DetNet: Design Backbone for Object Detection

Zeming Li[1][0000−0002−1599−2853], Chao Peng[2][0000−0003−4069−4775], Gang Yu[2][0000−0001−5570−2710], Xiangyu Zhang[2][0000−0003−2138−4608], Yangdong Deng[1][0000−0002−8257−693X], and Jian Sun[2][0000−0002−6178−4166]

[1] School of Software, Tsinghua University
{lizm15@mails.tsinghua.edu.cn, dengyd@tsinghua.edu.cn}
[2] Megvii Inc. (Face++), {pengchao, yugang, zhangxiangyu, sunjian}@megvii.com

Abstract. Recent CNN based object detectors, either one-stage methods like YOLO, SSD, and RetinaNet, or two-stage detectors like Faster R-CNN, R-FCN and FPN, are usually trying to directly finetune from ImageNet pre-trained models designed for the task of image classification. However, there has been little work discussing the backbone feature extractor specifically designed for the task of object detection. More importantly, there are several differences between the tasks of image classification and object detection. (i) Recent object detectors like FPN and RetinaNet usually involve extra stages against the task of image classification to handle the objects with various scales. (ii) Object detection not only needs to recognize the category of the object instances but also spatially locate them. Large downsampling factors bring large valid receptive field, which is good for image classification, but compromises the object location ability. Due to the gap between the image classification and object detection, we propose DetNet in this paper, which is a novel backbone network specifically designed for object detection. Moreover, DetNet includes the extra stages against traditional backbone network for image classification, while maintains high spatial resolution in deeper layers. Without any bells and whistles, state-of-the-art results have been obtained for both object detection and instance segmentation on the MSCOCO benchmark based on our DetNet (4.8G FLOPs) backbone. Codes will be released[3].

Keywords: Object Detection; Convolutional Neural Network, Image Classification

1 Introduction

Object detection is one of the most fundamental tasks in computer vision. Due to the rapid progress of deep convolutional neural networks (CNN) [17, 35, 36, 10, 16, 38, 12, 40, 15, 11], the performance of object detection has been significantly improved.

Recent CNN based object detectors can be categorized into one-stage detectors, like YOLO [29, 30], SSD [24], and RetinaNet [22], and two-stage detectors, e.g. Faster R-CNN [31], R-FCN [18], FPN [21]. Both of them depend on

[3] https://github.com/zengarden/DetNet

the backbone network pretrained for the ImageNet classification task. However, there is a gap between the image classification and the object detection problem, which not only needs to recognize the category of the object instances but also spatially localize the bounding-boxes. More specifically, there are two problems using the classification backbone for object detection tasks. (i) Recent detectors, e.g., FPN, involve extra stages compared with the backbone network for ImageNet classification in order to detect objects with various sizes. (ii) Traditional backbones produce higher receptive field based on large downsampling factors, which is beneficial to the visual classification. However, the spatial resolution is compromised which will fail to accurately localize the large objects and recognize the small objects.

A well designed detection backbone should tackle all of the problems above. In this paper, we propose DetNet, which is a novel backbone designed for object detection. More specifically, to address the large scale variations of the object instances, DetNet involves additional stages which are utilized in the recent object detectors like FPN. Different from traditional pre-trained models for ImageNet classification, we maintain the spatial resolution of the features even though extra stages are included. However, high resolution feature maps bring more challenges to build a deep neural network due to the computational and memory cost. To keep the efficiency of our DetNet, we employ a low complexity dilated bottleneck structure. By integrating these improvements, our DetNet not only maintains high resolution feature maps but also keeps the large receptive field, both of which are important for the object detection task.

To summarize, we have the following contributions:

- We are the first to analyze the inherent drawbacks of traditional ImageNet pre-trained model for fine-tuning recent object detectors.
- We propose a novel backbone, called DetNet, which is specifically designed for the object detection task by maintaining the spatial resolution and enlarging the receptive field.
- We achieve new state-of-the-art results on MSCOCO object detection and instance segmentation track based on a low complexity DetNet59 backbone.

2 Related Works

Object detection is a heavily researched topic in computer vision. It aims at finding "where" and "what" each object instance is when given an image. Old detectors extract image features by using hand-engineered object component descriptors, such as HOG [5], SIFT [26], Selective Search [37], Edge Box [41]. For a long time, DPM [8] and its variants were the dominant methods among traditional object detectors. With the rapid progress of deep convolutional neural networks, CNN based object detectors have yielded remarkable results and become a new trend in the detection literature. In the network structure, recent CNN based detectors are usually split into two parts. The one is the backbone network, and the other is the detection branch. We briefly introduce these two parts as follows.

2.1 Backbone Network

The backbone networks for object detection are usually borrowed from the ImageNet [32] classification. In the last few years, ImageNet has been regarded as the most authoritative datasets to evaluate the capability of deep convolution neural networks. Many novel networks are designed to get higher performance for ImageNet. AlexNet [17] is among the first to try to increase the depth of CNN. In order to reduce the network computation and increase the valid receptive field, AlexNet down-samples the feature map with 32 strides which is a standard setting for the following works. VGGNet [35] stacks 3x3 convolution operation to build a deeper network, while still involves 32 strides in feature maps. Most of the following researches adopt VGG like structure, and design a better component in each stage (split by stride). GoogleNet [36] proposes a novel inception block to involve more diverse features. ResNet [10] adopts "bottleneck" design with residual sum operation in each stage, which has been proved a simple and efficient way to build a deeper neural network. ResNext [38] and Xception [2] use group convolution layer to replace the traditional convolution. It reduces the parameters and increases the accuracy simultaneously. DenseNet [13] densely concat several layers, it further reduces parameters while keeping competitive accuracy. Another different research is Dilated Residual Network [39] which extracts features with less strides. DRN achieves notable results on segmentation, while has little discussion on object detection. There are still lots of research for efficient backbone, such as [11, 40, 15]. However they are usually designed for classification.

2.2 Object Detection Branch

Detection branch is usually attached to the base-model which is designed and trained for ImageNet classification dataset. There are two different design logic for object detection. The one is one-stage detector, which directly uses backbone for object instance prediction. For example, YOLO [29, 30] uses a simple efficient backbone DarkNet[29], and then simplifies detection as a regression problem. SSD [24] adopts reduced VGGNet[35] and extracts features in multi-layers, which enables network more powerful to handle variant object scales. RetinaNet [22] uses ResNet as a basic feature extractor, then involves "Focal" loss [22] to address class imbalance issue caused by extreme foreground-background ratio. The other popular pipeline is the two-stage detector. Specifically, recent two-stage detector will predict lots of proposals first based on the backbone, then an additional classifier is involved for proposal classification and regression. Faster R-CNN [31] directly generates proposals from the backbone by using Region Proposal Network (RPN). R-FCN [18] proposes to generate a position sensitive feature map from the output of the backbone, then a novel pooling methods called position sensitive pooling is utilized for each proposal. Deformable convolution Networks [4] tries to enable convolution operation with geometric transformations by learning additional offsets without supervision. It is among the first to ameliorate backbone for object detection. Feature Pyramid Network [21]

constructs feature pyramids by exploiting inherent multi-scale, pyramidal hierarchy of deep convolutional networks, specifically FPN combines multi-layer output by utilizing U-shape structure, and still borrows the traditional ResNet without further study. DSOD [33] first proposes to train detection from scratch, whose results are lower than pretrained methods.

In conclusion, traditional backbones are usually designed for ImageNet classification. What is the suitable backbone for object detection is still an unexplored field. Most of the recent object detectors, no matter one-stage or two-stage, follow the pipeline of ImageNet pre-trained models, which is not optimal for detection performance. In this paper, we propose DetNet. The key idea of DetNet is to design a better backbone for object detection.

3 DetNet: A Backbone network for Object Detection

3.1 Motivation

Recent object detectors usually rely on a backbone network which is pretrained on the ImageNet classification dataset. As the task of ImageNet classification is different from the object detection which not only needs to recognize the category of the objects but also spatially localize the bounding-boxes. The design principles for the image classification is not good for the localization task as the spatial resolution of the feature maps is gradually decreased for the standard networks like VGG16 and Resnet. A few techniques like Feature Pyramid Network (FPN) as in Fig. 1 A. [21] and dilation are applied to these networks to maintain the spatial resolution. However, there still exists the following three problems when trained with these backbone networks.

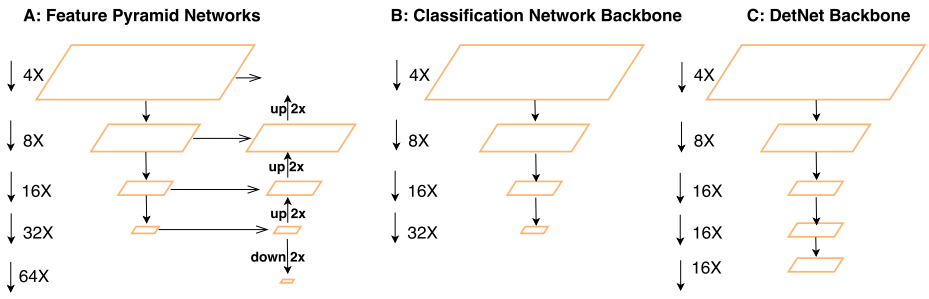

Fig. 1. Comparisons of different backbones used in FPN. Feature pyramid networks (FPN) with the traditional backbone is illustrated in (A). The traditional backbone for image classification is illustrated in (B). Our proposed backbone is illustrated in (C), which has higher spatial resolution and the same stages as FPN. We do not illustrate stage 1 (with stride 2) feature map due to the limitation of figure size.

The number of network stages is different. As shown in Fig. 1 B, typical classification network involves 5 stages, with each stage down-sampling feature maps by pooling 2x or stride 2 convolution. Thus the spatial size of the output feature map is "32x" sub-sampled. Different from traditional classification network, feature pyramid detectors usually adopt more stages. For example, in Feature Pyramid Networks (FPN) [21], an additional stage *P6* is added to handle larger objects. The stages of *P6* and *P7* are added in RetinaNet [22] in a similar way. Obviously, extra stages like *P6* are not pre-trained in the ImageNet dataset.

Weak visibility (localization) of large objects. The feature map with strong semantic information has strides of 32 respect to the input image, which brings large valid receptive field and leads the success of ImageNet classification task. However, large stride factor is harmful for the object localization. In Feature Pyramid Networks, the large object is generated and predicted within the deeper layers, the boundary of these object may be too blurry to get an accurate regression. This case is even worse when more stages are involved into the classification network, since more down-sampling brings more strides to object.

Invisibility (recall) of small objects. Another drawback of large stride is the missing of small objects. The information from the small objects will be easily weaken as the spatial resolution of the feature maps is decreased and the large context information is integrated. Therefore, Feature Pyramid Network predicts small object in shallower layers. However, shallow layers usually only have low semantic information which may be not sufficient to recognize the category of the object instances. Therefore the detectors usually enhance their classification capability by involving the context cues of high-level representations from the deeper layers. As Fig. 1 A shows, Feature Pyramid Networks relieve it by adopting bottom-up pathway. However, if the small objects are missing in deeper layers, these context cues will be decreased simultaneously.

To address these problems, we propose DetNet which has following characteristics. (i) The number of stages is directly designed for Object Detection. (ii) Even though we involve more stages (such as 6 stages or 7 stages) than traditional classification network, we maintain high spatial resolution of the feature maps, while keeping large receptive field.

DetNet has several advantages over traditional backbone networks like ResNet for object detection. First, DetNet has exactly the same number of stages as the detector used, therefore extra stages like *P6* can be pre-trained in the ImageNet dataset. Second, benefited by high resolution feature maps in the last stage, DetNet is more powerful in locating the boundary of astronomical objects and finding the small objects. More detailed discussion can be referred to Section 4.

3.2 DetNet Design

In this subsection, we will present the detailed structure of DetNet. We adopt ResNet-50 as our baseline, which is widely used as the backbone network in a lot

of object detectors. To fairly compare with the ResNet-50, we keep stage 1,2,3,4 the same as original ResNet-50 for our DetNet.

There are two challenges to make an efficient and effective backbone for object detection. On the one hand, keeping the spatial resolution for deep neural network costs extremely large amount of time and memory. On the other hand, reducing the down-sampling factor will lead to the small valid receptive field, which will be harmful to many vision tasks, such as image classification and semantic segmentation.

DetNet is carefully designed to address the two challenges. Specifically, DetNet follows the same setting for ResNet from the first stage to the fourth stage. The difference starts from the fifth stage and an overview of our DetNet for image classification can be found in Fig. 2 D. Let us discuss the implementation details of DetNet59 derived from the ResNet50. Similarly, our DetNet can be easily extended with deep layers like ResNet101. The detailed design of our DetNet59 is illustrated as follows:

- We introduce the extra stage, e.g., *P6*, in the backbone which will be utilized for object detection as in FPN. Meanwhile, we fix the spatial resolution as 16x downsampling after stage 4.
- Since the spatial size is fixed after stage 4, in order to introduce a new stage, we employ a dilated $[27, 25, 1]$ bottleneck with 1x1 convolution projection (Fig. 2 B) in the beginning of the each stage. We find the model in Fig. 2 B is important for multi-stage detectors like FPN.
- We apply bottleneck with dilation as a basic network block to efficiently enlarge the receptive field. Since dilated convolution is still time consuming, our stage 5 and stage 6 keep the same channels as stage 4 (256 input channels for bottleneck block). This is different from traditional backbone design, which will double channels in a later stage.

It is easy to integrate DetNet with any detectors with/without feature pyramid. Without losing representativeness, we adopt prominent detector FPN as our baselines to validate the effectiveness of DetNet. Since DetNet only changes the backbone of FPN, we fix the other structures in FPN except for backbone. Because we do not reduce spatial size after stage 4 of Resnet-50, we simplely sum the output of these stages in top-down path way.

4 Experiments

In this section, we will evaluate our approach on the popular MS COCO benchmark, which has 80 objects categories. There are 80k images in the training set, and 40k images in the validation dataset. Following a common practice, we further split the 40k validation set into 35k *large-val* datasets and 5k *mini-val* datasets. All of our validation experiments involve training set and the *large-val* for training (about 115k images), then test on 5k *mini-val* datasets. We also report the final results of our approach on COCO *test-dev*, which has no disclosed labels.

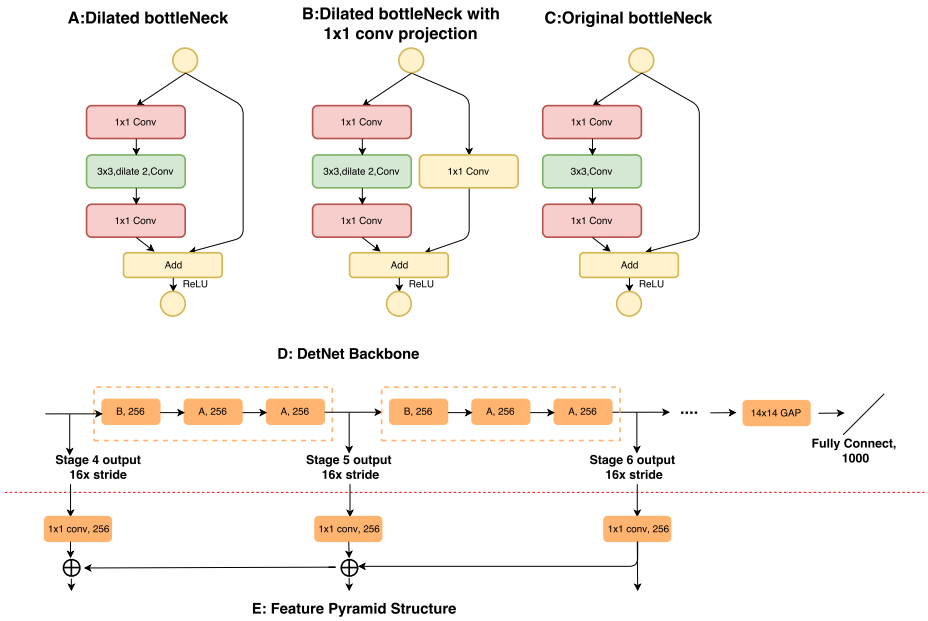

Fig. 2. Detail structure of DetNet (D) and DetNet based Feature Pyramid Network (E). Different bottleneck block used in DetNet is illustrated in (A, B). The original bottleneck is illustrated in (C). DetNet follows the same design as ResNet before stage 4, while keeps spatial size after stage 4 (e.g. stage 5 and 6).

We use standard coco metrics to evaluate our approach, including AP (averaged precision over intersection-over-union thresholds), AP_{50}, AP_{75} (AP at use different IoU thresholds), and AP_S, AP_M, AP_L (AP at different scales: small,middle,large).

4.1 Detector training and inference

Following training strategies provided by Detectron [4] repository [7], our detectors are end-to-end trained on 8 Pascal TITAN XP GPUs, optimized by synchronized SGD with a weight decay of 0.0001 and momentum of 0.9. Each mini-batch has 2 images, so the effective batch-size is 16. We resize the shorter edge of the image to 800 pixels, the longer edge is limited to 1333 pixels to avoid large memory cost. We pad the images within mini-batch to the same size by filling zeros into the right-bottom of the image. We use typical "2x" training settings used in Detectron [7]. Learning rate is set to 0.02 at the begin of the training, and then decreased by a factor of 0.1 after 120k and 160k iterations and finally terminates at 180k iterations. We also warm-up our training by using smaller learning rate 0.02×0.3 for first 500 iterations.

[4] https://github.com/facebookresearch/Detectron

All experiments are initialized with ImageNet pre-trained weights. We fix the parameters of stage 1 in the backbone network. Batch normalization is also fixed during detector fine-tuning. We only adopt a simple horizontal flip data augmentation. As for proposal generation, unless explicitly stated, we first pick up 12000 proposals with highest scores, then followed by non maximum suppression (NMS) operation to get at most 2000 RoIs for training. During testing, we use 6000/1000 (6000 highest scores for NMS, 1000 RoIs after NMS) setting. We also involve popular RoI-Align technique used in Mask R-CNN [9].

4.2 Backbone training and Inference

Following most hyper-parameters and training settings provided by ResNext [38], we train backbone on ImageNet classification datasets by 8 Pascal TITAN XP GPUs with 256 total batch size. Following the standard evaluation strategy for testing, we report the error on the single 224x224 center crop from the image with 256 shorter sides.

4.3 Main Results

We adopt FPN with the ResNet-50 backbone as our baseline because FPN is a prominent detector for many other vision tasks, such as instance segmentation and skeleton [9]. To validate the effectiveness of DetNet for FPN, we propose DetNet-59 which involves an additional stage compared with ResNet-50. More design details can be found in Section 3. Then we replace ResNet-50 backbone with DetNet-59 and keep the other structures the same as the original FPN.

We first train DetNet-59 on ImageNet classification, results are shown in Table 1. DetNet-59 has 23.5% top-1 error at the cost of 4.8G FLOPs,. Then we train FPN with DetNet-59, and compare it with ResNet-50 based FPN. From Table 1 we can see DetNet-59 has superior performance than ResNet-50 (over 2 points gains in mAP).

backbone	Classification		FPN results					
	Top1 err	FLOPs,	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-50	24.1	3.8G	37.9	60.0	41.2	22.9	40.6	49.2
DetNet-59	23.5	4.8G	**40.2**	61.7	43.7	23.9	43.2	52.0
ResNet-101	23.0	7.6G	39.8	62.0	43.5	24.1	43.4	51.7
DetNet-101	23.0	7.9G	41.8	62.8	45.7	25.4	45.2	55.1

Table 1. Results of different backbones used in FPN. We first report the standard Top-1 error on ImageNet classification (the lower error is, the better accuracy in classification). FLOPs means the computation complexity. We also illustrate FPN COCO results to investigate effectiveness of these backbone for object detection.

Since DetNet-59 has more parameters than ResNet-50 (because we involving additional stage for FPN *P6*), a natural hypothesis is that the improvement is

mainly due to more parameters. To validate the effectiveness of DetNet-59, we also train FPN with ResNet-101 which has 7.6G FLOPs complexity, the results is 39.8 mAP. ResNet-101 has much more FLOPs than DetNet-59, and still yields lower mAP than DetNet-59. We further add the FPN experiments based on DetNet-101. Specifically, DetNet-101 has 20 (6 in DetNet-59) repeated bottleneck blocks in ResNet stage 4. As expected, DetNet-101 has superior results than ResNet-101, which validates that DetNet is more suitable than ResNet as a backbone network for object detection.

As DetNet is directly designed for object detection, to further validate the advantage of DetNet, we train FPN based on DetNet-59 and ResNet-50 from scratch. The results are shown in Table 2. Noticing that we use multi-gpu synchronized batch normalization during training as in [28] in order to train from scratch. Concluding from the results, DetNet-59 still outperforms ResNet-50 by 1.8 points, which further validate that DetNet is more suitable for object detection.

backbone	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-50 from scratch	34.5	55.2	37.7	20.4	36.7	44.5
DetNet-59 from scratch	**36.3**	56.5	39.3	22.0	38.4	46.9

Table 2. FPN results on different backbones, which is trained from scratch. Since we don't involve ImageNet pre-trained weights, we want to directly compare backbone capability for object detection.

4.4 Results analysis

In this subsection, we will analyze how DetNet improves the object detection. There are two key-points in object detection evaluation: average precision (AP) and average recall (AR). AR means how much objects we can find out, AP means how much objects are correctly localized (right label for classification). AP and AR are usually evaluated on different IoU threshold to validate the regression capability for object location. The larger IoU is, the more accurate regression needs. AP and AR are also evaluated on different range of bounding box areas (small, middle, and large) to find the detail results on the various scales of the objects.

At first, we investigate the impact of DetNet on detection accuracy. We evaluate the performance on different IoU thresholds and object scales as shown in Table 3.

DetNet-59 has an impressive improvement in the performance of large object location, which brings 5.5 (40.0 vs 34.5) points gains in AP_{85}@large. The reason is that original ResNet based FPN has a big stride in deeper feature map, large objects may be challenging to get an accurate regression.

Models	scales	mAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{85}
ResNet-50	over all scales	37.9	60.0	55.1	47.2	33.1	22.1
	small	22.9	40.1	35.5	28.0	17.5	10.4
	middle	40.6	63.9	59.0	51.2	35.7	23.3
	large	49.2	72.2	68.2	60.8	46.6	34.5
DetNet-59	over all scales	40.2	61.7	57.0	49.6	36.2	25. 8
	small	23.9	41.8	36.8	29.8	17.7	10.5
	middle	43.2	65.8	61.2	53.6	39.9	27.3
	large	52.0	73.1	69.5	63	51.4	40.0

Table 3. Comparison of Average Precision (AP) of FPN on different IoU thresholds and different bounding box scales. AP_{50} is a effective metric to evaluate classification capability. AP_{85} requires accurate location of the bounding box prediction. Therefore it validates the regression capability of our approaches. We also illustrate AP at different scales to capture the influence of high resolution feature maps in backbone.

Models	scales	mAR	AR_{50}	AR_{60}	AR_{70}	AR_{80}	AR_{85}
ResNet-50	over all scales	52.8	80.5	74.7	64.3	46.8	34.2
	small	35.5	60.0	53.8	43.3	28.7	18.7
	middle	56.0	84.9	79.2	68.7	50.5	36.2
	large	67.0	95.0	90.9	80.3	63.1	50.2
DetNet-59	over all scales	56.1	83.1	77.8	67.6	51.0	38.9
	small	39.2	66.4	59.4	47.3	29.5	19.6
	middle	59.5	87.4	82.5	72.6	55.6	41.2
	large	70.1	95.4	91.8	82.9	69.1	56.3

Table 4. Comparison of Average Recall (AR) of FPN on different IoU thresholds and different bounding box scales. AR_{50} is a effective metric to show how many reasonable bounding boxes we find out (class agnostic). AR_{85} means how accurate of box location.

We also investigate the influence of DetNet for finding the small objects. As shown in Table 4, we make the detail statistics on averaged recall at different IoU threshold and scales. We conclude the table as follows:

- Compared with ResNet-50, DetNet-59 is more powerful for finding missing small objects, which yields 6.4 points gain (66.4 vs 60.0) in AR_{50} for the small object. DetNet keeps the higher resolution in deeper stages than ResNet, thus we can find smaller objects in deeper stages. Since we use up-sampling pathway in Fig. 1 A. Shallow layer can also involve context cues for finding small objects. However, AR_{85}@small is comparable (18.7 vs 19.6) between ResNet-50 and DetNet-59. This is reasonable. DetNet has no use for small object location, because ResNet based FPN has already used the large feature map for the small object.
- DetNet is good for large object localization, which has 56.3 (vs 50.2) in AR_{85} for large objects. However, AR_{50} in the large object does not change

too much (95.4 vs 95.0). In general, DetNet finds more accurate large objects rather than missing large objects.

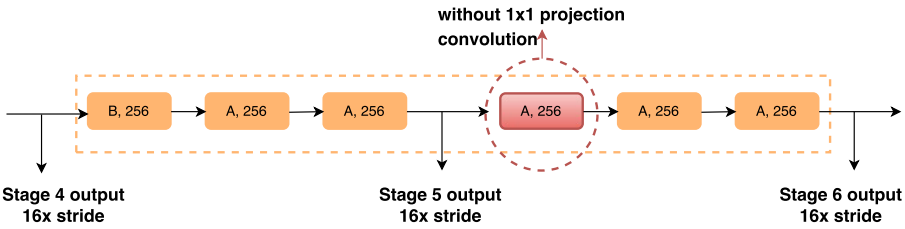

Fig. 3. The detail structure of DetNet-59-NoProj, which adopts module in Fig. 1 A to split stage 6 (while original DetNet-59 adopts Fig. 1 B to split stage 6). We design DetNet-59-NoProj to validate the importance of involving a new semantic stage as FPN for object detection.

4.5 Discussion

As mentioned in Section 3, the key idea of DetNet is a novel designed backbone specifically for object detection. Based on a prominent object detector like Feature Pyramid Network, DetNet-59 follows exactly the same number of stages as FPN while maintaining high spatial resolution. To discuss the importance of the backbone for object detection, we first investigate the influence of stages.

Since the stage-6 of DetNet-59 has the same spatial size as stage-5, a natural hypothesis is that DetNet-59 simply involves a deeper stage-5 rather than producing a new stage-6. To prove DetNet-59 indeed involves an additional stage, we carefully analyze the details of DetNet-59 design. As shown in Fig. 2 B. DetNet-59 adopts a dilated bottleneck with simple 1x1 convolution as the projection layer to split stage 6. It is much different from traditional ResNet, when spatial size of the feature map does not change, the projection will be simple identity in bottleneck structure(Fig. 2 A) rather than 1x1 convolution(Fig. 2 B). We break this convention. We claim the bottleneck with 1x1 convolution projection is effective to create a new stage even spatial size is unchanged.

To prove our idea, we involve DetNet-59-NoProj which is modified DetNet-59 by removing 1x1 projection convolution. Detail structure is shown in Fig. 3. There are only minor differences (red cell) between DetNet-59 (Fig. 2 D) and DetNet-59-NoProj (Fig. 3).

First we train DetNet-59-NoProj in ImageNet classification, results are shown in Table 5. DetNet-59-NoProj has 0.5 higher Top1 error than DetNet-59. Then We train FPN based on DetNet-59-NoProj in Table 5. DetNet-59 outperforms DetNet-59-NoProj over 1 point for object detection.

The experimental results validate the importance of involving a new stage as FPN used for object detection. When we use module in Fig. 2 A in our network, the output feature map is not much different from the input feature map, because output feature map is just sum of original input feature map and its transformation. Therefore, it is not easy to create a novel semantic stage for the network. While if we adopt module in Fig. 2 B, it will be more divergent between input and output feature map, which enables us to create a new semantic stage.

bacbone	Classification		FPN results					
	Top1 err	FLOPs	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DetNet-59	23.5	4.8G	**40.2**	61.7	43.7	23.9	43.2	52.0
DetNet-59-NoProj	24.0	4.6G	39.1	61.3	42.1	23.6	42.0	50.1

Table 5. Comparison of DetNet-59 and DetNet-59-NoProj. We report both results on ImageNet classification and FPN COCO detection. DetNet-59 consistently outperforms DetNet-59-NoProj, which validates the importance of the backbone design (same semantic stage) as FPN.

bacbone	Classification		FPN results					
	Top1 err	FLOPs,	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DetNet-59	23.5	4.8G	**40.2**	61.7	43.7	23.9	43.2	52.0
ResNet-50-dilated	–	6.1G	39.0	61.4	42.4	23.3	42.1	50.0

Table 6. Comparison of FPN results on DetNet-59 and ResNet-50-dilated to validate the importance of pre-train backbone for detection. ResNet-50-dilated means that we fine-tune detection based on ResNet-50 weights, while involving dilated convolution in stage-5 of the ResNet-50. We don't illustrate Top-1 error of ResNet-50-dilated because it can not be directly used for image classification.

Another natural question is that "what is the result if we train FPN initialized with ResNet-50 parameters, and dilate stage 5 of the ResNet-50 during detector fine-tuning (for simplify, we denote it as ResNet-50-dilated)". To show the importance of pre-train backbone for detection, we compare DetNet-59 based FPN with ResNet-50-dilate based FPN in Table 6. ResNet-50-dilated has more FLOPs than DetNet-59, while gets lower performance than DetNet-59. Therefore, we have shown the importance of directly training base-model for object detection.

4.6 Comparison to State of the Art

We evaluate DetNet-59 based FPN on MSCOCO [23, 20] detection test-dev dataset, and compare it with recent state-of-the-art methods listed in Table 7.

Fig. 4. Illustrative results of DetNet-59 based FPN.

Fig. 5. Illustrative results of DetNet-59 based Mask R-CNN.

Noticing that test-dev dataset is different from the mini-validation dataset used in ablation experiments. It has no disclosed labels and is evaluated on the server. Without any bells and whistles, our simple but efficient backbone achieves new state-of-the-art on COCO object detection, even outperforms strong competitors with ResNet-101 backbone. It is worth noting that DetNet-59 has only 4.8G FLOPs complexity while ResNet-101 has 7.6G FLOPs. We refer the original FPN results provided in Mask R-CNN [9]. It should be higher by using Detectron [7] repository, which will generate 39.8 mAP for FPN-ResNet-101.

To validate the generalization capability of our approach, we also evaluate DetNet-59 for MSCOCO instance segmentation based Mask R-CNN. Results are shown in Table. 8 for test-dev. Thanks for the impressive ability of our DetNet59, we obtain a new state-of-the-art result on instance segmentation as well.

Models	Backbone	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
SSD513 [24]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [24, 6]	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
Faster R-CNN +++ [10]	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN G-RMI [5] [14]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
RetinaNet [22]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FPN [9]	ResNet-101	37.3	59.6	40.3	19.8	40.2	48.8
FPN	**DetNet-59**	**40.3**	62.1	43.8	23.6	42.6	50.0

Table 7. Comparison of object detection results between our approach and state-of-the-art on MSCOCO test-dev datasets. Based on our simple and effective backbone DetNet-59, our model outperforms all previous state-of-the-art. It is worth noting that DetNet-59 yields better results with much lower FLOPs.

Models	Backbone	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
MNC [3]	ResNet-101	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [19] + OHEM [34]	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [19] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN [9]	ResNet-101	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	**DetNet-59**	**37.1**	60.0	39.6	18.6	39.0	51.3

Table 8. Comparison of instance segmentation results between our approach and other state-of-the-art on MSCOCO test-dev datasets. Benefit from DetNet-59, we achieve a new state-of-the-art on instance segmentation task.

Some of the results are visualized in Fig. 4 and Fig. 5. Detection results of FPN with DetNet-59 backbone are shown in Fig. 4. Instance segmentation results of Mask R-CNN with DetNet-59 backbone are shown in Figure 5. We only illustrate bounding boxes and instance segmentation no less than 0.5 classification scores.

5 Conclusion

In this paper, we design a novel backbone network specifically for the object detection task. Traditionally, the backbone network is designed for the image classification task and there is a gap when transferred to the object detection task. To address this issue, we present a novel backbone structure called DetNet, which is not only optimized for the classification task but also localization friendly. Impressive results have been reported on the object detection and instance segmentation based on the COCO benchmark.

[5] http://image-net.org/challenges/talks/2016/GRMI-COCO-slidedeck.pdf

References

1. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 (2016)
3. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3150–3158 (2016)
4. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. arXiv preprint arXiv:1703.06211 (2017)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 1, pp. 886–893. IEEE (2005)
6. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
7. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. https://github.com/facebookresearch/detectron (2018)
8. Girshick, R.B., Felzenszwalb, P.F., McAllester, D.: Discriminatively trained deformable part models, release 5 (2012)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. arXiv preprint arXiv:1703.06870 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
13. Huang, G., Liu, Z.: Densely connected convolutional networks
14. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. arXiv preprint arXiv:1611.10012 (2016)
15. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
18. Li, Y., He, K., Sun, J., et al.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems. pp. 379–387 (2016)
19. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 2359–2367 (2017)

20. Lin, T.Y., Dollár, P.: Ms coco api. https://github.com/pdollar/coco (2016)
21. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. arXiv preprint arXiv:1612.03144 (2016)
22. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. arXiv preprint arXiv:1708.02002 (2017)
23. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision. pp. 740–755. Springer (2014)
24. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision. pp. 21–37. Springer (2016)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
26. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
27. Mallat, S.: A wavelet tour of signal processing. Academic press (1999)
28. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. arXiv preprint arXiv:1711.07240 (2017)
29. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
30. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
33. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: Dsod: Learning deeply supervised object detectors from scratch. In: The IEEE International Conference on Computer Vision (ICCV). vol. 3, p. 7 (2017)
34. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 761–769 (2016)
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
36. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)
37. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2), 154–171 (2013)
38. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5987–5995. IEEE (2017)
39. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: Computer Vision and Pattern Recognition. vol. 1 (2017)

40. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017)
41. Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: European Conference on Computer Vision. pp. 391–405. Springer (2014)