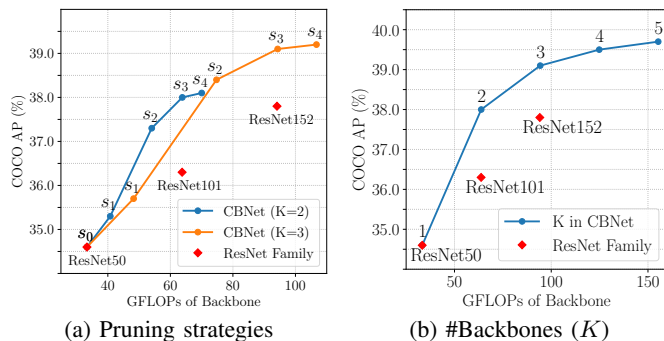


Fig. 6: Performance comparison of CBNet with different numbers of composite backbones (K) and pruning strategies.

detector, while the pruning strategy effectively speeds up the inference speed of CBNetV2.

6) *Importance of Identical Backbones for CBNet*: To verify the necessity of identical backbones in CBNet, we explore the diversity backbones by compositing ResNet50, ResNet101, Res2Net50, and Res2Net101. Note that no pruning is conducted for compositing diverse backbones and backbones from different families do not share the stem layer (Conv1 in Fig. 3). As shown in Table XVI, for backbones belonging to the same family, compositing identical backbones outperforms compositing diverse ones. For example, CB-ResNet50 achieves higher AP with fewer parameters than both ResNet50-C-ResNet101 and ResNet101-C-ResNet50. Similarly, CB-Res2Net50 gains higher or comparable AP with fewer parameters than both Res2Net50-C-Res2Net101 and Res2Net101-C-Res2Net50. For backbones from different families, the observation still holds. For example, CB-Res2Net50 achieves better performance than ResNet50-C-Res2Net101, Res2Net101-C-ResNet50, ResNet101-C-Res2Net50, and Res2Net50-C-ResNet101. These experimental results indicate that increasing the diversity of composite models is not the most efficient way for CBNet. We believe the reason is that using different backbones needs different optimization strategies, which usually output very different learned features and are difficult for joint training. CBNet intends to learn similar features for each grouped backbone, and the stronger the former backbones are, the more representative features the lead backbone outputs. Our experiments show that such a joint-training strategy works best for identical backbone grouping.

This validates the necessity of the identical backbones in CBNet and further distinguishes our approach from ensemble methods where diversity is a key character.

V. CONCLUSION

In this paper, we propose a novel and flexible backbone framework, called *Composite Backbone Network* (CBNet), to improve the performance of cutting-edge object detectors. CBNet consists of a series of backbones with the same network architecture in parallel, the Dense Higher-Level composition strategy, and the auxiliary supervision. Together they construct a robust representative backbone network that uses existing pre-trained backbones under the pre-training fine-tuning paradigm. CBNet has strong generalization capabilities

TABLE XVI: Importance of identical backbones for CBNet. Compositing two identical ResNet50 achieves better performance than compositing ResNet50 and ResNet101.

| Backbone | AP ^{box} | Params | FLOPs |
|------------------------|-------------------|----------------|--------------|
| ResNet50-C-ResNet101 | 37.8 | 88.7 M | 157 G |
| ResNet101-C-ResNet50 | 37.8 | 88.7 M | 157 G |
| CB-ResNet50 | 38.0 | 69.4 M | 121 G |
| Res2Net50-C-Res2Net101 | 41.1 | 89.5 M | 163 G |
| Res2Net101-C-Res2Net50 | 41.4 | 89.5 M | 163 G |
| CB-Res2Net50 | 41.2 | 69.7 M | 125 G |
| CB-Res2Net101 | 43.0 | 108.7 M | 188 G |
| ResNet50-C-Res2Net101 | 40.0 | 89.3 M | 162 G |
| Res2Net101-C-ResNet50 | 41.2 | 89.3 M | 162 G |
| ResNet101-C-Res2Net50 | 39.5 | 88.8 M | 162 G |
| Res2Net50-C-ResNet101 | 40.6 | 88.8 M | 162 G |
| CB-Res2Net50 | 41.2 | 69.7 M | 125 G |

for different backbones and head designs of the detector architecture. Extensive experimental results demonstrate that the proposed CBNet is compatible with various backbone networks, including CNN-based (ResNet, ResNeXt, Res2Net) and Transformer-based (Swin-Transformer) ones. At the same time, CBNet is more effective and efficient than simply increasing the depth and width of the network. Furthermore, CBNet can be flexibly plugged into most mainstream detectors, including one-stage (*e.g.*, RetinaNet) and two-stage (Faster R-CNN, Mask R-CNN, Cascade R-CNN, and Cascade Mask R-CNN) detectors, as well as anchor-based (*e.g.*, Faster R-CNN) and anchor-free-based (ATSS) ones. CBNet is compatible with feature enhancing networks (DCN and HRNet) and model ensemble methods. Specifically, the performances of the above detectors are increased by over 3% AP. In particular, our CB-Swin-L achieves a new record of 59.4% box AP and 51.6% mask AP on COCO test-dev, outperforming prior single-model single-scale results. With multi-scale testing, we achieve a new state-of-the-art result of 60.1% box AP and 52.3% mask AP without extra training data.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant 62176007. This work was also a research achievement of Key Laboratory of Science, Technology, and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank Dr. Han Hu, Prof. Ming-Ming Cheng and Shang-Hua Gao for the insightful discussions.

REFERENCES

- [1] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling, "Cbnet: A novel composite backbone network architecture for object detection," in *AAAI*, 2020. 1, 2, 4, 5, 6, 11
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012. 1, 3
- [3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *ECCV*, 2016. 1, 2
- [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016. 1, 2
- [5] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *TPAMI*, 2017. 1, 2

- [6] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *TPAMI*, 2020. 1, 2, 3
- [7] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, “Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection,” in *CVPR*, 2020. 1, 2, 3, 6, 7
- [8] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” in *ICCV*, 2017. 1, 2, 3
- [9] Z. Cai and N. Vasconcelos, “Cascade R-CNN: delving into high quality object detection,” in *CVPR*, 2018. 1, 2, 3
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *CVPR*, 2009. 1
- [11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017. 1
- [12] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *CVPR*, 2018. 1
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016. 1, 2, 3, 4, 5, 6
- [14] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *CVPR*, 2017. 1, 2, 3, 4
- [15] S. Gao, M. Cheng, K. Zhao, X. Zhang, M. Yang, and P. H. S. Torr, “Res2net: A new multi-scale backbone architecture,” *TPAMI*, 2021. 1, 2, 3, 4
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017. 1, 3
- [17] W. Wang, E. Xie, X. Li, D. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in *ICCV*, 2021. 1, 3, 6, 8, 9
- [18] Y. Li, H. Zhang, and Y. Zhang, “Rethinking training from scratch for object detection,” *arXiv preprint arXiv:2106.03112*, 2021. 1
- [19] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” in *ECCV*, 2014. 2, 4, 6
- [20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *ICCV*, 2021. 2, 3, 4, 6, 7, 8
- [21] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, “Deep high-resolution representation learning for visual recognition,” *TPAMI*, 2021. 2, 3, 6, 8
- [22] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in *ICCV*, 2017. 2, 9
- [23] M. Mao, B. Zhang, D. S. Doermann, J. Guo, S. Han, Y. Feng, X. Wang, and E. Ding, “Probabilistic ranking-aware ensembles for enhanced object detections,” *arXiv preprint arXiv:2105.03139*. 2, 4, 10
- [24] M. Liang and X. Hu, “Recurrent convolutional neural network for object recognition,” in *CVPR*, 2015. 3
- [25] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” in *CVPR*, 2017. 2, 3, 5, 6
- [26] G. Ghiasi, T. Lin, and Q. V. Le, “NAS-FPN: learning scalable feature pyramid architecture for object detection,” in *CVPR*, 2019. 2, 3, 7
- [27] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *CVPR*, 2020. 2, 7
- [28] P. Chen, M. Chang, J. Hsieh, and Y. Chen, “Parallel residual bi-fusion feature pyramid network for accurate single-shot object detection,” *TIP*, 2021. 2
- [29] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, “Libra r-cnn: Towards balanced learning for object detection,” in *CVPR*, 2019. 3
- [30] C. Zhu, Y. He, and M. Savvides, “Feature selective anchor-free module for single-shot object detection,” in *CVPR*, 2019. 3, 7
- [31] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: fully convolutional one-stage object detection,” in *ICCV*, 2019. 3, 7
- [32] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, “Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection,” in *NeurIPS*, 2020. 3, 7
- [33] H. Law and J. Deng, “Cornernet: Detecting objects as paired keypoints,” *IJCV*, 2020. 3
- [34] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “Centernet: Keypoint triplets for object detection,” in *ICCV*, 2019. 3
- [35] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi, “Foveabox: Beyond anchor-based object detection,” *TIP*, 2020. 3
- [36] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *ECCV*, 2020. 3, 7
- [37] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, “NAS-FCOS: fast neural architecture search for object detection,” in *CVPR*, 2020. 3, 7
- [38] X. Du, T. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, “Spinenet: Learning scale-permuted backbone for recognition and localization,” in *CVPR*, 2020. 3, 7
- [39] L. Yao, H. Xu, W. Zhang, X. Liang, and Z. Li, “SM-NAS: structural-to-modular neural architecture search for object detection,” in *AAAI*, 2020. 3, 7
- [40] H. Xu, L. Yao, Z. Li, X. Liang, and W. Zhang, “Auto-fpn: Automatic network architecture adaptation for object detection beyond classification,” in *ICCV*, 2019. 3, 7
- [41] T. Liang, Y. Wang, Z. Tang, G. Hu, and H. Ling, “Opanas: One-shot path aggregation network architecture search for object detection,” in *CVPR*, 2021. 3, 7
- [42] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015. 3
- [43] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *CVPR*, 2017. 3, 5
- [44] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, “Detnet: Design backbone for object detection,” in *ECCV*, 2018. 3
- [45] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang, “Fishnet: A versatile backbone for image, region, and pixel level prediction,” in *NeurIPS*, 2018. 3
- [46] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, “Detnas: Backbone search for object detection,” in *NeurIPS*, 2019. 3
- [47] L. Yao, R. Pi, H. Xu, W. Zhang, Z. Li, and T. Zhang, “Joint-detnas: Upgrade your detector with nas, pruning and dynamic distillation,” in *CVPR*, 2021. 3, 7, 9
- [48] A. Krogh and J. Vedelsby, “Neural network ensembles, cross validation, and active learning,” in *NeurIPS*, 1994. 4, 10
- [49] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *WIDM*, 2018. 4, 10
- [50] C. Zhang and Y. Ma, *Ensemble machine learning: Methods and applications*, 2012. 4
- [51] G. Brown, “Diversity in neural network ensembles,” Ph.D. dissertation, University of Birmingham, UK, 2004. 4
- [52] G. Brown, J. L. Wyatt, R. Harris, and X. Yao, “Diversity creation methods: a survey and categorisation,” *Inf. Fusion*, 2005. 4
- [53] M. Chen, J. Fu, and H. Ling, “One-shot neural ensemble architecture search by diversity-guided search space shrinking,” in *CVPR*, 2021. 4
- [54] A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov, T. Duerig, and V. Ferrari, “The open images dataset V4,” *IJCV*, 2020. 4
- [55] S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun, “Object detection networks on convolutional feature maps,” *TPAMI*, 2017. 4
- [56] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *CVPR*, 2017. 4
- [57] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, “Megdet: A large mini-batch object detector,” in *CVPR*, 2018. 4
- [58] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *CVPR*, 2018. 4
- [59] Z. Li, Y. Ma, Y. Chen, X. Zhang, and J. Sun, “Joint COCO and mapillary workshop at ICCV 2019: COCO instance segmentation challenge track,” *arXiv preprint arXiv:2010.02475*. 4
- [60] Y. Liu, G. Song, Y. Zang, Y. Gao, E. Xie, J. Yan, C. C. Loy, and X. Wang, “1st place solutions for openimage2019 - object detection and instance segmentation,” *arXiv preprint arXiv:2003.07557*. 4
- [61] L. Shen, Z. Lin, and Q. Huang, “Relay backpropagation for effective learning of deep convolutional neural networks,” in *ECCV*, 2016. 5
- [62] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015. 5
- [63] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016. 5
- [64] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable detr: Deformable transformers for end-to-end object detection,” in *ICLR*, 2021. 7
- [65] K. Kim and H. S. Lee, “Probabilistic anchor assignment with iou prediction for object detection,” in *ECCV*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., 2020. 7
- [66] C. Jiang, H. Xu, W. Zhang, X. Liang, and Z. Li, “SP-NAS: serial-to-parallel backbone search for object detection,” in *CVPR*, 2020. 7
- [67] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-yolov4: Scaling cross stage partial network,” in *CVPR*, 2021. 7

- [68] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "Global context networks," *TPAMI*, 2020. 7
- [69] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. J. Smola, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020. 7
- [70] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *CVPR*, 2021. 7
- [71] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019. 6, 11
- [72] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018. 6, 8
- [73] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pvt v2: Improved baselines with pyramid vision transformer," *Computational Visual Media*, 2022. 6, 8, 9
- [74] S. Qiao, L. Chen, and A. L. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *CVPR*, 2021. 6, 9
- [75] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," in *CVPR*, 2019. 6
- [76] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS-improving object detection with one line of code," in *ICCV*, 2017. 6
- [77] Y. Wu and K. He, "Group normalization," *IJCV*, 2020. 6
- [78] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *CVPR*, 2019. 6
- [79] R. Girshick, "Fast r-cnn," in *ICCV*, 2015. 6
- [80] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang, "Dynamic head: Unifying object detection heads with attentions," in *CVPR*, 2021. 7
- [81] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, "Swa object detection," *arXiv preprint arXiv:2012.12645*, 2020. 7
- [82] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu *et al.*, "Mmdetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019. 9
- [83] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. 9
- [84] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021. 9