Representation Sharing for Fast Object Detector Search and Beyond

Yujie Zhong, Zelu Deng, Sheng Guo, Matthew R. Scott, and Weilin Huang*

Malong LLC
{jaszhong, zeldeng, sheng, mscott, whuang}@malong.com

Abstract. Region Proposal Network (RPN) provides strong support for handling the scale variation of objects in two-stage object detection. For one-stage detectors which do not have RPN, it is more demanding to have powerful sub-networks capable of directly capturing objects of unknown sizes. To enhance such capability, we propose an extremely efficient neural architecture search method, named Fast And Diverse (FAD), to better explore the optimal configuration of receptive fields and convolution types in the sub-networks for one-stage detectors. FAD consists of a designed search space and an efficient architecture search algorithm. The search space contains a rich set of diverse transformations designed specifically for object detection. To cope with the designed search space, a novel search algorithm termed Representation Sharing (RepShare) is proposed to effectively identify the best combinations of the defined transformations. In our experiments, FAD obtains prominent improvements on two types of one-stage detectors with various backbones. In particular, our FAD detector achieves 46.4 AP on MS-COCO (under single-scale testing), outperforming the state-of-the-art detectors, including the most recent NAS-based detectors, Auto-FPN [42] (searched for 16 GPU-days) and NAS-FCOS [39] (28 GPU-days), while significantly reduces the search cost to 0.6 GPU-days. Beyond object detection, we further demonstrate the generality of FAD on the more challenging instance segmentation, and expect it to benefit more tasks.

1 Introduction

Object detection is a fundamental task in computer vision [31, 24, 17, 30, 18, 23, 15, 38, 45], but it remains challenging due to the large variation in object scales. To handle the scale variation, a straightforward method is to utilize multi-scale image inputs [34, 35], which usually lacks efficiency. A line of more efficient methods is to tackle the scale variation on the intermediate features [24, 17]. For example, Feature Pyramid Networks (FPN) [17] is a representative work that implements the detection of objects with different scales in multiple levels of feature pyramids. On the other hand, recent works also attempt to improve the detectors from the perspective of receptive fields (RFs) [23, 15]. They enhance the scale-awareness of the detectors by having multi-branch transformations with

* Corresponding author: whuang@malong.com

Table 1. Comparison against other NAS methods for object detection on MS-COCO [19]. *Trans.* indicates the number of transformation types in the search space ('skip-connect' is excluded). *Counterpart* denotes the baseline detectors (and backbone) for direct comparison. * means only the dilation rates are varied.

Method	Search Method	Trans.	GPU-days	Counterpart	Relative AP Imp.
NAS-FPN [7]	RL	2	> 100	RetinaNet (Res-50)	$\uparrow$ 2.9
DetNAS [1]	EA	4	44	FPN (ShuffleNetv2)	$\uparrow$ 2.0
NATS-det [26]	EA	$9^\star$	20	RetinaNet (Res-50)	$\uparrow$ 1.3
Auto-FPN [42]	Gradient	6	16	FPN (Res-50)	$\uparrow$ 1.9
NAS-FCOS [39]	RL	6	28	FCOS (Res-50)	$\uparrow$ 1.7
SM-NAS [43]	EA	-	> 100	-	-
FAD (ours)	Gradient	12	**0.6**	FCOS (Res-50)	$\uparrow$ 1.7

different combinations of kernel sizes and/or dilation rates. Then the features of different RFs are aggregated to enrich the information of different scales at each spatial location.

An object detector often has a backbone network followed by the detection-specific sub-networks (i.e. heads), which play an important role in object detection. The sub-networks compute the deep features which are used to directly predict the object category, localization and size. Unlike two-stage detectors in which the sub-networks operate on the fixed-size feature maps computed from each object proposal, generated by a region proposal network with ROI-pooling [31], the sub-networks in one-stage detectors should be capable of 'looking for' objects of arbitrary sizes directly. It becomes more challenging for an anchor-free detector. Because the multi-scale anchor boxes can be considered as a way to explicitly handle various sizes and shapes of objects, whereas an anchor-free detector only predicts a single object at each spatial location, without any prior information about the object size. Therefore, for one-stage detectors, especially the anchor-free ones, the capability of the sub-networks for capturing the objects with large scale variation becomes the key. In this work, we aim to enhance the power of the sub-networks in one-stage detectors, by searching for the optimal combination of the RFs and convolutions in a learning-based manner.

Neural Architecture Search (NAS) has gained increasing attention. It transfers the task of neural networks design from a heuristics-guided process to an optimization problem. Recently, it has been shown that NAS can achieve prominent results on object detection [7, 1, 42, 26, 43, 39]. In most of the work, the operations in the search space are directly extended from those used for image classification [48, 22] with limited variation on dilation rates. Therefore, their search spaces with respect to *transformations* are relatively limited, as listed in Table 1. Apart from the combination of RFs, we also investigate the importance of the diversity of the transformations in NAS search space for object detection. However, searching through such a large number of candidate transformations can be computationally expensive, especially for the RL-based [48, 27] and EA-based [29] approaches. Additionally, this problem can be more signif-

icant for object detection than image classification, due to the more complicated pipelines with larger input images.

To this end, we propose a computation-friendly method, named Fast And Diverse (FAD), to search for the task-specific sub-networks in one-stage object detectors. FAD consists of a designed search space and an efficient search algorithm. We first design a rich set of diverse transformations tailored for object detection, covering multiple RFs and various convolution types. To learn the optimal combinations more efficiently, a search method via *representation sharing* (RepShare) is proposed accordingly. By sharing intermediate representations, the proposed RepShare significantly reduces the searching time and memory cost for the architecture search. Furthermore, we propose an efficient method to reduce the interference between the transformations sharing the same representations, and at the same time, alleviate the degradation of search quality caused by RepShare.

To demonstrate the effectiveness of the proposed method, we redesign the sub-networks for modern one-stage object detectors, and propose a searchable module for replacement. *The architecture search for the module is extremely efficient using our FAD, which is more than* $25\times$ *faster than the fastest NAS approach for object detectors so far, while achieving a comparable AP improvement* (see Table 1). With ResNeXt-101 [41] as the backbone, our FAD detector achieves 46.4 AP on the MS-COCO [19] *test-dev* set using a single model under single-scale testing, without using any additional regularization or modules (e.g. deformable conv [3]). Moreover, we show that FAD can also benefit more challenging tasks, such as instance segmentation. The contributions of this work are summarized as:

- We present a novel method, named Fast And Diverse (FAD), to search meaningful transformations in the task-specific sub-networks for one-stage object detection. The search space is designed specifically for object detection, and we empirically investigate the importance of the RFs coverage and convolution types for object detection.
- We propose an efficient search method with a novel representation sharing (RepShare) algorithm, which can significantly reduce the search cost in both time and memory usage, e.g. being more than $25\times$ faster than all previous methods. To ensure the search quality, a new method is introduced to decouple the transformation selection from the shared representations.
- To evaluate our methods, we design a searchable module for one-stage object detection and instance segmentation. Extensive experiments show that our FAD detector obtains consistent performance improvements on different detection frameworks with various backbones, and even has fewer parameters.

2 Related Work

2.1 Object Detection and Instance Segmentation

In general, object detectors can be categorized into two groups: two-stage detectors and one-stage detectors. Modern two-stage detectors [31, 2] first adopt a

regional proposal network (RPN) to generate a set of object proposals, which are then fed to the R-CNN heads for object classification and bounding box regression. On the other hand, one-stage object detectors [30, 24, 18] directly perform object classification and box regression simultaneously at each spatial location on the feature maps produced from a backbone network. Taking RetinaNet as an example, it consists of a backbone network with a feature pyramid network (FPN) [17] and two sub-networks for classification and bounding box regression. Recent works attempt to get rid of hand-designed anchor boxes while achieving comparable performance [14, 4, 47, 38]. For instance, FCOS [38] additionally predicts a centerness score which indicates the distance of current location to the center of the corresponding object, and can even outperform RetinaNet.

Receptive fields (RF). RF is proved to be very important for object detectors [23, 15]. For instance, Liu et al. [23] designed a combination of kernel sizes and dilation rates, to simulate the impact of the eccentricities of population receptive fields in human visual cortex. TridentNet [15] tackles the scale variation using multi-branch modules with different dilation rates. In this work, we aim to search for an optimal combination of different conv layers and dilation rates jointly.

Instance segmentation. Instance segmentation is closely related to object detection, and the dominant instance segmentation methods often have two stages [10, 13]: they first detect the objects in an image, and then predict an object mask on each detected region. Mask R-CNN [10] is a representative work in this paradigm, which has an additional mask head on top of Faster R-CNN [31] to perform mask prediction on each object proposal. In this work, we apply the proposed FAD search method to instance segmentation, which has not been explored previously.

2.2 Neural Architecture Search

Recent attention has been moved from network design by hand to neural architecture search (NAS) [48, 27, 21, 25, 22]. A stream of efficient NAS methods is the differentiable NAS [25, 22]. In particular, DARTS [22] significantly increases the search efficiency by relaxing the categorical choice of operation to be continuous, so that the architecture can be optimized by gradient descent. In this work, we develop an efficient NAS algorithm for object detectors, by fast searching the optimized transformations.

NAS for Object Detection NAS has been applied to many vision tasks apart from image classification, such as object detection [7, 1, 42, 26]. For example, NAS-FPN [7] uses a RL-based NAS to search for an optimal FPN [17] on the RetinaNet. DetNAS [1] aims at finding the optimal shuffle-block-based backbone network in object detectors using an evolution algorithm [8, 28]. A channel-level NAS is proposed in NATS [26] to search for the backbone in object detectors. Alternatively, some recent works search for the detection-specific parts rather than the backbone for object detection. For instance, Auto-FPN [42] searches

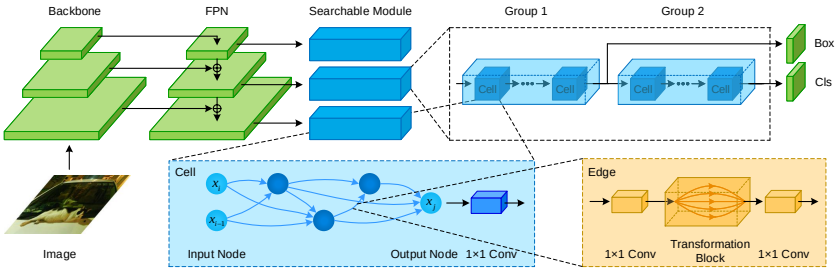

Fig. 1. Search space of FAD for one-stage object detectors. The backbone and FPN [17] in detectors remain the same, while each FPN level is connected to a searchable module. It consists of two groups of cells, with same cell architectures within each group. In a cell, the edges connecting nodes consist of two standard 1×1 conv layers and a transformation block in between. The cell structures and the transformations are to be searched. Each edge might have different RFs, resulting in combinations of RFs at each node which enrich the features for capturing information of various scales.

for a FPN structure and head structures. SM-NAS [43] also searches for two-stage detectors by first conducting a structural-level search and then a modular-level search. Instead of exploring novel structures, CR-NAS [16] aims to re-allocate the computation resources in the backbone. NAS-FCOS [39] is a FCOS-based detector in which the structure of its FPN and the following sub-networks are computed using RL-based NAS. In this work, we design the search space specifically, and propose the FAD method to search for the sub-networks in one-stage detectors.

3 Fast Diverse-Transformation Search

3.1 Search Space of FAD

One-stage detectors like RetinaNet [18] and FCOS [38] consist of a backbone network with FPN [17] and two parallel sub-networks for object classification and bounding box regression, respectively. In this section, we design a searchable module to replace the commonly-used sub-networks. This module is searched by the proposed FAD, and can be adapted to one-stage object detectors that follow a similar structure as RetinaNet [18] in a plug-and-play fashion. We then describe the novel search space of FAD which is tailored for object detection, including a variety of diverse transformations with different RFs.

Object Detector with FAD As shown in Figure 1, the proposed searchable module is comprised of two groups of cells, which are connected sequentially with a shortcut from the input of the module to that of the second group. The module outputs both object classification and bounding box prediction. The architectures and parameters are shared across different FPN levels.

Classification and regression. In FAD, the bounding box prediction is performed on the output of the first group, while the classification is computed from the output of the second group. The intuition behind this design is that the two tasks should not be implemented on the exactly same feature maps due to different objectives: bounding box regression needs to focus on the local detailed information, while object classification is implemented on the features with more semantic information (i.e. the feature maps on deeper layers). Therefore, we perform bounding box regression on the output of the first group.

Design of Search Space In the following, we describe the design of the search space for FAD, which is inspired by the insights from modern neural architectures [37, 11] and object detectors [23, 15]. Three important considerations in our design are the coverage of RFs, the diversity in convolution types and the computational efficiency.

Groups and cells. A group contains M repeated cells, and each cell is defined as a module that contains multiple nodes and edges. Similar to [22, 20], each cell is formulated as a directed acyclic graph of nodes. Each node is a stack of feature maps and each edge is an atomic block for search. In this work, we empirically set the number of nodes in each cell to be 3, excluding the input and output nodes. In our design, an edge consists of two 1×1 conv layers f and a transformation block T between the two (Figure 1 bottom-right). The transformation block contains a set of candidate transformations which will be described in Sec. 3.2. Each conv layer in the transformation is followed by a group-normalization layer [40] and a ReLU. Given a node x_j, all the predecessors x_i connected to it, and an edge pointing from x_i to x_j, we can have the following expression:

$$x_j = \sum_{i<N}^{N} f_{i,j}^{c',c}(T_{i,j}^{c',c'}(f_{i,j}^{c,c'}(x_i))), \tag{1}$$

where $f_{i,j}^{c,c'}$ and $f_{i,j}^{c',c}$ are the two 1×1 conv layers, with one transforming the input channel c to the channel used in the transformation block $T_{i,j}^{c',c'}$ and the other vice versa. x_j is computed based on N total number of predecessors. The two 1×1 convolution enable a flexibility in the channel size in T, similar to the inception module [37], while maintaining the same channel size for all the nodes. We empirically found that maintaining a relatively large channel size for nodes is beneficial to the performance. The representations of the intermediate nodes in a cell are concatenated and passed to a 1×1 conv layer to reduce the number of channels back to c. This additional conv layer ensures the consistent channel size between the input and output of each cell. Furthermore, the idea of having two groups of cells enables a larger flexibility for the architecture search, i.e. a larger search space. Within each group, the cells share the same structure. Therefore, once the search is completed, the cells in each group can be repeated for multiple times, offering a great scalability in architecture depth.

Diverse transformations. Our initial design of the candidate transformations covers 4 different sizes of RFs (Figure 2 bottom left). In particular, for the transformations that are responsible for a RF larger than 5, we use more efficient operations by having a base filter followed by a dilated convolution which spreads out the base filter to reach larger RFs. Moreover, the dilated conv layers are depthwise separable [32, 12], in order to keep the computation efficient. The memory-efficient design introduced in Section 3.2, allows us to include more types of convolutions. Hence we have two streams of transformations: the standard conv and the depthwise separable conv. Namely, for the 6 transformations shown in the bottom-left corner of Figure 2, the 'conv' layers can be all standard convolution or depthwise separable convolution.

There are no pooling layers involved in the search space as we empirically found that they are not helpful in our scenario. This is probably because the spatial resolution of the feature maps remains the same in the sub-networks. Moreover, skip-connection is not included in the transformation. Lastly, a 'none' path, indicating the importance of input edges with respect to each node, is added to the transformation block. In summary, the proposed transformation block contains 13 distinct transformations in total, including 2 types of conv layers and 3 dilation rates, and covering 4 sizes of RFs, as illustrated in Figure 2. Therefore, we build a meaningful search space with strongly diverse transformations. The resultant search space has roughly 2.3×10^{13} unique paths in total, with one cell per group in search time.

FAD for Instance Segmentation We expect that the mask prediction task can also benefit from the combination of RFs and diverse transformations. With minimal modification, FAD readily applicable to general instance segmentation frameworks, e.g. Mask R-CNN [10] and Mask Scoring R-CNN [13]. Specifically, we replace the conv layers before the deconvolutional layer in the mask head by the proposed searchable module, and search for its architecture in an end-to-end fashion. The search space is degined by following that of object detectors.

3.2 Fast Search with Representation Sharing

In this section, we propose a novel algorithm to significantly reduce the search cost in both time and memory, followed by the description of search procedure.

Representation Sharing The proposed acceleration method for architecture search, named RepShare, is performed in two steps: filter decomposing and intermediate representation sharing. We elaborate these two steps in the following.

Decomposing large filters. As proposed in [33], filters with large kernel sizes can be replaced by multiple 3×3 filters. For example, a stack of three 3×3 filters in fact has an equivalent size of receptive filed as a 7×7 filter. The stacked filters have the advantages of fewer parameters and more non-linearities in between for learning more discriminative representations. Following this intuition, we

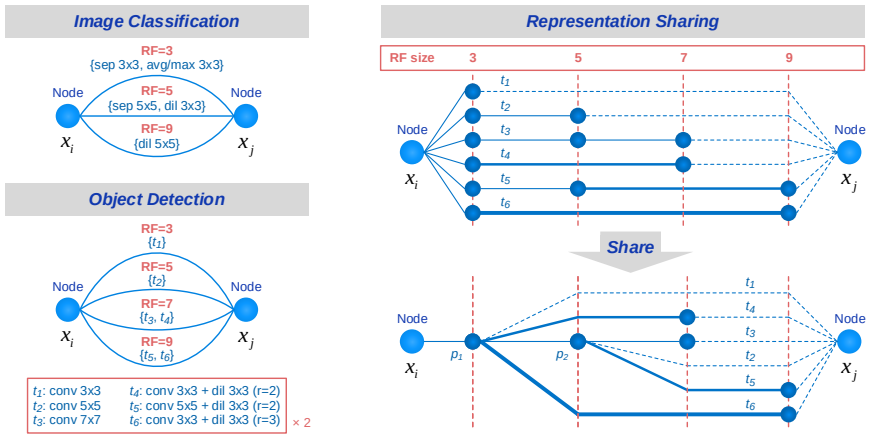

Fig. 2. Transformations and representation sharing. Left: comparison between the transformations used for image classification and those proposed for object detection in the search space. The proposed transformations are listed at the bottom. *conv* can be the standard or the depthwise separable convolution. **Right**: RepShare. Each sphere and solid line denotes a representation and a conv layer, respectively. First, large filters are decomposed into stacks of 3×3 filters. Second, p_1 and p_2 are shared across transformations. Note that the 1×1 conv layers are not shown for simplicity.

decompose the filters with large kernel size and construct a transformation block only containing filters of size 3×3 (t_1 to t_6 shown in Figure 2 top-right). However, the replacement of large filters with stacks of small ones significantly increases the memory overhead during the search. Taking the proposed transformations as an example, more than twice intermediate representations are generated after the decomposition.

Representation Sharing. To reduce this memory overhead, we further propose a novel approach. Namely, for each receptive field (RF) level, all the intermediate representations that are not directly connected to node x_j are shared (Figure 2 bottom-right). To be specific, we denote t_3 in top-right of Figure 2 as the stem. In the stem, there are 3 intermediate representations having different sizes of RFs with respect to the node x_i. We merge the transformations by sharing the intermediate representations in the stem. For example in Figure 2 (top-right), to merge the t_1 into the stem, we directly connect the first intermediate representation in the stem to node x_j, and therefore the original t_1 (conv 3×3) transformation is replaced by this new transformation. Specifically, the RepShare reduces the number of representations computed in each transformation block from 26 to 12. Therefore, it can significantly speed up the search process. Moreover, the search speed is further boosted by the memory-efficiency of RepShare since the search can be done using a single GPU, which avoids the computational overhead introduced by training with multiple GPUs (e.g. parameter update).

Relation to other efficient search methods. The proposed RepShare has similar spirits to some recent approaches. For instance, parameter sharing introduced in [27] takes the advantage of sharing the same sets of parameters among child models to greatly speed up the search in RL-based NAS methods. It is inspired by parameter inheritance [29] which also reuses the same parameters for child models across mutation to avoid training from scratch. RepShare is more than using the same parameters, but also the same computation. Furthermore, apart from accelerating the search, RepShare further reduces the memory consumption. Single-path NAS [36] also share computations, but is different from ours. It considers a small kernel (e.g. 3×3) as the core of a large one (e.g. 5×5), and uses a learnable threshold to compare the importance of the two kernels, and selects the optimal one.

Decoupling Shared Representations Similar to parameter sharing described in [27] in which child models are coupled to some extent due to reusing the same weights, RepShare also exhibits such behaviour. In RepShare, transformations sharing the same representations might interfere with each other, and thus the parameters directly corresponding to the shared representations are not well optimized in the search. It causes that those transformations are difficult to outstand in the architecture derivation. For example, in Figure 2 (bottom-right), two intermediate representations are shared. Namely, p_1 is shared across all six transformations and p_2 is shared across t_2, t_3 and t_5. Due to the coupling effect (i.e. interference between transformations), t_1 and t_2 are not able to learn the optimal parameters on their own, which may degrade the search quality. Notably, this effect mainly happens on t_1 and t_2, since their outputs are exactly the shared representations; while other transformations (t_3 to t_6) have the flexibility to compensate this effect due to additional operations on the share representations.

Decoupling with extra functions. To address this issue in RepShare, we propose a simple yet effective method to decouple the transformations (that directly depend on the shared representations, i.e. t_1 and t_2) from the shared representations. Namely, an additional function H is applied between each shared representation and its corresponding transformation output. With this additional function, for example, the output of t_1 is no longer p_1, but $H(p_1)$. In this case, t_1 and t_2 are decoupled from p_1 and p_2, respectively. For the choice of H, we use a standard 1×1 conv layer followed by a ReLU. This light-weight extra function produces minimal computational overhead and is applied to both conv streams (i.e. the standard and depthwise separable convolution streams).

Optimization and Deriving Architectures In a cell, each edge contains a transformation block in which the final transformation is determined from a set of candidates illustrated in Figure 2. In order to search using back-propagation, we follow the continuous relaxation for the search space as [22], and adapt it to the proposed RepShare paradigm. For each of the two streams (Figure 2 bottom-right) in the transformation block, the output of a transformation ($T_{i,j}$)

is essentially the sum of all the intermediate representations multiplied with corresponding α. Therefore, we can have:

$$T_{i,j}(x'_i) = \sum_{p \in P} \frac{\exp\left(\alpha_{i,j}^{p}\right)}{\sum_{p' \in P} \exp\left(\alpha_{i,j}^{p'}\right)} \, p, \tag{2}$$

where x'_i is the output of the first 1×1 conv layer in the transformation block. p and p' are the intermediate representations out of all representations P. α^p is the α corresponding to p.

Optimization and derivation of discrete architectures. During the architecture search, α and the network weights w are jointly optimized in a bilevel optimization scheme, as in [22, 20]. In particular, the first-order approximation is adopted. At the end of the search, a discrete architecture is decoded by retaining one transformation per edge and two input edges for each node based on the largest α in each transformation block. Since the intermediate representations are selected instead of operations, they should then be mapped to the corresponding actual transformations in the derived architecture, i.e. the transformations in Figure 2 (top-right).

4 Experiments

In this section, the proposed FAD is evaluated in two tasks: object detection and instance segmentation. In the Supplementary Material (SM), we further conduct experiments for image classification to analyze the effect of decoupling in RepShare.

4.1 Object Detection

Implementation details. Although the proposed module can be adopted to different one-stage object detectors, we perform the architecture search using FAD on FCOS [38], due to its efficiency. The search is conducted on the PASCAL VOC [5]. We also perform the search directly on MS-COCO [19] and make comparisons in Table 2. More implementation details, including the search and the detector training, can be found in the SM.

Ablation Study We conduct ablation study on the search cost, search spaces, as well as different backbones and detectors. More studies on the marco-structure of the module, and network width and depth are presented in SM.

Search cost. The time required for a complete architecture search using our FAD is 0.6 GPU-days. A single TITAN XP is used for the search. Table 1 compares the search cost of FAD against other NAS-based methods for object detection. As we can see, the search speed for FAD is at least 25× faster than other recent

Table 2. Comparison for the architecture search. *Memory* and *bs* denotes the memory usage and images per GPU. Both *Subset* and *Full* refer to the proposed search space. *Sep.* and *Std.* mean that only depthwise and standard conv are used, respectively. ResNet-50 is used as the backbone. Results are obtained on the MS-COCO *minival* split. All the searches are performed on VOC, expect for † which is on MS-COCO.

Method	RepShare	Search Space	Trans.	RFs	Memory (G)	GPU-days	AP
FCOS [38]	-	-	-	-	-	-	38.6
Random	-	Full	12	3,5,7,9	-	-	39.0
FAD	✗	DARTS [22]	7	5,7,9	~ 10 ($bs = 4$)	0.4	39.0
FAD	✓	Subset 1	4	3,5	~ 7 ($bs = 4$)	0.25	39.2
FAD	✓	Subset 2	8	3,5,7	~ 11 ($bs = 4$)	0.5	39.7
FAD	✓	Sep. only	6	3,5,7,9	~ 10 ($bs = 4$)	0.36	39.5
FAD	✓	Std. only	6	3,5,7,9	~ 9.5 ($bs = 4$)	0.4	39.9
FAD	✓	w/o decouple	12	3,5,7,9	~ 12 ($bs = 4$)	0.6	40.0
FAD	✓	Full	12	3,5,7,9	~ 12 ($bs = 4$)	0.6	**40.3**
FAD	✗	Full	12	3,5,7,9	~ 9 ($bs = 1$)	2.3	**40.3**
FAD †	✓	Full	12	3,5,7,9	~ 9 ($bs = 4$)	5.5	**40.3**

approaches, while achieving a similar relative AP improvement on MS-COCO. Meanwhile, the architecture explored by FAD is scalable in depth by simply adding the repetitive cells in the groups, which provides greater flexibility to the module.

Search space. To demonstrate the superiority of the proposed search space, we reuse the same search procedure but replace the proposed search operations with that in DARTS [22], which are listed in Figure 2 (top-left). Note that the depthwise separable convolution is doubled in DARTS, and hence the RFs change accordingly. As we can see from Table 2, the operations used in DARTS only bring a marginal improvement of 0.4 AP, compared to the original FCOS, while *the proposed transformations improve the performance significantly, from* 38.6 *to* 40.3. To further study the importance of the full transformation set, we search by using two transformation subsets. Namely, the two subsets contain transformations with the RFs smaller than 7 and 9, respectively. Our results show that with less transformations in the search space, the performance degrades accordingly. Moreover, we search by using only one type of convolution (either the standard or the depthwise separable) for the conv layers with dilation rate of 1. Not surprisingly, both of them fail to achieve a similar performance as the full search space. This illustrates the power of the proposed transformations which fully benefit from the better combinations of RFs and convolution types. Besides, the performance slightly degrades without decoupling. More results on decoupling can be found in the SM. Another observation is that the proxyless search on MS-COCO can achieve similar performance on detection, but it takes much longer search time. Hence, we use the architecture searched on VOC for object detection for the rest of this work.

Table 3. FAD on different detectors and backbones. The → indicates the change from original detector to FAD. *Dim.* is the channel size in the subnets, or c' in the transformation block in FAD. Results are obtained on MS-COCO *minival*.

Method	Backbone	Dim.	Params (M)	FLOPs (G)	AP
FCOS	MobileNetV2	256 → 96	9.8 → 9.0	124 → 108	31.3 → 32.7
	Res-50	256 → 96	32.2 → 31.5	201 → 185	38.6 → 40.3
	Res-101	256 → 96	51.2 → 50.4	277 → 261	43.0 → 44.2
	Res-X-101	256 → 96	90.0 → 89.2	439 → 423	44.7 → 45.8
	Res-X-101	256 → 128	90.0 → 91.2	439 → 465	44.7 → **46.0**
RetinaNet	Res-50	256 → 96	33.8 → 33.0	234 → 218	36.1 → 37.7
	Res-101	256 → 96	52.7 → 52.0	310 → 294	37.7 → 39.4
	Res-X-101	256 → 128	91.5 → 92.7	472 → 498	39.8 → **41.6**
Subnet only	-	256 → 96	4.9 → 4.1	105 → 89	-

In addition, our FAD is also compared with the 'random' baseline. Namely, a transformation is randomly sampled in each block and two edges are randomly sampled for each node. It can be found that the proposed FAD indeed finds much better architectures. The last conclusion to draw in Table 2 is that, comparing to the search without RepShare, *RepShare enables an almost* 4× *faster search with only one third of the GPU memory usage*, without harming the performance.

Adaptation to different backbone networks. We replace the ResNet-50 in the detector by using three different networks: MobileNetV2 [12], ResNet-101 [11] and ResNeXt-101 [41]. As shown in Table 3, our FAD obtains a consistent improvement (about 1.4 AP on average) for all the backbones compared, with even fewer parameters and FLOPs. This indicates that the architecture of FAD generalizes well to the backbone networks with different capacity. A direct comparison on the sub-networks (without the backbone and FPN) shows a 16.3% and 15.2% decrease on the number of parameters and the FLOPs. Hence, we can conclude that the performance gain is obtained from the better architecture searched rather than the network capacity itself.

Transferability. Our FAD is expected to be readily applicable to different types of one-stage object detectors (with the two-subnet structure). To examine this property, we further plug the proposed searchable module into RetinaNet [18]. Table 3 reveals that FAD can also improve the performance of RetinaNet by a large margin even with fewer parameters. Therefore, we see that the searched sub-networks can boost the performance of different types of detectors (and potentially more powerful detectors in the future) in a plug-and-play fashion.

Comparison with the state-of-the-art We compare FAD with the state-of-the-art object detectors on the MS-COCO *test–dev* split, including some recent NAS-based object detectors. All the methods are evaluated under the single-model and single-scale setting. Table 4 shows that, by having 128 channels in

Table 4. Comparison with the state-of-the-art object detectors on the MS-COCO *test-dev* split (including concurrent work [9, 46, 44, 39]). FCOS [38] is used as the base detector for FAD. All the results are tested under the single-scale and single-model setting. Note that models using additional regularization method [6] and deformable convolution [3] are excluded in the table (except for NAS-FCOS [39]).

Two-stage detectors	Backbone	**AP**	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
TridentNet[15]	ResNet-101	42.7	63.6	46.5	23.9	46.6	56.6
Auto-FPN [42]	ResNeXt-64x4d-101	44.3	-	-	-	-	-
SM-NAS: E5 [43]	Searched	45.9	64.6	49.6	27.1	49.0	58.0
Hit-Detector [9]	Searched	44.5	-	-	-	-	-
One-stage detectors							
RetinaNet [18]	ResNeXt-101	40.8	61.1	44.1	24.1	44.2	51.2
CenterNet511 [4]	Hourglass-104	44.9	62.4	48.1	25.6	47.4	57.4
FSAF [47]	ResNeXt-64x4d-101	42.9	63.8	46.3	26.6	46.2	52.7
FCOS [38]	ResNeXt-64x4d-101	44.7	64.1	48.4	27.6	47.5	55.6
FreeAnchor [45]	ResNeXt-101	44.9	64.3	48.5	26.8	48.3	55.9
SAPD [46]	ResNeXt-64x4d-101	45.4	65.6	48.9	27.3	48.7	56.8
ATSS [44]	ResNeXt-64x4d-101	45.6	64.6	49.7	28.5	48.9	55.6
NAS-FPN [7] (7 @ 384)	ResNet-50	45.4	-	-	-	-	-
NAS-FCOS [39]	ResNeXt-64x4d-101	46.1	-	-	-	-	-
FAD @ 96	ResNet-101	44.1	62.7	47.9	26.8	47.1	54.6
FAD @ 128	ResNet-101	44.5	63.0	48.3	27.1	47.4	55.0
FAD @ 128	ResNeXt-64x4d-101	46.0	64.9	50.0	29.1	48.8	56.6
FAD @ 128-256	ResNeXt-64x4d-101	**46.4**	65.4	50.4	29.4	49.3	57.4

the first group and 256 in the second (with 98.3M parameters), FAD @128-256 achieves 46.4 AP which surpasses all the recent object detectors, including two concurrent work, NAS-FCOS [39] and Hit-Detector [9]. Note that NAS-FCOS includes the deformable convolution [3] in the search space, which is not considered in other NAS-based detectors (including our FAD), and it is well-known for giving large AP improvements. On the other hand, the search of FAD is almost 50× faster than that of NAS-FCOS on the same dataset (i.e. VOC).

Searched Architectures The derived architectures by FAD are presented in Figure 3. We have two interesting observations. First, the edges correspond to a mixture of RFs (especially for the cell group for classification) and convolution types, which again validates our motivation. Another important insight is that the transformations with large RFs (i.e. 7 and 9) appear near the input node, while those with small RFs (i.e. 3 and 5) are closer to the output node. This is consistent with the DetNAS architecture explored in [1].

4.2 Instance Segmentation

To showcase the generality of the proposed FAD, we apply it to another useful task – instance segmentation. Different from object detection, only one group of

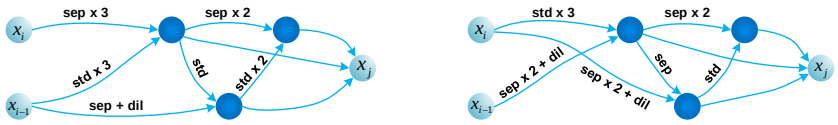

Fig. 3. Architectures searched for object detection. The left and right cells are for the first and second group, respectively. *std*, *sep* and *dil* denote the standard, depthwise separable and dilated conv.

Table 5. Comparison on instance segmentation *mask* AP on the MS-COCO *minival* split. *P.* is for parameters (M) and *F.* is for FLOPs (G).

Method	Backbone	P.	F.	**AP**	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [10]	Res-50	44.3	285	34.2	55.7	36.3	15.4	36.8	50.9
	Res-101	63.3	362	36.1	58.1	38.3	16.4	38.9	53.4
Mask FAD	Res-50	44.4	287	35.5	56.8	37.9	16.0	38.4	52.7
	Res-101	63.4	364	37.0	58.6	39.5	17.0	39.8	54.9
MS R-CNN [13]	Res-50	60.7	326	35.6	56.2	38.2	16.6	37.8	52.0
	Res-101	79.6	402	37.4	58.3	40.2	17.5	40.2	54.4
MS FAD	Res-50	60.8	328	36.3	56.3	39.2	16.1	38.8	53.4
	Res-101	79.7	404	**38.0**	58.7	41.0	17.6	41.0	55.1

cell is searched in the mask head. The search is conducted on MS-COCO, which takes 2.6 GPU-days. For a fair comparison, we exactly follow [10, 13] for training the searched networks. The search and training details are described in the SM.

Results Table 5 shows that, with similar number of parameters and FLOPs, all FAD outperform their counterparts with same backbones on both Mask R-CNN and MS R-CNN. Notably, Mask FAD has relatively larger improvements in terms of AP_M and AP_L (e.g. 1.6 and 1.8 AP on ResNet-50) than AP_S (0.6 AP), possibly due to the transformations with larger RFs. Another surprising result is that Mask FAD (ResNet-50) achieves similar AP as MS R-CNN (ResNet-50), i.e. 35.5 vs. 35.6, despite a simpler pipeline and 26.9% fewer parameters. The improvements are prominent since we only modify the mask head architecture which only accounts for 2.25M parameters, i.e. 2.8% to 5% of the whole networks.

5 Conclusion

In this work, we propose FAD to efficiently search for better sub-networks with diverse transformations and optimal combinations of RFs for one-stage object detection and instance segmentation. To demonstrate the effectiveness of the proposed search space and search method, we design a searchable module for the two tasks at hand (and potentially applicable to other tasks). Extensive experiments show that the architectures searched by our FAD can consistently outperform their counterparts on different detectors and segmentation networks.

References

1. Chen, Y., Yang, T., Zhang, X., Meng, G., Pan, C., Sun, J.: DetNAS: Backbone search for object detection. arXiv preprint arXiv:1903.10979 (2019)
2. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object detection via region-based fully convolutional networks. In: Advances in neural information processing systems. pp. 379–387 (2016)
3. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
4. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: Keypoint triplets for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6569–6578 (2019)
5. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. International journal of computer vision **88**(2), 303–338 (2010)
6. Ghiasi, G., Lin, T.Y., Le, Q.V.: DropBlock: A regularization method for convolutional networks. In: Advances in Neural Information Processing Systems. pp. 10727–10737 (2018)
7. Ghiasi, G., Lin, T.Y., Le, Q.V.: NAS-FPN: Learning scalable feature pyramid architecture for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7036–7045 (2019)
8. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of genetic algorithms, vol. 1, pp. 69–93. Elsevier (1991)
9. Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X., Xu, C.: Hit-Detector: Hierarchical trinity architecture search for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11405–11414 (2020)
10. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
11. 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)
12. 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)
13. Huang, Z., Huang, L., Gong, Y., Huang, C., Wang, X.: Mask scoring R-CNN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6409–6418 (2019)
14. Law, H., Deng, J.: CornerNet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 734–750 (2018)
15. Li, Y., Chen, Y., Wang, N., Zhang, Z.: Scale-aware trident networks for object detection. arXiv preprint arXiv:1901.01892 (2019)
16. Liang, F., Lin, C., Guo, R., Sun, M., Wu, W., Yan, J., Ouyang, W.: Computation reallocation for object detection. arXiv preprint arXiv:1912.11234 (2019)
17. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)

18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
19. 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)
20. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 82–92 (2019)
21. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)
22. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
23. Liu, S., Huang, D., et al.: Receptive field block net for accurate and fast object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 385–400 (2018)
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. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. In: Advances in neural information processing systems. pp. 7816–7827 (2018)
26. Peng, J., Sun, M., Zhang, Z., Tan, T., Yan, J.: Efficient neural architecture transformation searchin channel-level for object detection. arXiv preprint arXiv:1909.02293 (2019)
27. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268 (2018)
28. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4780–4789 (2019)
29. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 2902–2911. JMLR. org (2017)
30. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
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. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification. Ph. D. dissertation (2014)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
34. Singh, B., Davis, L.S.: An analysis of scale invariance in object detection snip. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3578–3587 (2018)
35. Singh, B., Najibi, M., Davis, L.S.: SNIPER: Efficient multi-scale training. In: Advances in neural information processing systems. pp. 9310–9320 (2018)

36. Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., Marculescu, D.: Single-Path NAS: Designing hardware-efficient convnets in less than 4 hours. arXiv preprint arXiv:1904.02877 (2019)
37. 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)
38. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. arXiv preprint arXiv:1904.01355 (2019)
39. Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C.: NAS-FCOS: Fast neural architecture search for object detection. arXiv preprint arXiv:1906.04423 (2019)
40. Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
41. Xie, Saining, R.G.P.D.Z.T., He, K.: Aggregated residual transformations for deep neural networks (2017)
42. Xu, H., Yao, L., Zhang, W., Liang, X., Li, Z.: Auto-FPN: Automatic network architecture adaptation for object detection beyond classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6649–6658 (2019)
43. Yao, L., Xu, H., Zhang, W., Liang, X., Li, Z.: SM-NAS: Structural-to-modular neural architecture search for object detection. arXiv preprint arXiv:1911.09929 (2019)
44. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9759–9768 (2020)
45. Zhang, X., Wan, F., Liu, C., Ji, R., Ye, Q.: FreeAnchor: Learning to match anchors for visual object detection. In: Advances in Neural Information Processing Systems. pp. 147–155 (2019)
46. Zhu, C., Chen, F., Shen, Z., Savvides, M.: Soft anchor-point object detection. arXiv preprint arXiv:1911.12448 (2019)
47. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. arXiv preprint arXiv:1903.00621 (2019)
48. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)