# CBNet: A Novel Composite Backbone Network Architecture for Object Detection

**Yudong Liu,[1] Yongtao Wang,[1*] Siwei Wang,[1] Tingting Liang,[1] Qijie Zhao,[1] Zhi Tang,[1] Haibin Ling[2]**
[1]Wangxuan Institute of Computer Technology, Peking University
[2]Department of Computer Science, Stony Brook University
{bahuangliuhe, wyt, wangsiwei17, liangtingting, zhaoqijie, tangzhi}@pku.edu.cn
hling@cs.stonybrook.edu

## Abstract

In existing CNN based detectors, the backbone network is a very important component for basic feature[1] extraction, and the performance of the detectors highly depends on it. In this paper, we aim to achieve better detection performance by building a more powerful backbone from existing ones like ResNet and ResNeXt. Specifically, we propose a novel strategy for assembling multiple identical backbones by composite connections between the adjacent backbones, to form a more powerful backbone named *Composite Backbone Network* (CBNet). In this way, CBNet iteratively feeds the output features of the previous backbone, namely *high-level features*, as part of input features to the succeeding backbone, in a stage-by-stage fashion, and finally the feature maps of the last backbone (named *Lead Backbone*) are used for object detection. We show that CBNet can be very easily integrated into most state-of-the-art detectors and significantly improve their performances. For example, it boosts the mAP of FPN, Mask R-CNN and Cascade R-CNN on the COCO dataset by about 1.5 to 3.0 points. Moreover, experimental results show that the instance segmentation results can be improved as well. Specifically, by simply integrating the proposed CBNet into the baseline detector Cascade Mask R-CNN, we achieve a new state-of-the-art result on COCO dataset (mAP of 53.3) with a single model, which demonstrates great effectiveness of the proposed CBNet architecture. Code will be made available at https://github.com/PKUbahuangliuhe/CBNet.

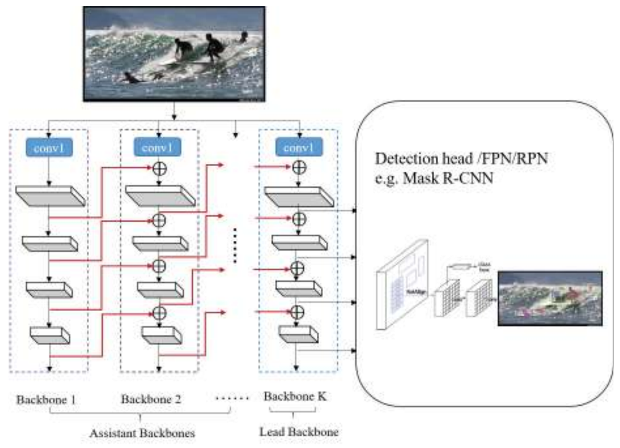

Figure 1: Illustration of the proposed Composite Backbone Network (CBNet) architecture for object detection. CBNet assembles multiple identical backbones (Assistant Backbones and Lead Backbone) by composite connections between the parallel stages of the adjacent backbones. This way, it iteratively feeds the output features of the previous backbone as part of input to the succeeding backbone, in a stage-by-stage fashion, and finally outputs the features of the last backbone (*i.e.*, Lead Backbone) for object detection. The red arrows represent composite connections.

## 1 Introduction

Object detection is one of the most fundamental problems in computer vision, which can serve a wide range of applications such as autonomous driving, intelligent video surveillance, remote sensing, and so on. In recent years, great progresses have been made for object detection thanks to the booming development of the deep convolutional networks (Krizhevsky, Sutskever, and Hinton 2012), and a few excellent detectors have been proposed, *e.g.*, SSD (Liu et al. 2016), Faster R-CNN (Ren et al. 2015), RetinaNet(Lin et al. 2018), FPN (Lin et al. 2017a), Mask R-CNN (He et al. 2017), Cascade R-CNN (Cai and Vasconcelos 2018), *etc*.

Generally speaking, in a typical CNN based object detector, a backbone network is used to extract basic features for detecting objects, which is usually designed for the image classification task and pretrained on the ImageNet dataset (Deng et al. 2009). Not surprisingly, if a backbone can extract more representational features, its host detector will perform better accordingly. In other words, a more powerful backbone can bring better detection performance, as demonstrated in Table 1. Hence, starting from AlexNet (Krizhevsky, Sutskever, and Hinton 2012), deeper and larger (*i.e.*, more powerful) backbones have been exploited by the

*Corresponding author

[1]Here and after, "basic feature" refers in particular to the features that are extracted by the backbone network and used as the input to other functional modules in the detector like detection head, RPN and FPN.

| Backbone Network | $AP_{box}$ | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| *ResNet50* | 41.5 | 60.1 | 45.5 |
| *ResNet101* | 43.3 | 61.7 | 47.2 |
| *ResNeXt101* | 45.9 | 64.4 | 50.2 |
| *ResNeXt152* | 48.3 | 67.0 | 52.8 |
| ***Dual-ResNeXt152 (ours)*** | **50.0** | **68.8** | **54.6** |
| ***Triple-ResNeXt152 (ours)*** | **50.7** | **69.8** | **55.5** |

Table 1: Results of the state-of-the-art detector Cascade Mask R-CNN on the COCO `test-dev` dataset (Lin et al. 2014) with different existing backbones and the proposed Composite Backbone Networks (Dual-ResNeXt152 and Triple-ResNeXt152), which are reproduced by Detectron (Girshick et al. 2018; Cai and Vasconcelos 2018). It shows that deeper and larger backbones bring better detection performance, while our Composite Backbone Network architecture can further strengthen the existing very powerful backbones for object detection such as ResNeXt152.

state-of-the-art detectors, such as VGG (Simonyan and Zisserman 2014), ResNet (He et al. 2016), DenseNet (Huang et al. 2017), ResNeXt (Xie et al. 2017). Despite encouraging results achieved by the state-of-the-art detectors based on deep and large backbones, there is still plenty of room for performance improvement. Moreover, it is very expensive to achieve better detection performance by designing a novel more powerful backbone and pre-training it on ImageNet. In addition, since almost all of the existing backbone networks are originally designed for image classification, directly employing them to extract basic features for object detection may result in suboptimal performance.

To deal with the issues mentioned above, as illustrated in Figure 1, we propose to assemble multiple identical backbones, in a novel way, to build a more powerful backbone for object detection. In particular, the assembled backbones are treated as a whole that we call *Composite Backbone Network* (CBNet). More specifically, CBNet consists of multiple identical backbones (specially called *Assistant Backbones* and *Lead Backbone*) and composite connections between neighbor backbones. From left to right, the output of each stage in an Assistant Backbone, namely higher-level features, flows to the parallel stage of the succeeding backbone as part of inputs through composite connections. Finally, the feature maps of the last backbone named Lead Backbone are used for object detection. Obviously, the features extracted by CBNet for object detection fuse the high-level and low-level features of multiple backbones, hence improve the detection performance. It is worth mentioning that, *we do not need to pretrain CBNet for training a detector integrated with it*. For instead, *we only need to initialize each assembled backbone of CBNet with the pretrained model of the single backbone* that is widely and freely available today, such as ResNet and ResNeXt. In other words, adopting the proposed CBNet is more economical and efficient than designing a novel more powerful backbone and pre-training it on ImageNet.

On the widely tested MS-COCO benchmark (Lin et al. 2014), we conduct experiments by applying the proposed Composite Backbone Network to several state-of-the-art object detectors, such as FPN (Lin et al. 2017a), Mask R-CNN (He et al. 2017) and Cascade R-CNN (Cai and Vasconcelos 2018). Experimental results show that the mAPs of all the detectors consistently increase by 1.5 to 3.0 points, which demonstrates the effectiveness of our Composite Backbone Network. Moreover, with our Composite Backbone Network, the results of instance segmentation are also improved. Specifically, using Triple-ResNeXt152, *i.e.*, Composite Backbone Network architecture of three ResNeXt152 (Xie et al. 2017) backbones, we achieve the new state-of-the-art result on COCO dataset, that is, mAP of 53.3, outperforming all the published object detectors.

To summarize, the major contributions of this work are two-fold:

- We propose a novel method to build a more powerful backbone for object detection by assembling multiple identical backbones, which can significantly improve the performances of various state-of-the-art detectors.
- We achieve the new state-of-the-art result on the MSCOCO dataset with a single model, that is, the mAP of 53.3 for object detection.

In the rest of the paper, after reviewing related work in Sec. 2, we describe in details the proposed CBNet for object detection in Sec. 3. Then, we report the experimental validation in Sec. 4, and draw the conclusion in Sec. 5.

## 2 Related work

**Object detection** Object detection is a fundamental problem in computer vision. The state-of-the-art methods for general object detection can be briefly categorized into two major branches. The first branch contains one-stage methods such as YOLO (Redmon et al. 2016), SSD (Liu et al. 2016), Retinanet (Lin et al. 2017b), FSAF (Zhu, He, and Savvides 2019) and NAS-FPN (Ghiasi, Lin, and Le 2019). The other branch contains two-stage methods such as Faster R-CNN (Ren et al. 2015), FPN (Lin et al. 2017a), Mask R-CNN(He et al. 2017), Cascade R-CNN(Cai and Vasconcelos 2018) and Libra R-CNN (Pang et al. 2019). Although breakthrough has been made and encouraging results have been achieved by the recent CNN based detectors, there is still large room for performance improvement. For example, on MS COCO benchmark (Lin et al. 2014), the best publicly reported mAP is only 52.5 (Peng et al. 2018), which is achieved by model ensemble of four detectors.

**Backbone for Object detection** Backbone is a very important component of a CNN based detector to extract basic features for object detection. Following the original works (*e.g.*, R-CNN (Girshick et al. 2014) and OverFeat (Sermanet et al. 2013)) of applying deep learning to object detection, almost all of the recent detectors adopt the *pretraining and fine-tuning* paradigm, that is, directly use the networks which are pre-trained for ImageNet classification task as their backbones. For instance, VGG (Simonyan and Zisserman 2014), ResNet (He et al. 2016), ResNeXt (Xie et al. 2017) are widely used by the state-of-the-art detectors. Since these backbone networks are originally designed for image

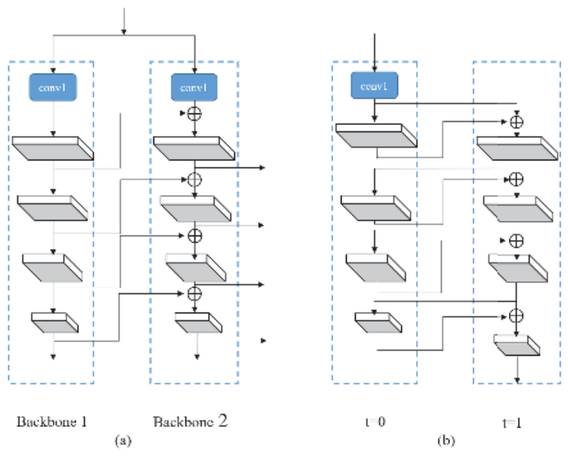

Figure 2: Comparison between (a). our proposed CBNet architecture ($K = 2$) and (b). the unrolled architecture of RCNN (Liang and Hu 2015)($T = 2$).

classification task, directly employing them to extract basic features for object detection may result in suboptimal performance. More recently, two sophisticatedly designed backbones, *i.e.*, DetNet (Li et al. 2018) and FishNet (Sun et al. 2018), are proposed for object detection. These two backbones are specifically designed for the object detection task, and they still need to be pretrained for ImageNet classification task before training (fine tuning) the detector based on them. It is well known that designing and pretraining a novel and powerful backbone like them requires much manpower and computation cost. In an alternative way, we propose a more economic and efficient solution to build a more powerful backbone for object detection, by assembling multiple identical existing backbones (*e.g.*, ResNet and ResNeXt).

**Recurrent Convolution Neural Network** As shown in Figure 2, the proposed architecture of Composite Backbone Network shares some similarity with an unfolded recurrent convolutional neural network (RCNN) (Liang and Hu 2015) architecture, but is significantly different from RCNN. First, as illustrated in Figure 2, the architecture of CBNet is very different, especially for the connections between the parallel stages. Second, in RCNN, the parallel stages of different time steps share the parameters, while in the proposed CBNet, the parallel stages of backbones do not share the parameters. Moreover, if we use RCNN as the backbone of a detector, we need to pretrain it on ImageNet. By contrast, no pretraining for CBNet is needed, since it instead takes existing backbones directly.

## 3 Proposed method

This section elaborates the proposed CBNet in detail. We first describe its architecture and variants in Section 3 and Section 3 respectively. And then, we describe the structure of detection network with CBNet in Section 3.

### Architecture of CBNet

The architecture of the proposed CBNet consists of $K$ identical backbones ($K \geq 2$). Specially, we call the case of K = 2 (as shown in Figure 2.a) as **Dual-Backbone (DB)** for simplicity, and the case of K=3 as **Triple-Backbone (TB)**.

As illustrated in Figure 1, the CBNet architecture consists of two types of backbones: the Lead Backbone $B_K$ and the Assistant Backbones $B_1, B_2, ..., B_{K-1}$. Each backbone comprises $L$ stages (generally $L = 5$), and each stage consists of several convolutional layers with feature maps of the same size. The $l$-th stage of the backbone implements a non-linear transformation $F^l(\cdot)$.

In the traditional convolutional network with only one backbone, 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:

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

Unlike this, in the CBNet architecture, we novelly employ Assistant Backbones $B_1, B_2, ..., B_{k-1}$ to enhance the features of the Lead Backbone $B_k$, by iteratively feeding the output features of the previous backbone as part of input features to the succeeding backbone, in a stage-by-stage fashion. To be more specific, the input of the $l$-th stage of the backbone $B_k$ is the fusion of the output of the previous $l-1$-th stage of $B_k$ (denoted as $x_k^{l-1}$) and the output of the parallel stage of the previous backbone $B_{k-1}$ (denoted as $x_{k-1}^l$). This operation can be formulated as following:

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

where $g(\cdot)$ denotes the composite connection, which consists of a $1\times1$ convolutional layer and batch normalization layer to reduce the channels and an upsample operation. As a result, the output features of the $l$-th stage in $B_{k-1}$ is transformed to the input of the same stage in $B_k$, and added to the original input feature maps to go through the corresponding layers. Considering that this composition style feeds the output of the adjacent higher-level stage of the previous backbone to the succeeding backbone, we call it as **Adjacent Higher-Level Composition (AHLC)**.

For object detection task, only the output of Lead Backbone $x_K^l (l = 2, 3, \ldots, L)$ are taken as the input of the RPN/detection head, while the output of each stage of Assistant Backbones is forwarded into its adjacent backbone. Moreover, the $B_1, B_2, ..., B_{K-1}$ in CBNet can adopt various backbone architectures, such as (He et al. 2016) or ResNeXt (Xie et al. 2017), and can be initialized from the pre-trained model of the single backbone directly.

### Other possible composite styles

**Same Level Composition (SLC)** An intuitive and simple composite style is to fuse the output features from the same stage of backbones. This operation of Same Level Composite (SLC) can be formulated as:

$$x_k^l = F_k^l(x_k^{l-1} + x_{k-1}^{l-1}), l \geq 2. \tag{3}$$

To be more specific, Figure 3.b illustrates the structure of SLC when $K = 2$.

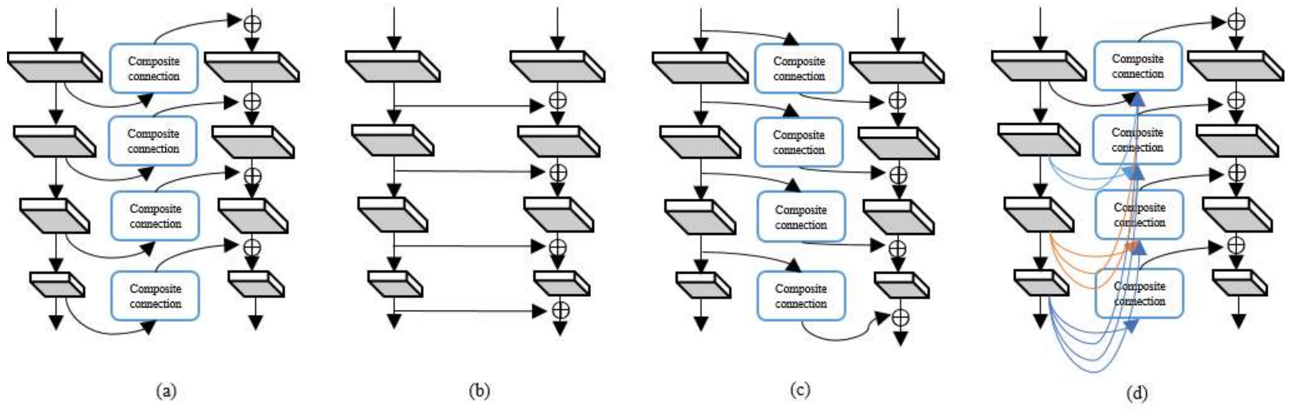

Figure 3: Four kinds of composite styles for Dual-Backbone architecture (an Assistant Backbone and a Lead Backbone). (a) Adjacent Higher-Level Composition (AHLC). (b) Same Level Composition (SLC). (c) Adjacent Lower-Level Composition (ALLC). (d) Dense Higher-Level Composition (DHLC). The composite connection denotes in blue boxes represents some simple operations, *i.e.*, element-wise operation, scaling, $1\times1$ Conv layer and BN layer.

**Adjacent Lower-Level Composition (ALLC)** Contrary to AHLC, another intuitive composite style is to feed the output of the adjacent lower-level stage of the previous backbone to the succeeding backbone. This operation of Adjacent Lower-Level Composition (ALLC). The operation of Inverse Level Composite (ILC) can be formulated as:

$$x_k^l = F_k^l(x_k^{l-1} + g(x_{k-1}^{l-2})), l \geq 3. \quad (4)$$

To be more specific, Figure 3.c illustrates the structure of ILC when $K = 2$.

**Dense Higher-Level Composition (DHLC)** In DenseNet (Huang et al. 2017), each layer is connected to all subsequent layers to build a dense connection in a stage. Inspired by it, we can utilize dense composite connection in our CBNet architecture. The operation of DHLC can be expressed as follows:

$$x_k^l = F_k^l\Big(x_k^{l-1} + \sum_{i=l}^{L} g_i(x_{k-1}^i)\Big), \ \ l \geq 2. \quad (5)$$

As shown in Figure 3.d, when $K = 2$, we assemble the features from all the higher-level stages in the Assistant Backbone, and add the composite features to the output features of the previous stage in the Lead Backbone.

## Architecture of detection network with CBNet

The CBNet architecture is applicable with various off-the-shelf object detectors without additional modifications to the network architectures. In practice, we attach layers of the Lead Backbone with functional networks, RPN (Ren et al. 2015) , detection head (Redmon et al. 2016; Ren et al. 2015; Zhang et al. 2018; Lin et al. 2017a; He et al. 2017; Cai and Vasconcelos 2018).

## 4 Experiments

In this section, we present experimental results on the bounding box detection task and instance segmentation task of the challenging MS-COCO benchmark (Lin et al. 2014). Following the protocol in MS-COCO, we use the `trainval35k` set for training, which is a union of 80k images from the train split and a random 35k subset of images from the 40k image validation split. We report COCO AP on the `test-dev` split for comparisons, which is tested on the evaluation server.

## Implementation details

Baselines methods in this paper are reproduced by ourselves based on the Detectron framework (Girshick et al. 2018). All the baselines are trained with the single-scale strategy, except Cascade Mask R-CNN ResNeXt152. Specifically, the short side of input image is resized to 800, and the longer side is limited to 1,333. We conduct experiments on a machine with 4 NVIDIA Titan X GPUs, CUDA 9.2 and cuDNN 7.1.4 for most experiments. In addition, we train Cascade Mask R-CNN with Dual-ResNeXt152 on a machine with 4 NVIDIA P40 GPUs and Cascade Mask R-CNN with Triple-ResNeXt152 on a machine with 4 NVIDIA V100 GPUs. The data augmentation is simply flipping the images. For most of the original baselines, batch size on a single GPU is two images. Due to the limitation of GPU memory for CBNet, we put one image on each GPU for training the detectors using CBNet. Meanwhile, we set the initial learning rate as half of the default value and train for the same epochs as the original baselines. It is worth noting that, we do not change any other configuration of these baselines except the reduction of the initial learning rate and batch size.

During the inference, we completely use the configuration in the original baselines (Girshick et al. 2018). For Cascade Mask R-CNN with different backbones, we run both single-scale test and multi-scale test. And for other baseline detectors, we run single-scale test, in which the short side of input image is resized to 800, and the longer side is limited to 1,333. It is noted that we do not utilize Soft-NMS (Bodla et al. 2017) during the inference for fair comparison .

| Baseline detector | Single | DB | TB | $AP_{bbox}$ | $AP_{50}$ | $AP_{75}$ | $AP_{mask}$ | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|---|---|---|
| FPN + ResNet101 | ✓ | | | 39.4 | 61.5 | 42.8 | - | - | - |
| | | ✓ | | 41.0 | 62.4 | 44.5 | - | - | - |
| | | | ✓ | **41.7** | **64.0** | **45.5** | - | - | - |
| Mask R-CNN + ResNet101 | ✓ | | | 40.0 | 61.2 | 43.6 | 35.9 | 57.9 | 38.0 |
| | | ✓ | | 41.8 | 62.9 | 45.6 | 37.0 | 59.5 | 39.3 |
| | | | ✓ | **42.4** | **64.0** | **46.7** | **38.1** | **59.9** | **40.8** |
| Cascade R-CNN + ResNet101 | ✓ | | | 42.8 | 62.1 | 46.3 | - | - | - |
| | | ✓ | | 44.3 | 62.5 | 48.1 | - | - | - |
| | | | ✓ | **44.9** | **63.9** | **48.9** | - | - | - |
| Cascade Mask R-CNN + ResNeXt152 | ✓ | | | 48.3 | 67.0 | 52.8 | 41.0 | 64.1 | 44.2 |
| | | ✓ | | 50.0 | 68.8 | 54.6 | 42.0 | 64.6 | 45.6 |
| | | | ✓ | **50.7** | **69.8** | **55.5** | **43.3** | **66.9** | **46.8** |

Table 2: Detection results on the MS-COCO `test-dev` set. We report both object detection and instance segmentation results on four kinds of detectors to demonstrate the effectiveness of CBNet. Single: with/without baseline backbone. DB: with/without Dual-Backbone architecture. TB: with/without Triple-Backbone architecture. Column 5-7 show the results of object detection while column 8-10 show the results of instance segmentation.

## Detection results

To demonstrate the effectiveness of the proposed CBNet, we conduct a series of experiments with the baselines of state-of-the-art detectors, *i.e.*, FPN (Lin et al. 2017a), Mask R-CNN (He et al. 2017) and Cascade R-CNN (Cai and Vasconcelos 2018), and the results are reported in Table 2. In each row of Table 2, we compare a baseline (provided by Detectron (Girshick et al. 2018)) with its variants using the proposed CBNet, and one can see that our CBNet consistently improves all of these baselines with a significant margin. More specifically, the mAPs of these baselines increase by 1.5 to 3 percent.

Furthermore, as presented in Table 3, a new state-of-the-art detection result of 53.3 mAP on the MS-COCO benchmark is achieved by Cascade Mask R-CNN baseline equipped with the proposed CBNet. Notably, this result is achieved just by single model, without any other improvement for the baseline besides taking CBNet as backbone. Hence, this result demonstrates great effectiveness of the proposed CBNet architecture.

Moreover, as shown in Table 2, the proposed CBNet also improves the performances of the baselines for instance segmentation. Compared with bounding boxes prediction (*i.e.*, object detection), pixel-wise classification (*i.e.*, instance segmentation) tends to be more difficult and requires more representational features. And these results demonstrate the effectiveness of CBNet again.

## Comparisons of different composite styles

We further conduct experiments to compare the suggested composite style AHLC with other possible composite styles illustrated in Figure 3, including SLC, ALLC, and DHLC. All of these experiments are conducted based on the Dual-Backbone architecture and the baseline of FPN ResNet101.

**SLC *v.s.* AHLC** As presented in Table 4, SLC gets even worse result than the original baseline. We think the major reason is that the architecture of SLC will bring serious parameter redundancy. To be more specific, the features extracted by the same stage of the two backbones in CBNet are similar, hence SLC cannot learn more semantic information than using single backbone. In other words, the network parameters are not fully utilized, but bring much difficulty on training, leading to a worse result.

**ALLC *v.s.* AHLC** As shown in Table 4, there is a great gap between ALLC and AHLC. We infer that, in our CBNet, if we directly add the lower-level (i.e., shallower) features of the previous backbone to the higher-level (i.e., deeper) ones of the succeeding backbone, the semantic information of the latter ones will be largely harmed. On the contrary, if we add the deeper features of the previous backbone to the shallow ones of the succeeding backbone, the semantic information of the latter ones can be largely enhanced.

**DHLC *v.s.* AHLC** The results in Table 4 show that DHLC does not bring performance improvement as AHLC, although it adds more composite connections than AHLC. We infer that, the success of Composite Backbone Network lies mainly in the composite connections between adjacent stages, while the other composite connections do not enrich much feature since they are too far away.

Obviously, CBNets of these composite styles have same amount of the network parameter (i.e., about twice amount of the network parameters than single backbone), but only AHLC brings optimal detection performance improvement. These experiment results prove that *only increasing parameters or adding additional backbone may not bring better result.* Moreover, these experiment also show that *composite connections should be added properly.* Hence, these experiment results actually demonstrate that *the suggested composite style AHLC is effective and nontrivial.*

## Sharing weights for CBNet

Due to the fuse of more backbones, CBNet increases the number of network parameters. To further demonstrate that the improvement of detection performance mainly comes from the composite architecture rather than the increase of network parameters, we conduct experiments on FPN, with

| Method | Backbone | $AP_{box}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *one stage:* | | | | | | | |
| SSD512 (Liu et al. 2016) | VGG16 | 28.8 | 48.5 | 30.3 | 10.9 | 31.8 | 43.5 |
| RetinaNet (Lin et al. 2017b) | ResNeXt101 | 40.8 | 61.1 | 44.1 | 24.1 | 44.2 | 51.2 |
| RefineDet (Zhang et al. 2018)* | ResNet101 | 41.8 | 62.9 | 45.7 | 25.6 | 45.1 | 54.1 |
| CornerNet (Zhang et al. 2018)* | Hourglass-104 | 42.2 | 57.8 | 45.2 | 20.7 | 44.8 | 56.6 |
| M2Det (Zhao et al. 2018)* | VGG16 | 44.2 | 64.6 | 49.3 | 29.2 | 47.9 | 55.1 |
| FSAF (Zhu, He, and Savvides 2019)* | ResNext-101 | 44.6 | 65.2 | 48.6 | 29.7 | 47.1 | 54.6 |
| NAS-FPN (Ghiasi, Lin, and Le 2019) | AmoebaNet | 48.3 | - | - | - | - | - |
| *two stage:* | | | | | | | |
| Faster R-CNN (Ren et al. 2015) | VGG16 | 21.9 | 42.7 | - | - | - | - |
| R-FCN (Dai et al. 2016) | ResNet101 | 29.9 | 51.9 | - | 10.8 | 32.8 | 45.0 |
| FPN (Lin et al. 2017a) | ResNet101 | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Mask R-CNN (He et al. 2017) | ResNet101 | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 |
| Cascade R-CNN (Cai and Vasconcelos 2018) | ResNet101 | 42.8 | 62.1 | 46.3 | 23.7 | 45.5 | 55.2 |
| Libra R-CNN (Pang et al. 2019) | ResNext-101 | 43.0 | 64.0 | 47.0 | 25.3 | 45.6 | 54.6 |
| SNIP (model ensemble) (Singh and Davis 2018)* | - | 48.3 | 69.7 | 53.7 | 31.4 | 51.6 | 60.7 |
| SINPER (Singh, Najibi, and Davis 2018)* | ResNet101 | 47.6 | 68.5 | 53.4 | 30.9 | 50.6 | 60.7 |
| Cascade Mask R-CNN (Girshick et al. 2018)* | ResNeXt152 | 50.2 | 68.2 | 54.9 | 31.9 | 52.9 | 63.5 |
| MegDet (model ensemble) (Peng et al. 2018)* | - | 52.5 | - | - | - | - | - |
| *ours:(single model)* | | | | | | | |
| Cascade Mask R-CNN * | ***Dual-ResNeXt152*** | **52.8** | **70.6** | **58.0** | **34.9** | **55.4** | **65.3** |
| Cascade Mask R-CNN * | ***Triple-ResNeXt152*** | **53.3** | **71.9** | **58.5** | **35.5** | **55.8** | **66.7** |

Table 3: Object detection comparison between our methods and state-of-the-art detectors on COCO `test-dev` set. * : utilizing multi-scale testing.

| DB | Composite style | $AP_{box}$ | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| | - | 39.4 | 61.5 | 42.8 |
| ✓ | SLC | 38.9 | 60.8 | 42.0 |
| ✓ | ALLC | 36.5 | 57.6 | 39.6 |
| ✓ | DHLC | 40.7 | 61.8 | 44.0 |
| ✓ | AHLC | **41.0** | **62.4** | **44.5** |

Table 4: Comparison between different composite styles, the baseline is FPN ResNet101 (Lin et al. 2017a). DB: with/without Dual-Backbone. ”SLC” represents Same Level Composition, ”ALLC” represents Adajacent Lower-Level Composition, ”DHLC” is Dense Higher-Level Composition and ”AHLC” is Adjacent Higher-Level Composition.

| Baseline detector | *DB* | Share | $AP_{box}$ | *mb* |
|---|---|---|---|---|
| FPN + ResNet101 | | | 39.4 | 470 |
| | ✓ | ✓ | 40.4 | 492 |
| | ✓ | | 41.0 | 815 |

Table 5: Comparison of with/without sharing weights for Dual-Backbone architecture. DB: with/without Dual-Backbone. Share: with/without sharing weights. $AP_{box}$: detection results on COCO `test-dev` dataset. *mb*: the model size.

the configuration of sharing the weighs of two backbones in Dual-ResNet101, and the results are shown in Table 5. We can see that when sharing the weights of backbones in CBNet, the increment of parameters is negligible, but the detection result is still much better than the baseline (*e.g.*, mAP 40.4 *v.s.* 39.4). However, when we do not share the weights, the improvement is minor (mAP from 40.4 to 41.0), which proves that *it is the composite architecture that boosts the performance dominantly, rather than the increase of network parameters*.

## Number of backbones in CBNet

We conduct experiments to investigate the relationship between the number of backbones in CBNet and the detection performance by taking FPN-ResNet101 as the baseline, and the results are shown in Figure 4. It can be noted that the detection mAP steadily increases with the number of backbones, and tends to converge when the number of backbones reaches three. Hence, considering the speed and memory cost, we suggest to use Dual-Backbone and Triple-Backbone architectures.

## An accelerated version of CBNet

The major drawback of the proposed CBNet is that it will slows down the inference speed of the baseline detector since it uses more backbones to extract features thus increases the computation complexity. For example, as shown in Table 6, DB increases the AP of FPN by 1.6 percent but slows down the detection speed from 8.1 fps to 5.5 fps. To alleviate this problem, we further propose an accelerated version of the CBNet as illustrated in Figure 5, by removing the two early stages of the Assistant Backbone. As demonstrated in Table 6, this accelerated version can significantly improve the speed (from 5.5 fps to 6.9 fps) while not harming the detection accuracy (i.e., AP) a lot (from 41.0 to 40.8).

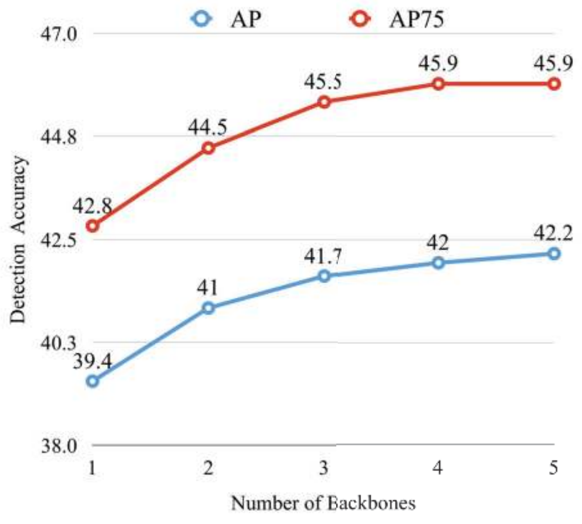

Figure 4: Object detection results on the MS-COCO `test-dev` dataset using different numbers of backbones in CBNet architecture based on FPN ResNet101.

| Baseline detector | DB | $\Psi$ | $AP_{box}$ | *fps* |
|---|---|---|---|---|
| FPN + ResNet101 | | | 39.4 | 8.1 |
| | ✓ | | **41.0** | 5.5 |
| | ✓ | ✓ | **40.8** | 6.9 |

Table 6: Performance comparison between the original DB and the accelerated version. DB: with/without Dual-Backbone. $\Psi$: with/without the acceleration modification of the CBNet architecture illustrated in Figure 5.

## Effectiveness of basic feature enhancement by CBNet

We think the critical reason for CBNet to outperform the single backbone network is: *it can extract more representational basic features than the original single backbone network, which is originally designed for classification problem.* To verify this, as illustrated in Figure 6, we visualize and compare the intermediate feature maps extracted by our CBNet and the original single backbone in the detectors for some examples. The example image in Figure 6 contains two foreground objects: a person and a tennis ball. Obviously, the person is a large-size object and the tennis ball is a small-size one. Hence, we correspondingly visualize the large scale feature maps (for detecting small objects) and the small scale feature maps (for detecting large objects) extracted by our CBNet and the original single backbone. One can see that, the feature maps extracted by our CBNet consistently have stronger activation values at the foreground object and weaker activation values at the background. This visualization example shows that our CBNet is more effective to extract representational basic features for object detection.

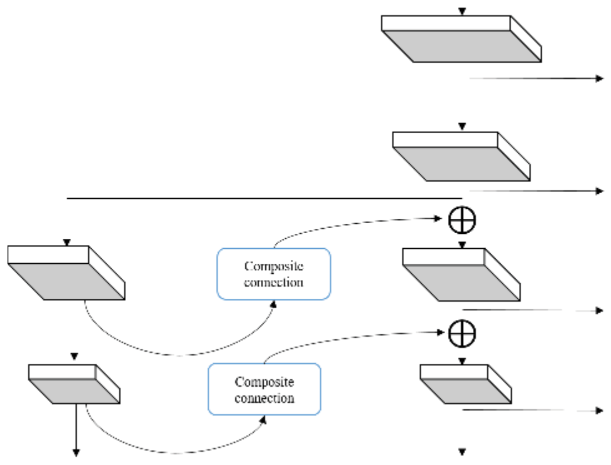

Figure 5: An accelerated version of CBNet ($K = 2$).

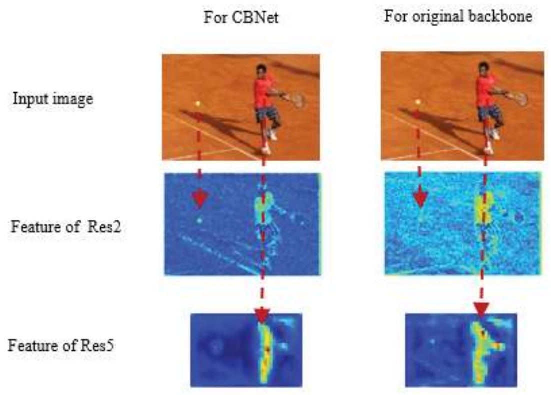

Figure 6: Visualization comparison of the features extracted by our CBNet (Dual-ResNet101) and the original backbone (ResNet101). The baseline detector is FPN-ResNet101. For each backbone, we visualize the Res2 and Res5 according to the size of the foreground objects, by averaging feature maps along channel dimension. **Best viewed in color**.

## 5 Conclusion

In this paper, a novel network architecture called *Composite Backbone Network* (CBNet) is proposed to boost the performance of state-of-the-art object detectors. CBNet consists of a series of backbones with same network structure and uses composite connections to link these backbones. Specifically, the output of each stage in a previous backbone flows to the parallel stage of the succeeding backbone as part of inputs through composite connections. Finally, the feature maps of the last backbone namely Lead Backbone are used for object detection. Extensive experimental results demonstrate that the proposed CBNet is beneficial for many state-of-the-art detectors, such as FPN, Mask R-CNN, and Cascade R-CNN, to improve their detection accuracy. To be more specific, the mAPs of the detectors mentioned above on the COCO dataset are increased by about 1.5 to 3 points, and a new state-of-the art result on COCO with the mAP of 53.3 is achieved by simply integrating CBNet into the Cascade Mask R-CNN baseline. Simultaneously, experimental results show that it is also very effective to improve the instance segmentation performance. Additional ablation studies further demonstrate the effectiveness of the proposed architecture and the composite connection module.

## Acknowledgment

This work is supported by National Natural Science Foundation of China under Grant 61673029. This work is also a research achievement of Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Ling is supported in part by Yahoo Faculty Research and Engagement Program Award and Amazon AWS Machine Learning Research Award.

## References

Bodla, N.; Singh, B.; Chellappa, R.; and Davis, L. S. 2017. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, 5561–5569.

Cai, Z., and Vasconcelos, N. 2018. Cascade r-cnn: Delving into high quality object detection. In *CVPR*.

Dai, J.; Li, Y.; He, K.; and Sun, J. 2016. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, 379–387.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. Ieee.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7036–7045.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 580–587.

Girshick, R.; Radosavovic, I.; Gkioxari, G.; Dollár, P.; and He, K. 2018. Detectron. https://github.com/facebookresearch/detectron.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*, 2980–2988. IEEE.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*, volume 1, 3.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; and Sun, J. 2018. Detnet: Design backbone for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 334–350.

Liang, M., and Hu, X. 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3367–3375.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017a. Feature pyramid networks for object detection. In *CVPR*, volume 1, 4.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017b. Focal loss for dense object detection. In *ICCV*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2018. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*, 21–37. Springer.

Pang, J.; Chen, K.; Shi, J.; Feng, H.; Ouyang, W.; and Lin, D. 2019. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 821–830.

Peng, C.; Xiao, T.; Li, Z.; Jiang, Y.; Zhang, X.; Jia, K.; Yu, G.; and Sun, J. 2018. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6181–6189.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 91–99.

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; and LeCun, Y. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, B., and Davis, L. S. 2018. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3578–3587.

Singh, B.; Najibi, M.; and Davis, L. S. 2018. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, 9333–9343.

Sun, S.; Pang, J.; Shi, J.; Yi, S.; and Ouyang, W. 2018. Fishnet: A versatile backbone for image, region, and pixel level prediction. In *Advances in Neural Information Processing Systems*, 762–772.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 5987–5995. IEEE.

Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; and Li, S. Z. 2018. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4203–4212.

Zhao, Q.; Sheng, T.; Wang, Y.; Tang, Z.; Chen, Y.; Cai, L.; and Ling, H. 2018. M2det: A single-shot object detector based on multi-level feature pyramid network. *arXiv preprint arXiv:1811.04533*.

Zhu, C.; He, Y.; and Savvides, M. 2019. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 840–849.