OPANAS: One-Shot Path Aggregation Network Architecture Search for Object Detection

Tingting Liang[1] Yongtao Wang[1*] Zhi Tang[1] Guosheng Hu[2] Haibin Ling[3]

[1]Wangxuan Institute of Computer Technology, Peking University [2]Anyvision
[3]Department of Computer Science, Stony Brook University

{tingtingliang, wyt, tangzhi}@pku.edu.cn huguosheng100@gmail.com hling@cs.stonybrook.edu

Abstract

Recently, neural architecture search (NAS) has been exploited to design feature pyramid networks (FPNs) and achieved promising results for visual object detection. Encouraged by the success, we propose a novel One-Shot Path Aggregation Network Architecture Search *(OPANAS) algorithm, which significantly improves both searching efficiency and detection accuracy. Specifically, we first introduce six heterogeneous information paths to build our search space, namely* top-down, bottom-up, fusing-splitting, scale-equalizing, skip-connect *and* none. *Second, we propose a novel search space of FPNs, in which each FPN candidate is represented by a densely-connected directed acyclic graph (each node is a feature pyramid and each edge is one of the six heterogeneous information paths). Third, we propose an efficient one-shot search method to find the optimal path aggregation architecture; specifically, we first train a super-net and then find the optimal candidate with an evolutionary algorithm. Experimental results demonstrate the efficacy of the proposed OPANAS for object detection: (1) OPANAS is more efficient than state-of-the-art methods* (e.g., *NAS-FPN and Auto-FPN) at significantly smaller searching cost* (e.g., *only 4 GPU days on MS-COCO); (2) the optimal architecture found by OPANAS significantly improves main-stream detectors including RetinaNet, Faster R-CNN and Cascade R-CNN, by 2.3∼3.2 % mAP compared to their FPN counterparts; and (3) a new state-of-the-art accuracy-speed trade-off (52.2 % mAP at 7.6 FPS) is achieved at smaller training costs than comparable recent arts. Code will be released at* https://github.com/VDIGPKU/OPANAS.

1. Introduction

Recognizing objects at vastly different scales is one of the major challenges in computer vision. To address this issue, great progress has been made in designing deep convolutional networks in the past few years. Intuitively, directly extracting feature pyramid [22] from CNN at different stages provides an efficient solution. Each level of the feature pyramid corresponds to a specific scale in the original image. However, high-level features are with more semantics while the low-level ones are more content descriptive [31]. Such a semantic gap is unable to deliver strong features for multi-scale visual recognition tasks (*e.g.*, object detection, and segmentation). To alleviate the discrepancy, different feature fusion strategies have been proposed. Feature Pyramid Network (FPN) [17] is arguably the most popular basic architecture and inspires many important variants. It adopts a backbone model, typically designed for image classification, and builds a top-down information flow by sequentially combining two adjacent layers in feature hierarchy in the backbone. By such design, low-level features are complemented by semantic information from high-level features. Despite simple and effective, FPN may not be the optimal architecture design.

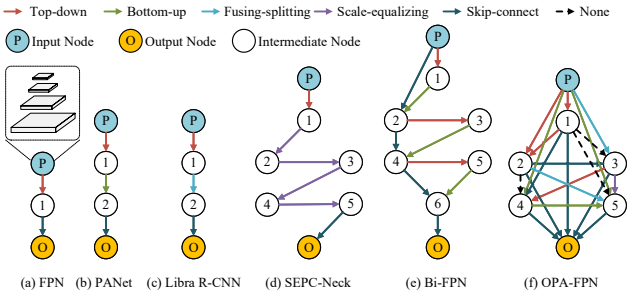

Figure 1: Different FPN architectures: (a) FPN [17], (b) PANet [21], (c) Libra R-CNN [23], (d) SEPC-Neck [28], (e) BiFPN [25], and (f) our searched optimal FPN.

Two lines of research have been conducted to advance FPN-based detection algorithms. On one hand, various approaches (*e.g.*, PANet [21], BiFPN [25], Libra R-CNN [23] and SEPC [28]) enrich FPN by aggregating multiple heterogeneous information paths and achieve impres-

*indicates corresponding author.

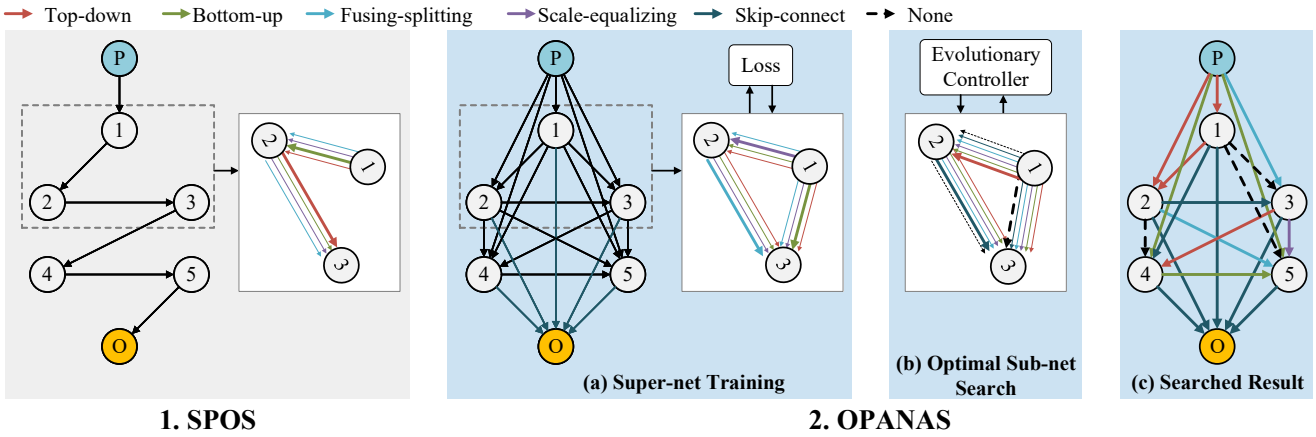

Figure 2: **1.** Single-path FPN super-net from SPOS search space [12]. **2.** Our OPANAS: (a) super-net training, *i.e.*, the optimization of super-net weights; (b) optimal sub-net search with an evolutionary algorithm; (c) the searched optimal architecture. Note that two information paths (skip-connect and none) work only for (b).

sive results. However, as shown in Fig. 1 (a-e), they only explore aggregations of up to *three* types of information paths, (*i.e.*, *top-down and bottom-up* [21], *top-down and fusing-splitting* [23], and *top-down and scale-equalizing* [28]). Moreover, most of these methods follow a straightforward *chain-style aggregation structure*, except BiFPN that adds additional skip-connect on PANet with several repetitions, but *remains in a simple topology*. On the other hand, Neural Architecture Search (NAS)-based FPN architectures [10, 26, 29] have achieved remarkable performance gain beyond manually designed architectures, but with following limitations: (1) *inefficiency*, the searching processes are often computationally expensive (*e.g.*, 300 TPU days [10]) due to the extremely large search space, and (2) *weak adaptability*, their searched architectures are specialized for certain detector with special training skills (*e.g.*, large batch size or longer training schedule).

Inspired by these studies and meanwhile to address aforementioned issues, we propose a new efficient and effective NAS framework, named OPANAS (One-Shot Path Aggregation Neural Architecture Search, see Fig. 2) to automatically search a better FPN for object detection. Firstly, we carefully design four parameterized information paths (*top-down*, *bottom-up*, *scale-equalizing* and *fusing-splitting*, see Fig. 3 (a-d)) and two parameter-free ones (*skip-connect* and *none*, see Fig. 3 (e-f)) to build our search space. Clearly, these six modules introduce different information flows, different connections between backbone and detection head, and lead to complementary and highly interpretable aggregation modules. Note that the four parameterized ones are relatively heavy and the two parameter-free ones are light-weighted, and they work together to achieve a promising accuracy-efficiency trade-off.

Secondly, to achieve the optimal aggregation of the six information paths, we propose a novel FPN search space, in which each FPN candidate is represented by a densely-connected directed acyclic graph (each node is a feature pyramid and each edge is a specific one of the six heterogeneous information paths as shown in Fig. 3). Notably, our search space contains richer aggregation topological structures of FPNs than existing methods as in Fig. 1, and hence enables richer cross-level and cross-module interactions.

Thirdly, we propose an efficient one-shot search method to search the optimal FPN architecture, that is, we first train a super-net and then search the optimal sub-net from the super-net with an evolutionary algorithm that has strong global optimum search capability. Experiments show that our method is efficient as the differentiable NAS methods, *i.e.*, DARTS [20] and Fair DARTS [5], while the searched FPN architecture can achieve better detection accuracy with less parameters and FLOPs. Moreover, following the simple vanilla training protocol, our searched FPN architecture can consistently improve the detection accuracy of the main-stream detectors including RetinaNet, Faster R-CNN and Cascade R-CNN by 2.3$\sim$3.2 mAP, with less parameters and FLOPs. These results demonstrate the efficacy of the proposed OPANAS for object detection.

Our contributions can be summarized as:

- We carefully design 6 information paths that can aggregate multi-level information, and thus enable the effective and complementary combination of low-, medium- and high-level information. To our knowledge, we are the first to investigate the aggregations of multiple ($>$3) information paths.
- We propose a novel one-shot method, OPANAS, to efficiently and effectively search the optimal aggregation of the 6 kinds of information paths.
- Working as a plug and play module, our searched architecture can easily be adapted to main-stream detectors including RetinaNet, Faster R-CNN and Cascade

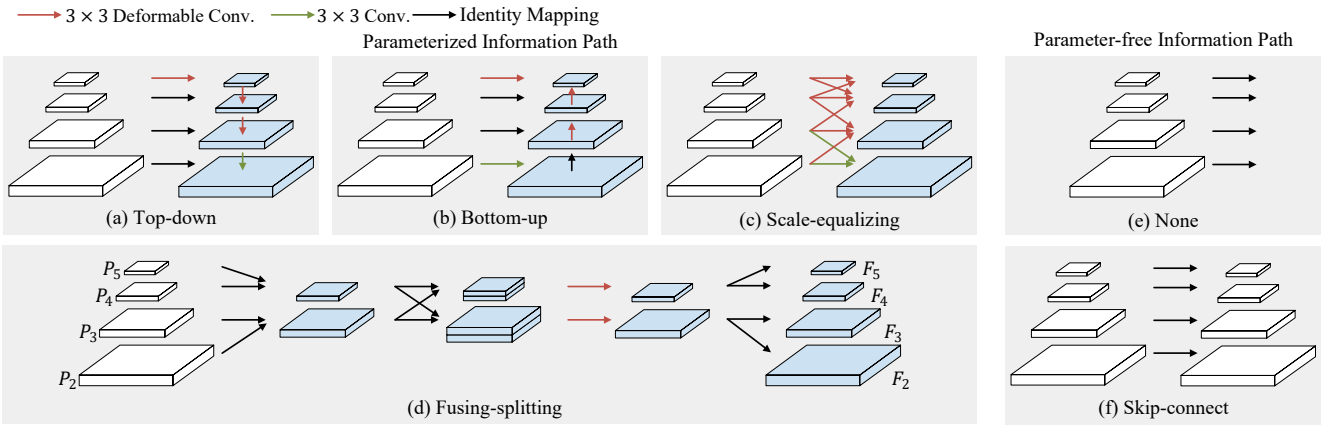

Figure 3: The proposed six heterogeneous information paths mapping 4-level pyramid features $\{P_2, P_3, P_4, P_5\}$ to $\{F_2, F_3, F_4, F_5\}$. (a)-(d) are parameterized and (e)-(f) are parameter-free.

R-CNN, and significantly improve their detection accuracy by 2.3$\sim$3.2 % mAP. Notably, we achieve a new state-of-the-art accuracy-speed trade-off (52.2 % mAP at 7.6 FPS).

2. Related Work

2.1. Object detection

Existing deep learning-based detectors can be briefly categorized into two streams: one-stage detectors such as SSD [22] and RetinaNet [18], which utilize CNN directly to predict the bounding boxes; and two-stage methods such as Faster R-CNN [24] and Mask R-CNN [13], which generate the the final detection results after extracting region proposals upon a region proposal network (RPN). Although encouraging signs of progress have been made, existing detectors are still suffering from the problems caused by the scale variation across object instances. The feature pyramid is popularly used to deal with scale variation [17], which introduces a top-down information flow.

Beyond FPN, some recent extensions employ two or three types of information paths. For example, PANet [21] introduces an extra bottom-up path after the top-down path of classic FPN [17], and Libra R-CNN [23] adopts Non-Local module [27] to fuse the features produced by the classic FPN [17] and then transfers the fused feature into multi-scale pyramid features. Multi-level FPN [32] first fuses the backbone features as the base feature and then introduces multiple U-shape modules to extract multi-level pyramid features and builds a powerful one-stage detector. SEPC [28] stacks 4 scale equalizing modules behind classic FPN to enhance cross-scale correlation. More recently, BiFPN [25] exploits a simplified architecture of PANet and stacks it repeatedly with skip-connect to build a more powerful one-stage detector named EfficientDet. Though promising results are achieved by EfficientDet, its training cost is extremely expensive, *i.e.*, large batch-size (128 on 32TPU) with a long training schedule (300 or 500 epochs). Generally speaking, these FPNs suffer from intrinsic architecture limitations since they only aggregate at most three types of information paths with naive topological structure.

2.2. Neural Architecture Search

More recently, neural architecture search (NAS) is applied to automatically search an FPN architecture for a specific detector. NAS-FPN [10], NAS-FCOS [26] and SpineNet [6] use reinforcement learning to control the architecture sampling and obtain promising results. SM-NAS [30] uses evolutionary algorithm and partial order pruning method to search the optimal combination of different parts of the detectors. The above NAS methods are effective though can be time-consuming. Auto-FPN [29], Hit-Detector [11] uses gradient-based method to search the optimal detector, which can significantly reduce searching time. However, gradient-based methods tend to trap into local minima in certain nodes of super-net during the progress of optimization and introduce further complexity [12]. Recently, researchers [1, 2, 12] propose one-shot method to decouple the super-net training and architecture search in two sequential steps. DetNAS [3] follows this idea to search an efficient backbone for object detection. One limitation of the single-path approach is that the search space is restricted to a sequential structure as shown in Fig. 2.1.

The aforementioned methods take the layer-wise operations as transform blocks (*i.e*, single-scale feature as nodes), which are completely separated from manual design. Such design forms a large search space which contains architectures beyond human design, while also includes many poor-performing architectures, leading to low search efficiency. To reduce the post-processing overhead, we propose multi-level information path aggregation as our search space. With the help of carefully designed information

paths, our search can be efficient and robust.

3. Methodology

In this work, we first propose six types of information paths, which capture diverse multi-level information. Second, to search the optimal aggregations of these information paths, we introduce an efficient One-Shot Path Aggregation Network Architecture Search (OPANAS) algorithm. Last, we detail the optimization and searching process.

3.1. Information Paths

To effectively aggregate different levels of pyramidal features, we propose 6 information paths, which can capture low-, medium- and high-level information. Similar to classic FPN [17], these information paths map the input pyramidal features $\{P_2, P_3, P_4, P_5\}$ (see Fig. 3) to $\{F_2, F_3, F_4, F_5\}$. However, the proposed information paths can capture much richer and diverse information than FPN, which will be described as following.

Top-down Information Path The top-down information path is modified from the classic FPN [17] (Fig. 3 (a)). For this path, the output pyramidal features (denoted as $F_2^t, F_3^t, F_4^t, F_5^t$) are sequentially constructed in a top-down manner, *i.e.*, the smaller scale (high-level, *e.g.*, F_5^t) feature map is constructed first. Specifically, each feature map (F_i^t) is iteratively built by combining input pyramid feature map of the same level (P_i) and the higher-level output feature (F_{i+1}^t):

$$F_i^t = \mathbf{W}_i^t \otimes (U(F_{i+1}^t) + P_i), \tag{1}$$

where $U(\cdot)$ denotes upsampling with factor of 2. For high-level features ($i = 3, 4, 5$), $\mathbf{W}_i^t$ is the 3×3 deformable convolution filter to alleviate discrepancy of a feature pyramid [28], and $\mathbf{W}_2^t$ is a normal 3×3 convolution filter.

Bottom-up Information Path For the bottom-up information path, the output pyramidal features (denoted as $F_2^b, F_3^b, F_4^b, F_5^b$) are sequentially constructed in a bottom-up manner, *i.e.*, the large scale (low-level, *e.g.*, F_2^b) feature map is constructed first as shown in Fig. 3 (b). Each feature map (F_i^b) is obtained by merging the input feature maps (P_i) of the same level, and the output feature map below it (F_{i-1}^b):

$$F_i^b = \mathbf{W}_i^b \otimes (D(F_{i-1}^b) + P_i), \tag{2}$$

where $D(\cdot)$ denotes downsampling with factor of 2 and $\mathbf{W}_i^b$ is the convolution filter with the same setting as above.

Scale-equalizing Information Path The scale-equalizing information path is motivated by SEPC [28], which stacks scale-equalizing pyramid convolutions after the classic FPN to capture inter-scale correlation. Here we take a single pyramid convolution operation as an information path. As shown in Fig. 3 (c), each feature map (F_i^s) is obtained by merging the adjacent-level input feature maps (P_i):

$$F_i^s = U(\mathbf{W}_1^s \otimes P_{i+1}) + \mathbf{W}_0^s \otimes P_i + \mathbf{W}_{-1}^s \otimes P_{i-1}, \tag{3}$$

where $\mathbf{W}_1^s, \mathbf{W}_0^s, \mathbf{W}_{-1}^s$ are 3×3 deformable convolution filters and the stride of $\mathbf{W}_{-1}^s$ is set to 2 to down-sample.

Fusing-splitting Information Path We design a two-step fusing-splitting information path, which first combines the high- and low-level input pyramidal features, and then splits the combined features to multi-scale output features in Fig. 3 (d). In practice, the highest two input feature maps are merged into α_s, and the lowest two are merged into α_l through an element-wise sum:

$$\alpha_s = P_4 + U(P_5), \quad \alpha_l = D(P_2) + P_3. \tag{4}$$

After obtaining the combined features, we fuse them through concatenation,

$$\begin{aligned} \beta_s &= \mathbf{W}_s^f \otimes \mathrm{concat}(\alpha_s, D(\alpha_l)), \\ \beta_l &= \mathbf{W}_l^f \otimes \mathrm{concat}(U(\alpha_s), \alpha_l), \end{aligned} \tag{5}$$

where $\mathbf{W}_s^f$ and $\mathbf{W}_l^f$ are 3×3 deformable convolution filters, and $\mathrm{concat}(\cdot)$ represents concatenation along the channel dimension. After these operations, feature maps β_s, β_l carry information fused from all level features. Finally, we resize them into multi-scale pyramid feature maps,

$$F_2^f = U(\beta_l), F_3^f = \beta_l; F_4^f = \beta_s, F_5^f = D(\beta_s). \tag{6}$$

Skip-connect Information Path and None Specially, we add a skip-connect path to perform identity mapping. Moreover, a 'none' information path is exploited to remove redundant information paths. These two parameter-free information paths are designed to reduce the complexity of the model, leading to a better accuracy-efficiency trade-off.

3.2. One-Shot Search

We propose a one-shot search method to efficiently and effectively search the optimal aggregation of the above six types of information paths. Specifically, we first construct a super-net $\mathcal{A}$, which is a fully-connected Multigraph DAG (directed acyclic graph). The node of DAG stands for feature maps (in the way of a feature pyramid), and there are six edges of different types between two nodes, and each edge represents one information path. The whole optimization includes two steps: (i) super-net training and (ii) optimal sub-net search, as shown in Fig. 2. For (i), we train the super-net until convergence (optimization of the weights of the super-net) using a fair sampling strategy detailed in

Method	Backbone	Time (fps)	FLOPs	Params	Search Part	Search Cost (GPU-day)	mAP
NAS-FPN(7@ 256)[10]	ResNet50	17.8†	281G	60.3M	Neck	333×#TPUs	39.9
DetNAS-FPN-Faster[3]	Searched	-	-	-	Backbone	44	40.2
Auto-FPN[29]	ResNet50	7.7	260G	32.5M	Neck & Head	16	40.5
Faster OPA-FPN@64	ResNet50	**22.0**	**123G**	**29.5M**	Neck	**4**	**41.9**
Auto-FPN[29]	X-64x4d-101	5.7	493G	90.0M	Neck & Head	16	44.3
SM-NAS[30]	Searched	9.3†	-	-	Backbone & Neck & Head	188	45.9
NAS-FCOS(@128-256)[26]	X-64x4d-101	-	362G	-	Neck & Head	28	46.1
Cascade OPA-FPN@160	ResNet50	**12.6**	**326G**	**60.6M**	Neck	**4**	**47.0**
NAS-FPN(7@ 384)&DropBlock[10]	AmoebaNet	3.6†	1126G	166.5M	Neck	333×#TPUs	48.3
SP-NAS[15]	Searched	2.1†	949G	-	Backbone & Neck & Head	> 26	49.1
EfficientDet-D7 (1536 × 1536)[25]	EfficientNet-B6	3.8†	**325G**	**51.9M**	-	-	**52.2**
SpineNet-190[6]	SpineNet-190	-	1885G	164.0M	Backbone & Neck	700×#TPUs	52.1
Cascade OPA-FPN@160 (1200 × 900)	Res2Net101-DCN	**7.6**	432G	80.3M	Neck	**4**	**52.2**

[1] † FPS marked with † are from papers, and all others are measured on the same machine with 1 V100 GPU.

Table 1: Comparison with SOTA methods on COCO `test-dev` set . **Here and after, '@c' denotes the feature channel is c, *e.g.*, '@160' implies that the feature channel is 160.**

Section 3.2.1. The weights of super-net are fixed once this training is done (one-shot optimization). For (ii), we use evolutionary algorithm (EA) to search for the optimal sub-net a^*, which is a DAG with only one optimal edge between two nodes. Obviously, the optimal sub-net represents the desired optimal FPN aggregating multiple information paths. Note that (ii) is very efficient because each sampled sub-net a just goes through the inference process by using the weights of the super-net trained in (i). This is the main reason why one-shot optimization is very efficient.

To detail the optimization process, we first introduce the components of the super-net. The super-net is a DAG consists of $N+2$ nodes (N is a predefined constant value), where the input node P represents the extracted feature from the backbone, and the output node O is the final output feature pyramid. Similarly, intermediate nodes $x_i (i = 1, 2, ..., N)$ are also feature pyramids. Each directed edge (i, j) is associated with some information path $\mathbf{IP}(i, j)$ that transforms x_i to x_j. We assume the intermediate nodes are fully connected with former nodes, and identity mapped to the output node through summation. In such DAG model, each node $i \in \{1, 2, \ldots, N\}$ aggregates inputs from previous nodes, where

$$x_j = \sum_{i<j} \mathbf{IP}(i, j)(x_i). \tag{7}$$

In this way, OPANAS allows $6^{N(N+1)/2}$ possible DAGs without considering graph isomorphism with N intermediate nodes. In particular, to maximize the search space without affecting the convergence, we set $N = 5$, and the total number of sub-nets is approximately $6^{15} \approx 10^{12}$.

Second, we formulate the super-net training as following. The architecture space $\mathcal{A}$ is encoded in a super-net, denoted as $\mathcal{N}(\mathcal{A}, W)$, where W stands for the weights of super-net. Thus, the super-net training can be formulated as:

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \mathcal{L}_{\text{train}} \left(\mathcal{N}(\mathcal{A}, W)\right). \tag{8}$$

This super-net training is detailed in Section 3.2.1.

Third, we discuss the optimal sub-net search in (ii). We aim to search the optimal sub-net $a^* \in \mathcal{A}$ that maximizes the validation accuracy, which can be formulated as:

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \text{ACC}_{\text{val}} \left(\mathcal{N}\left(a, W_{\mathcal{A}}(a)\right)\right). \tag{9}$$

We use an evolutionary algorithm to conduct this optimal sub-net search as detailed in Section 3.2.2.

3.2.1 Super-net Training

Edge Importance Weighting Unlike the existing One-Shot method SPOS [12] only having edges between adjacent nodes, our OPANAS is densely connected to explore richer topological structures for aggregation. To adapt to our multiple paths (edges) optimization, we associate an *edge importance weight* to each edge. To guarantee the consistency between training and test, we set these weights to be continuous. Consequently, each node $i \in \{1, 2, \ldots, N\}$ aggregates weighted inputs from the previous nodes, then Eq. (7) can be formulated as:

$$x_j = \sum_{i<j} \gamma_{i,j} \mathbf{IP}(i, j)(x_i), \tag{10}$$

where $\gamma_{i,j}$ denotes the edge importance weight between node i and j. Concomitantly, the optimization in Eq. (8) is modified as:

$$W_{\mathcal{A}} = \underset{W,\gamma}{\arg\min} \mathcal{L}_{\text{train}} \left(\mathcal{N}(\mathcal{A}, W, \gamma)\right). \tag{11}$$

To assist the convergence of model, we add L_1 regularization to these edge importance weights with a hyper-parameter μ to balance with the original bounding box loss. Thus the total loss function is:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{bbox}} + \mu\mathcal{L}_1 \\ &= \sum(\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}) + \mu\|\gamma\|_1.\end{aligned} \tag{12}$$

$\mathcal{L}_{\text{cls}}, \mathcal{L}_{\text{loc}}$ are objective functions corresponding to recognition and localization task respectively.

Fair Sampling Instead of training the whole super-net directly, we sample K sub-nets per training iteration to reduce the GPU memory cost. Note that 'skip-connection' and 'none' are parameter-free and do not require any optimization. Hence, they are only considered during the searching process. Consequently, in super-net training, only $K = 4$ types of information paths are involved. To alleviate training unfairness between the K parameterized information paths, we adopt strict fair sampling strategy [4] in our super-net training. To be more specific, in the n-th super-net training step, K sub-nets are sampled with no intersection. That is, each edge of them is associated with different information path, and the weights of the super-net are updated after accumulating gradients from the K sampled sub-nets. By this sampling strategy, all information paths are ensured to be equally sampled and trained within each training step, and each edge is activated only once within each training step. Consequently, the expectation and variance of edge E_i with information path i ($i = 0, 1, 2$ and 3 correspond to *top-down*, *bottom-up*, *scale-equalizing* and *fusing-splitting*, respectively) are given by,

$$\mathbf{E}(Y_{E_i}) = n \times P_{E_i} = n/K, \quad \mathbf{Var}(Y_{E_i}) = 0. \tag{13}$$

The variance does not change with n, thus fairness is assured at every training step.

3.2.2 Sub-net Search with Evolutionary Algorithm

We conduct the sub-net search with an evolutionary algorithm. Specifically, during the optimal sub-net search in Fig. 2 (b), we first randomly sample N_S sub-nets, each passes the coarse search, from the super-net and rank their performance. Note that evaluating a sub-net requires only inference without training, which makes the search very efficient. Then we repeatedly generate new sub-nets through crossover and mutation on top k performing sub-nets. Following an evolutionary algorithm [12], crossover denotes that two randomly selected sub-nets are crossed to produce a new one, mutation means a randomly selected sub-net mutates its every edge with probability 0.1 to produce a new sub-net. In this work, we set population size $N_S = 50$, max iterations $T = 12$ and $k = 10$.

Detector	Method	FLOPs	Params	mAP	FPS
RetinaNet	Baseline	239G	37.7M	35.7	**19.7**
	SEPC-Neck	314G	45.3M	**38.0**	15.1
	Balanced FPN	240G	38.0M	36.4	18.5
	PAFPN	245G	40.1M	35.9	17.8
	OPA-FPN@168	**207G**	**36.5M**	**38.0**	18.1
Faster R-CNN	Baseline	207G	41.5M	36.4	**20.6**
	SEPC-Neck	509G	49.1M	39.0	10.9
	Balanced FPN	208G	41.8M	37.2	19.3
	PAFPN	232G	45.1M	36.5	19.1
	OPA-FPN@112	**197G**	**35.5M**	**39.6**	17.3
Cascade R-CNN	Baseline	235G	69.2M	40.3	**18.1**
	SEPC-Neck	536G	76.3M	42.6	9.9
	Balanced FPN	236G	69.4M	41.2	17.0
	PAFPN	259G	72.7M	40.5	16.8
	OPA-FPN@120	**225G**	**50.6M**	**42.8**	15.0

Table 2: Comparisons of model adaptability for mainstream detectors on COCO with FPN [17] (baseline), SEPC-Neck [28] (stacking 4 scale-equalizing information paths behind FPN), Balanced FPN [23], PAFPN [21].

4. Experiments

4.1. Implementation Details

4.1.1 Datasets and Evaluation Criteria

We conduct experiments on the COCO [19] and PASCAL VOC [7] benchmarks. For COCO, the training is conducted on the 118k training images, and ablation studies are evaluated on the 5k `minival` images. We also report the results on the 20k images in `test-dev` for comparison with state-of-the-art (SOTA). For evaluation, we adopt the metrics from the COCO detection evaluation criteria, including the mean Average Precisions (mAP) across IoU thresholds ranging from 0.5 to 0.95 at different scales. For PASCAL VOC, training is performed on the union of VOC 2007 trainval and VOC 2012 trainval (10K images) and evaluation is performed on VOC 2007 test (4.9K images), mAP with an IoU threshold of 0.5 is used for evaluation.

4.1.2 Super-net Training and Sub-net Searching Phase

We consider a total of $N = 5$ intermediate nodes for super-net training and optimal sub-net searching. We choose Faster R-CNN (ResNet50 [14]) as the baseline. During super-net training, we use input-size 800×500 and sample $1/5$ images from training-set of COCO to further reduce the search cost. As for PASCAL VOC, the input-size is set to 384×384. We use SGD optimizer with initial learning rate 0.02, momentum 0.9, and 10^{-4} as weight decay. We train super-net for 12 epochs with batch-size of 16. Edge importance weight γ is initialized as 1, and hyper-parameter μ is 10^{-4}. In total, the whole search phase is completed in 4 days (1 day for super-net training and 3 days for optimal

Dataset	Method	Search on	FLOPs	Params	mAP
COCO	Basline [17]	-	207	41.5M	38.6
	Auto-FPN	VOC	-	31.3M	38.9
	Auto-FPN	COCO	260G	32.6M	40.5
	OPA-FPN@64	VOC	128G	**29.8M**	41.5
	OPA-FPN@64	COCO	124G	29.8M	**41.6**
VOC	Basline [17]	-	207G	41.2M	79.7
	Auto-FPN	VOC	-	31.2M	81.8
	Auto-FPN	COCO	256G	32.5M	81.3
	OPA-FPN@64	VOC	127G	**29.5 M**	**82.7**
	OPA-FPN@64	COCO	124G	29.6M	82.5

Table 3: Comparisons of model transferability between different datasets with Auto-FPN [29].

sub-net search) using 1 V100 GPU.

4.1.3 Full Training Phase

In this phase, we fully train the searched model.SGD is performed to train the full model with batch-size of 16. The initial learning rate is 0.02; 10^{-4} as weight decay; 0.9 as momentum. Single-scale training with input 1333×800 size is trained for 12 epochs, and the learning rate is decreased by 0.1 at epoch 8 and 11. While multi-scale training (pixel size=$400 \sim 1400$) is trained for 24 epochs with learning rate decreased by 0.1 at epoch 16 and 22. We use single-scale training for ablation studies if not specified, and we compare with SOTA with multi-scale training.

4.2. Results

4.2.1 Comparison with SOTA

The searched optimal architecture by OPANAS, denoted as **OPA-FPN**, is illustrated in Fig. 2 (c). It is evaluated together with state-of-the-art detectors including hand-crafted [25] and NAS-based ones [30, 10, 26, 6, 29, 3]. Note that these methods search different components of detectors. Specifically, SM-NAS searches the overall architecture of Cascade R-CNN, NAS-FPN searches the neck architecture, and Auto-FPN searches the architectures of both neck (FPN) and detection head. As shown in Tab. 1, compared with representative results achieved by these SOTA methods, our method achieves better or very competitive results in terms of amount of parameters, computation complexity, accuracy, and inference speed. Notably, our method can search for the best neck architecture more efficiently, *e.g.*, 4 GPU days on COCO. Specially, our searched OPA-FPN equipped with Cascade R-CNN Res2Net101-DCN [9] achieves a new state-of-the-art accuracy-speed trade-off (52.2 % mAP at 7.6 FPS), outperforming SpineNet (the SOTA NAS based method) and EfficientDet (based on the SOTA NAS searched backbone). These results demonstrate the effectiveness of our carefully designed search space and the efficiency of the search algorithm.

Method	FLOPs	Params	mAP
Baseline [17]	207G	41.5M	36.4
Top-down	$209^{+0.7\%}$G	$41.8^{+0.5\%}$M	$37.5^{+1.1}$
Bottom-up	$209^{+0.7\%}$G	$41.8^{+0.5\%}$M	$30.0^{-6.4}$
Scale-equalizing	$234^{+13.0\%}$G	$41.1^{-0.9\%}$M	$32.6^{-3.8}$
Fusing-splitting	$\mathbf{181}^{-\mathbf{12.4\%}}$G	$40.1^{-3.4\%}$M	$37.6^{+1.2}$
OPA-FPN@112	$197^{-3.4\%}$G	$\mathbf{35.5}^{-\mathbf{14.5\%}}$M	$\mathbf{39.6}^{+\mathbf{3.2}}$

Table 4: Comparisons with the single-information-path FPN architectures on COCO `minival`.

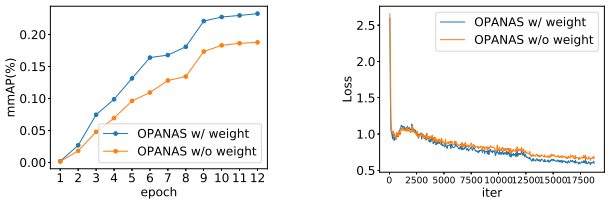

Figure 4: Comparisons of the intermediate results of super-net training w/ or w/o edge importance weighting. mmAP denotes the mean mAP of random sampled 50 sub-nets from the current super-net training epoch.

4.2.2 Model Adaptability for Main-stream Detectors

To further verify the performance of the OPA-FPN on main-stream detectors, we adapt it to RetinaNet [18], Faster R-CNN [24] and Cascade R-CNN [8] in Tab. 2. Under the same training strategy with baseline, we obtain a lighter model with better performance on each detector: improving RetinaNet by 2.3% mAP with 13% FLOPs decreasing, and improving Cascade R-CNN by 2.5% mAP with 27% parameter amount decreasing. Moreover, comparing with other hand-craft FPNs, our architecture achieves clearly better results in terms of amount of parameters, computation complexity, accuracy.

4.2.3 Model Transferability Between COCO and VOC

To evaluate the transferability of our architecture on different datasets, we transfer the searched architecture between COCO and VOC with multi-scale training, as shown in Tab. 3. With the architecture searched on COCO, our method boosts the performance by 3.0% mAP for COCO, 2.8% mAP for VOC, respectively. When searched on VOC, our method boosts the performance by 3.0% mAP for VOC, 2.9% mAP for COCO, respectively. No matter which dataset we search on, our architecture performs better than Auto-FPN [29] (*e.g.*, 41.6% vs.40.5% in terms of mAP) with fewer computation cost (*e.g.*, FLOPs 128G vs. 260G). These results further demonstrate the effectiveness of our method.

Densely Connected	Fair Sampling	Edge Importance Weight	τ	mAP
			0.4390	38.4
✓			0.3584	39.2
✓	✓		0.5865	39.5
✓	✓	✓	**0.6145**	**39.6**

Table 5: Correlation analysis of proposed method.

4.3. Ablation Study

4.3.1 Information Path Aggregation

To demonstrate the effectiveness of aggregating different information paths in our searched OPA-FPN, we first compare it with the architectures using single information path in Tab. 4. We choose the original Faster R-CNN (ResNet50) + vanilla FPN [17] as the baseline, and we adjust the channel dimensions of our searched OPA-FPN and detection head, aiming to align the complexity with the baseline. OPA-FPN significantly surpasses the baseline and the single-information-path architectures, with fewer FLOPs/Params (*e.g,* FLOPs 197G vs. 207G, Params 35.5M vs. 41.5M), achieving an effective aggregation and exploration of information paths.

4.3.2 Edge Importance Weighting

To verify the effectiveness of the edge importance weight γ in Eq. (10), we illustrate the intermediate results of training super-net with fair sampling in Fig. 4, and observe that the edge importance weighting brings clear benefits for the training of super-net. These results prove that distinguishing the importance of different edges is effective for densely connected super-net training.

4.3.3 Correlation Analysis

Recently, the effectiveness of weight sharing-based NAS methods is questioned because of the lack of (1) fair comparison on the same search space and (2) adequate analysis on the correlation between the super-net performance and the stand-alone sub-net model performance [12]. Here we adopt Kendall Tau [16] to measure the correlation of model ranking obtained from super-net. Specifically, we randomly sample 15 sub-nets from the trained super-net and conduct full train to evaluate their performance. In Tab. 5, when changing from single-path super-net used by SPOS [12] to our densely connected super-net, there is a drop in correlation but the detection accuracy increases. However, by adopting fair sampling and edge importance weight, we achieve a much higher correlation value, showing that our method can achieve a higher correlation between super-net and sub-net with the fair sampling and the proposed edge importance weighting.

Search Method	Search time (GPU days)	FLOPs	Params	Best mAP of searched arch
Random	55	231/206G	36.4/35.9M	37.5/39.1
SPOS	4	207G	36.3M	38.4
DARTS	5	198G	37.8M	39.1
FairDarts	4	269G	36.2M	39.4
OPANAS	4	**197G**	**35.5M**	**39.6**

Table 6: Comparisons with more NAS baselines on COCO `minival`. The baseline detector is Faster R-CNN ResNet50, the search space of SPOS is illustrated in Fig. 2.1, and the others all adopt the proposed search space illustrated in Fig. 2.2

4.3.4 Comparisons with More NAS Baselines

We further compare our method with more existing NAS methods, including a) Random Search: we randomly sample 15 architectures from the proposed search space and conduct full training under the same training setting in our experiments; b) SPOS: we train the single-path one-shot FPN super-net and perform EA search strategy following [12]; c) DARTS: a very popular differentiable NAS method [20]; and d) Fair DARTS: an improved version of DARTS with softmax relaxation and zero-one loss [15]. As the results reported in Tab. 6, compared with other NAS methods, our method can find a better architecture with less or comparable time. These results prove the effectiveness of our super-net training and optimal sub-net search strategy.

5. Conclusion

In this paper, we propose One-Shot Path Aggregation Network Architecture Search (OPANAS), which consists of a novel search space and an efficient searching algorithm, to automatically find an effective FPN architecture for visual object detection. In particular, we introduce six types (*i.e.*, top-down, bottom-up and scale-equalizing, fusing-splitting, balanced, skip-connect, and none) of information paths as candidate operations, and exploit densely connected DAG to represent FPN to aggregate them. An efficient one-shot search method is further invented to search the optimal FPN, based on the super-net training with fair sampling and edge importance weighting. Extensive experimental results demonstrate the superiority of the proposed OPANAS in both efficiency and effectiveness.

Acknowledgements

This work was supported by National Key R& D Program of China No. 2019YFB1406302. This work was also a research achievement of Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 3

[2] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 3

[3] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NeuralIPS*, pages 6638–6648, 2019. 3, 5, 7

[4] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *CoRR*, 2019. 6

[5] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020. 2

[6] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, 2020. 3, 5, 7

[7] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 6

[8] Pedro F. Felzenszwalb, Ross B. Girshick, and David A. McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248, 2010. 7

[9] S. Gao, M. Cheng, K. Zhao, X. Zhang, M. Yang, and P. H. S. Torr. Res2net: A new multi-scale backbone architecture. *TPAMI*, pages 1–1, 2019. 7

[10] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. NAS-FPN: learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019. 2, 3, 5, 7

[11] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. Hit-detector: Hierarchical trinity architecture search for object detection. In *CVPR*, 2020. 3

[12] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 2, 3, 5, 6, 8

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[15] Chenhan Jiang, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. SP-NAS: serial-to-parallel backbone search for object detection. In *CVPR*, 2020. 5, 8

[16] M.G. Kendall. A new measure of rank correlation. In *Biometrika*, volume 30, pages 81–93, 1938. 8

[17] Tsung-Yi Lin, Piotr Dollár, and Ross B. Girshick et al. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017. 1, 3, 4, 6, 7, 8

[18] Tsung-Yi Lin, Priya Goyal, and Ross B. Girshick et al. Focal loss for dense object detection. In *ICCV*, pages 2999–3007, 2017. 3, 7

[19] Tsung-Yi Lin, Michael Maire, and Serge J. Belongie et al. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755, 2014. 6

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 2, 8

[21] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 1, 2, 3, 6

[22] Wei Liu, Dragomir Anguelov, and Dumitru Erhan et al. SSD: single shot multibox detector. In *ECCV*, pages 21–37, 2016. 1, 3

[23] Jiangmiao Pang, Kai Chen, and Jianping Shi et al. Libra R-CNN: towards balanced learning for object detection. In *CVPR*, pages 821–830, 2019. 1, 2, 3, 6

[24] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 3, 7

[25] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 1, 3, 5, 7

[26] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. NAS-FCOS: fast neural architecture search for object detection. In *CVPR*, 2020. 2, 3, 5, 7

[27] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018. 3

[28] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wayne Zhang. Scale-equalizing pyramid convolution for object detection. In *CVPR*, pages 13356–13365. IEEE, 2020. 1, 2, 3, 4, 6

[29] Hang Xu, Lewei Yao, and Wei Zhang et al. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, pages 6649–6658, 2019. 2, 3, 5, 7

[30] Lewei Yao, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. SM-NAS: structural-to-modular neural architecture search for object detection. In *AAAI*, 2020. 3, 5, 7

[31] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1

[32] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling:. M2det: A single-shot object detector based on multi-level feature pyramid network. In *AAAI*, pages 9259–9266, 2019. 3