


Article

FastDARTSDet: Fast Differentiable Architecture Joint Search on Backbone and FPN for Object Detection

Chunxian Wang ^{1,2}, Xiaoxing Wang ¹, Yiwen Wang ¹, Shengchao Hu ¹ , Hongyang Chen ³, Xuehai Gu ⁴, Junchi Yan ^{1,*} and Tao He ^{5,*}

¹ School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

² Auto-Driving Research Center, IM Motors Technology Co., Ltd., Shanghai 201804, China

³ Auto-Driving Research Center, SAIC Motor Passenger Vehicle Company, Shanghai 201804, China

⁴ Nanjing Laisi Network Technology Research Institute Co., Ltd., Nanjing 210008, China

⁵ COWAROBOT Co., Ltd., Wuhu 241060, China

* Correspondence: yanjunchi@sjtu.edu.cn (J.Y.); tommie.he@cowarobot.com (T.H.)

Abstract: Neural architecture search (NAS) is a popular branch of automatic machine learning (AutoML), which aims to search for efficient network structures. Many prior works have explored a wide range of search algorithms for classification tasks, and have achieved better performance than manually designed network architectures. However, few works have explored NAS for object detection tasks due to the difficulty to train convolution neural networks from scratch. In this paper, we propose a framework, named as FastDARTSDet, to directly search on a larger-scale object detection dataset (MS-COCO). Specifically, we propose to apply differentiable architecture search method (DARTS) to jointly search backbone and feature pyramid network (FPN) architectures for object detection task. Extensive experimental results on MS-COCO show the efficient and efficacy of our method. Specifically, our method achieves 40.0% mean average precision (mAP) on the test set, outperforming many recent NAS methods.

Keywords: neural architecture search; object detection; automatic machine learning



Citation: Wang, C.; Wang, X.; Wang, Y.; Hu, S.; Chen, H.; Gu, X.; Yan, J.; He, T. FastDARTSDet: Fast Differentiable Architecture Joint Search on Backbone and FPN for Object Detection. *Appl. Sci.* **2022**, *12*, 10530. <https://doi.org/10.3390/app122010530>

Academic Editor: Sungho Kim

Received: 4 August 2022

Accepted: 12 October 2022

Published: 19 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Machine learning has achieved great success in various tasks, including computer vision [1–3], nature language processing [4,5], and digital image and signal processing [6–8]. Neural architecture search aims to automatically find the best neural network architecture in some search space instead of manually design based on a large amount of trails and expert knowledge. Several approaches for NAS have been explored, including reinforcement learning, evolutionary algorithm, and differentiable NAS (DARTS). The works [9–12] adopt reinforcement learning by considering the generation of an architecture as the agent's action. Another reinforcement learning based method [13], named ENAS, proposes weight sharing strategy among the same operations in different architectures, which can significantly reduce the time cost during the search process. Another work [14] adopts maximum flow in graph theory to address NAS task. Other works [15–20] adopt genetic algorithms by first encoding the neural architecture and then proposing a group of architectures as a population. Population of architectures are selected according to their performance and then new individuals are generated by crossover and mutation strategies. Apart from the above methods, one-shot neural architecture search [21] has been one of the most popular searching paradigms of neural architecture search (NAS) for its high efficiency. Unlike reinforcement-learning-based methods [9–12] and evolutionary algorithms [15–20] that generate one candidate network at a time, one-shot-based methods construct a supernet containing all the connections in the search space and jointly train all operation weights. DARTS [22] further introduces architecture parameters and address NAS as a bi-level

optimization problem which is solved by stochastic gradient descent (SGD). After the gradient-based search phase, DARTS discretizes the supernet and infers the final architecture simply according to the value of architecture parameters, which is referred to as the discretization procedure. Some works [23,24] notice the huge memory cost of one-shot NAS and are dedicated to more efficient searching methods. Moreover, the works [25,26] attempt to improve the optimization algorithm for more stable searching. Differentiable architecture search [22] has taken the dominance with a myriad of follow-up works [26–30] to reduce the search cost.

However, for object detection task, due to the complexity of network and difficult to train models, the above methods that focus on classification task can not directly search on object detection dataset. Unlike classification models, object detection models contain two modules: backbone and feature pyramid network (FPN). To reduce the search cost, recent works only search for the architectures of backbone [31,32] or FPN [33–35] separately. This problem has aroused widespread concern in the industry, because in practice, manually adjusting each module based on the standard detection model is inefficient and suboptimal. It is difficult to use and evaluate the trade-off between reasoning time and accuracy and the presentation ability of each module in different datasets. In particular, authors in SM-NAS [36] find that “the combination of cascade RCNN and resnet18 (not the standard detection model) is faster and more accurate than the combination of FPN and resnet50 in coco [37] and BDD [38] (automatic driving data set)”. Though many prior works have been dedicated to exploring neural architecture search for object detection tasks, they suffer vast GPU memory cost and searching time, e.g., 44 GPU-days for DetNAS [31]. The above NAS methods for object detection adopt RetinaNet and Faster-RCNN framework. However, few NAS methods adopt YOLO framework [39–42], which are more efficient detectors. EAUTOdet [43] proposes an efficient architecture search method for YOLO framework and is able to discover effective architectures in a few GPU-days. Inspired by the above methods, we refer to YOLOv5 framework and propose to apply DARTS [22] to jointly search for the architectures of backbone and FPN. Our method aims to discover optimal architectures in a few GPU-days.

The contributions of our approach can be summarized as: (1) applying DARTS to object detection task and supporting search for the architectures of backbone and FPN; and (2) strong performance on COCO dataset outperforming many recent manually-designed networks.

2. Related Work

This section introduces the prior works related to our method. We first introduce some classic object detection methods in Section 2.1. Then, we introduce the efficient neural architecture search methods, most of which aim at classification task. Finally, we introduce the recent NAS methods for object detection task.

2.1. Object Detection

Existing framework of object detector usually consists of several parts: a CNN backbone to extract features, a feature fusion module to fuse extracted features at different scales, a region proposal network (RPN) to generate candidate target region (two-stage detectors [2]), and a detection head to predict and classify bounding boxes. Since each module plays an important role in object detectors, recent advances focus on the designing of each module. For example, except for directly using the existing backbone for classification, such as the VGG [44] and ResNet [1], some researchers put forward a new backbone specially for the detectors (DetNet [45]). FPN [46] is one of the typical networks exploring the design of the feature fusion neck, which designs a top-down architecture to fuse features at different scales. Though great progress has been made through these designs, many of these detectors just focus on one module and ignore the relationship between the backbone and head which may cause the sub-optimal result. R-CNN [47] considers region proposals and achieves high accuracy, it can not detect the object in real-time speed even with Fast

R-CNN [48] and Faster R-CNN [2] due to the region generation process. Apart from the above classic object detection methods, there are also some manually-designed detectors for rotation detection either, including regression-based methods [49,50] and classification-based models [51–53], especially for high-precision detection of small objects [54–56].

2.2. One-Shot Architecture Search

One-shot NAS [21] regards neural network architectures as Directed Acyclic Graphs (DAG), and constructs a supernet containing all types of operations and connections in the search space. Each candidate neural network architecture can be seen as a sub-graph of the supernet. Based on one-shot NAS, [22] involve architecture parameters to represent the importance of candidate operations and connections, which are then optimized alternately with the network weights based on SGD. XNAS [57] address NAS as an online selection task, and adopt the prediction with experts advice (PEA) theory to select operations and connections from the search space. Other methods [30,58] propose to reduce the GPU memory requirement by gradually removing connections in the supernet. Ref. [30] proposes to prune the connections with low confidence and increase the depth of the supernet (the number of cells). Ref. [23] proposes PC-DARTS to reduce the GPU memory requirement by sampling $1/K$ channels for each operation in the one-shot model, where K is a hyperparameter controlling the rate of activated channels of each convolution at each iteration. PC-DARTS also introduces to accumulate the architecture importance of different iteration to stabilize the optimization. Ref. [58] utilizes the Bayesian learning and compression to compute the entropy of the connections, according to which the architecture could be pruned to reduce the GPU memory cost and accelerate the searching phase.

2.3. NAS for Object Detection

DetNAS [31] refers to DARTS and propose to search neural architectures for object detection by SGD. However, it requires vast GPU memory and search time to discover an architecture. Recently, researchers [27–29] proposed the adoption of a single-path searching strategy to reduce the memory cost of searching for image classification. However, such a single-path searching strategy increases the difficulty of training supernets. Since the object detection neural architectures are more complex and hard to train, the single-path strategy is not quite suitable for searching object detection architectures. Moreover, object detection networks have feature pyramid networks (FPN) to fuse features at different scales, which differs from those for classification. Since FPN plays an important role in object detection networks, many works are dedicated to searching architectures for optimal FPN. NAS-FPN [33] aims to search FPN architecture for RetinaNet [59], a popular one-stage detection framework. Specifically, FPN architectures are generated by an RNN controller, which is trained by reinforcement learning (RL). However, NAS-FPN requires vast GPU memory and search time. EAUTOdet [43] proposed an efficient architecture search method for YOLO framework and achieves great performance on COCO dataset. Inspired by the above related works, we propose to search on YOLO framework and propose to directly apply DARTS [22] to joint search for backbone and FPN architectures.

3. Method

3.1. Search Space

This work designs search spaces for backbone network and feature pyramid network separately. Specifically, for backbone, we consider efficiency and effectiveness and refer to DARTS [22] to construct a backbone by stacking normal and reduced cells. Each cell contains two input nodes and four intermediate nodes. Each intermediate nodes are connected with all their predecessors. For FPN, we resort to PANet [60] and design search spaces for both top-down and bottom-up fusion modules.

3.1.1. Search Space for Backbone

The macro architecture of backbone is shown in Figure 1, and we propose to search operation types and connections for each cell. The backbone consists of $3N$ normal cells and three reduced cells. Unlike DARTS where normal/reduced cells share the same architecture, we independently search each of the cells, i.e., all cells can have different architectures. The importance of candidate operations and connections are represented by the architecture parameters α , which is trained by SGD algorithm during the search stage. After searching, we derive the final architecture according to the magnitude of α . Specifically, for each intermediate node, we preserve two connections sourced from different predecessors, and select the best operation on the two connections.

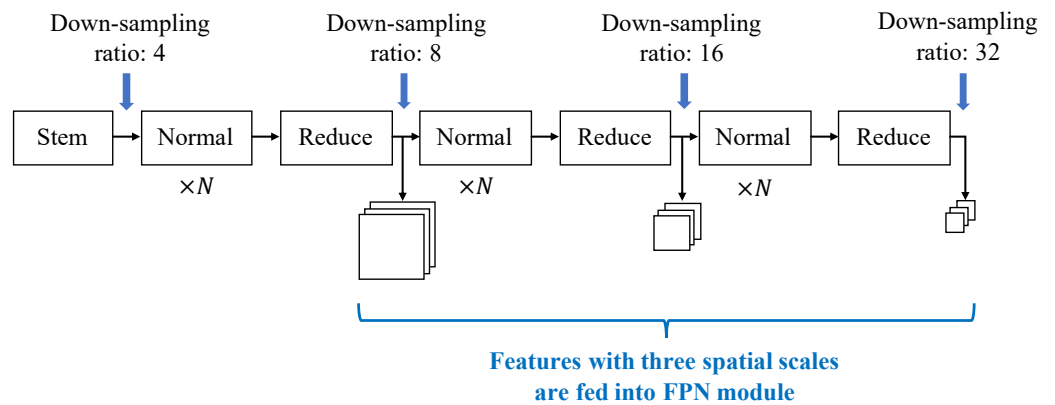


Figure 1. Macro architecture of the backbone supernet. Each cell is independently searched. N is the number of repeated normal cells, which affects the model size.

3.1.2. Search Space for FPN

We extract three features of different spatial sizes from the backbone and pass them through FPN. As shown in Figure 2, we adopt nodes to denote feature maps and edges to denote operations. Each normal cell is a supernet that is independently searched during the search process. Similar to the search space of backbone, there are 7 candidate operations. Similar to the backbone, we introduce architecture parameters $\hat{\gamma}$ to denote the importance of candidate operations for each normal cell, whose normalized weights are denoted as $\gamma = \text{softmax}(\hat{\gamma})$. To introduce the feature of nodes in a normal cell, we take the j -th node as an example without loss of generality. The feature of node v_j is $z_j = \sum_{i < j} \sum_{o \in \mathcal{O}} \gamma_{ij}^o \cdot e_{ij}^o(z_i)$, where \mathcal{O} is the candidate operation set, z_i is the feature of node v_i (a predecessor of node v_j), and e_{ij}^o denotes the operation o on edge e_{ij} that connects node v_i and v_j . After searching, only two connections are preserved for each node, and only one operation for each connection is selected according to the magnitude of γ .

3.2. The Proposed FastDARTSDet

Based on DARTS [22], we build a supernet containing all candidate operations and connections in the search space. Nodes of supernet represents feature maps and edges denotes operations. We utilize e_{ij} to denote the edge from node v_i to v_j , and utilize $\{e_{ij}^o, o \in \mathcal{O}\}$ to denote candidate operations on edge e_{ij} , where \mathcal{O} is an operation candidate set for each edge. Similar to prior works [21,22], we design $\mathcal{O} = \{\text{zero, identity, max pooling, average pooling, } 3 \times 3 \text{ convolution, } 5 \times 5 \text{ convolution, } 3 \times 3 \text{ dilated convolution, and } 5 \times 5 \text{ dilated convolution}\}$. To search architectures via gradient-based method, we follow DARTS [22] and define the architecture parameters $\hat{\alpha}_{ij}^o$ to represent the importance of

different candidate operations on each edge. The output of operations e_{ij}^o are averaged with weight α_{ij}^o to obtain the output of edge e_{ij} , represented as $o_j(x_i)$:

$$o_j(x_i) = \sum_{o \in \mathcal{O}} \alpha_{ij}^o e_{ij}^o(x_i), \quad \alpha_{ij}^o = \frac{\exp(\hat{\alpha}_{ij}^o)}{\sum_{o' \in \mathcal{O}} \exp(\hat{\alpha}_{ij}^{o'})} \in [0, 1] \quad (1)$$

where α denotes the normalized architecture parameters, x_i denotes the output of v_i , and, thus, the output of v_j can be computed as follows:

$$x_j = \sum_{i < j} o_j(x_i) \quad (2)$$

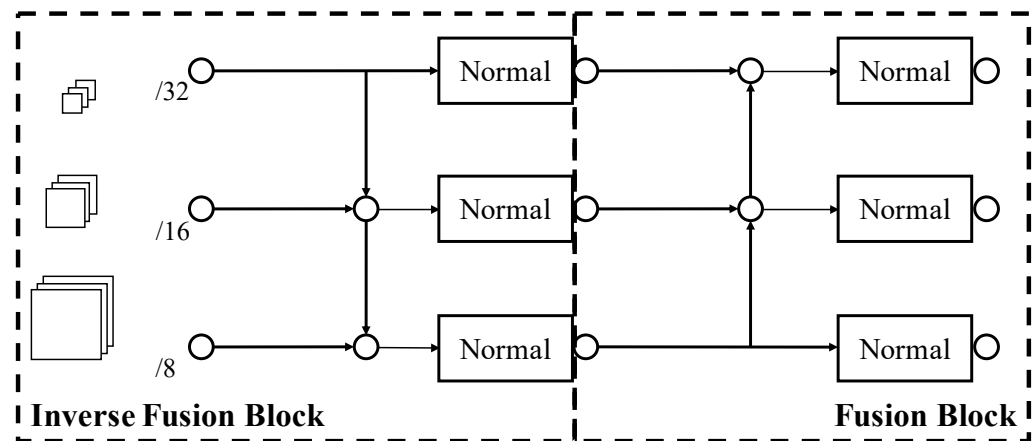


Figure 2. Macro architecture of the supernet for feature pyramid network (FPN). Features in three spatial scales are fed into the FPN module. Each normal cell is a supernet independently searched during the search process.

Similar to DARTS [22], we address NAS as a bi-level optimization problem and solve it by SGD algorithm. The optimization problem is formalized as follows:

$$\begin{aligned} \min_{\alpha, \gamma} \quad & \mathcal{L}_{val}(\mathbf{W}^*(\alpha, \gamma)) \\ \text{s.t.} \quad & \mathbf{W}^*(\alpha, \gamma) = \arg \min_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, \alpha, \gamma) \end{aligned} \quad (3)$$

where \mathbf{W} denotes the supernet weights, α denotes normalized architecture parameters for backbone following Equation (1), γ denotes normalized architecture parameters for FPN module.

Unlike DARTS that search on a rather simple classification task, we propose to search on a much more complex computer vision task, object detection. The major differences lie in two aspects: (1) the detection datasets are larger than classification datasets, and (2) the detection models are more complex than the classification datasets. Specifically, detection models usually contain multiple parts, including backbone, feature pyramid network, and detector head. In this work, by defining the joint search space of backbone and FPN in the Section 3.1, we adopt the differentiable-based neural architecture search algorithm to search CNN architectures for object detection task.

4. Experiments

4.1. Protocols

The models are evaluated on MS-COCO 2017 dataset [37].

4.1.1. MS-COCO Dataset

Ms-COCO dataset is a large-scale image dataset developed and maintained by Microsoft. The tasks of frequency aggregation include recognition, segmentation, and detection. In the classic case, the target location is determined through the bounding box. At the beginning, it is mainly used for face detection and pedestrian detection. The dataset, such as Caltech pedestrian dataset, contains 350,000 bounding box tags. Pascal VOC data includes 20 targets, more than 11,000 images and more than 27,000 target bounding boxes. Recently, there are detection datasets obtained under ImageNet data, 200 categories, 400,000 images, and 350,000 bounding boxes. Because some targets have a strong relationship rather than existing independently, it is meaningful to detect a certain target in a specific scene. Therefore, accurate location information is more important than bounding box.

4.1.2. Experimental Settings

All our models are trained from scratch, that is, no ImageNet pretrained weights are adopted to initialize the our model. In the search process, A supernet with one normal cell ($N = 1$) at each block is build, as shown in Figure 1. The architecture parameters are defined to represent the importance of candidate operations and connections. The training set of COCO is divided into two parts for training architecture parameters and network weights, respectively. The final architecture is derived after alternately optimizing architecture parameters and network weights for 30 epochs by SGD optimizer. In the evaluation process, the discovered architectures are trained from scratch for 300 epochs by SGD optimizer. The models are also deepened by increasing the number of normal cells N to 2. The hyper-parameters are set as those provided by YOLOv5 for a fair comparison. Our codes are based on PyTorch and all our experiments are conducted on V100 GPU.

4.2. Performance of the Supernet

Here, The performance curve of supernet during the search process is illustrated in Figure 3, including the curve of loss functions, precision, recall, and mean average precision (mAP). Blue lines in Figure 3 is the averaged value among all 80 classes in COCO dataset. Three left columns report the tendency of bounding box loss, objectiveness loss, and classification loss. Two right figures on the top report the precision and recall of training set. Two right figures on the bottom report the mAP of validation set. Figure 3 shows that the supernet gradually converges during the search process, demonstrating the great convergence ability of our search method.

The precision-recall, and F1-score curve of each classes are illustrated in Figure

4.3. Performance of the Searched Model

The performance curve of the searched model during the evaluation process is illustrated in Figure 5. The model is trained from scratch for 300 epochs with the same hyper-parameters as YOLOv5. The precision-recall and F1-score curve of each classes are illustrated in Figure 6. Figure 7 illustrates the confusion matrix among 80 classes of the searched model. 4.

To visualize the model performance, The ground truth bounding boxes and the predicted bounding boxes of several images is displayed in Figure 8a,b. Overall, our model performs good and is able to detect most of the objects in the images. However, for small objects and occluded objects, the performance of our model is barely satisfactory. In addition, our model may also be confused about similar categories. For example, in the fourth image at the bottom, a boy takes a piece of paper, though our model detect the location of the paper but it mis-classify it as a laptop since laptop is pretty similar to a paper.

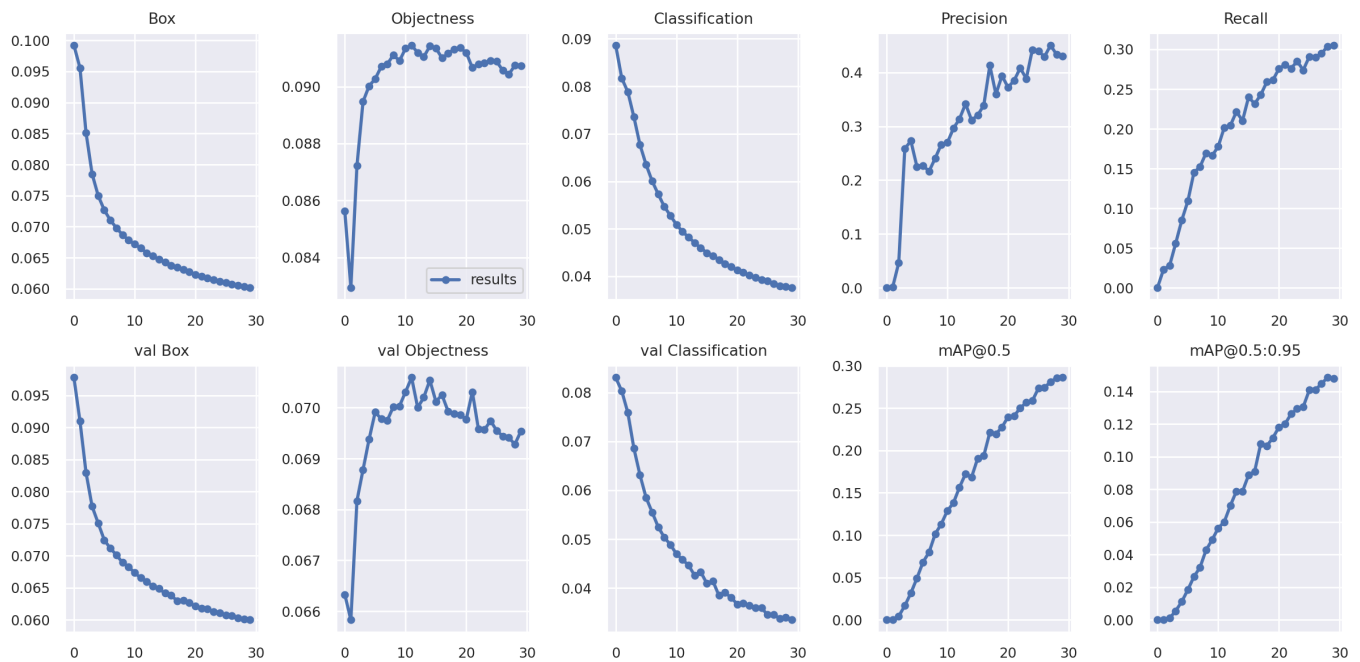


Figure 3. Illustration of the performance curves during the search process. The supernet is trained for 30 epochs.

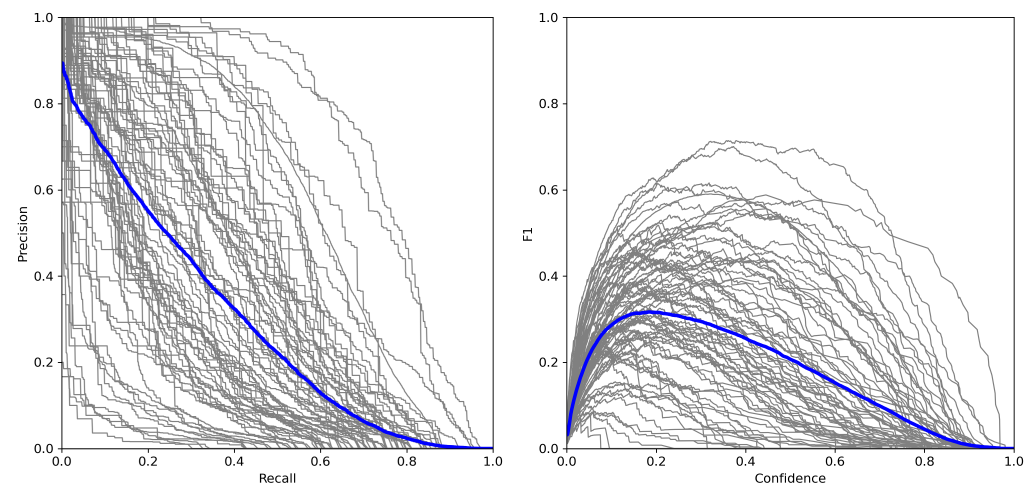


Figure 4. Curves of precision-recall and F1-score during the search process. The supernet is trained for 30 epochs. Blue line denotes the averaged value among all 80 classes.

4.4. Ablation Study

To demonstrate the advantage of collaborative search, The results that only search Backbone and FPN are reported in Table 1 and illustrated in Figure 9. Specifically, the default architecture of backbone is set as the architecture of DARTS [22] and the default architecture of FPN is set as the architecture of PANet [60]. The depth of network is controlled by the number of stacked normal cells N . The baseline (default architectures of both backbone and FPN) only achieves 33.9% mAP when $N = 1$ and 38.1% mAP when $N = 2$. If the backbone is searched independently, the performance of the discovered model improves by 0.2% (for $N = 1$) and 0.6% (for $N = 2$); if the FPN is searched independently, the performance of the discovered model improves by 1% (for $N = 1$) and 0.9% (for $N = 2$); If the backbone and FPN are searched jointly, the performance of the discovered model improves by 2.5% (for $N = 1$) and 1.8% (for $N = 2$). In general, we can obtain the following conclusions: first of all, if compared with baseline only, it can be found that whether it is a separate

search or a collaborative search, whether it is a Backbone search only or a FPN search only, the final structure is significantly improved compared with baseline. The results also reflect the effectiveness of our algorithm. Secondly, by comparing the collaborative search and separate search, we can find that even if everyone is better than the baseline, the effect of collaborative search is more obvious. From the empirical analysis, the results are consistent with the conjecture. Previous studies tend to search only one part and fix the other part, which will lead to ignoring the connections in different structures. End-to-end networks should be regarded as a whole rather than a segmented part. Finally, we notice that the increase in the result gradually decreases with the increase in the model size: for $N = 1$, the performance of joint search surpasses that of independent search by nearly 2% mAP, while for $N = 2$, the improvement is less than 1%. This is because the complexity of the model search increases with the increase in the model size. If the same experimental parameters are maintained, the effect will inevitably decrease.

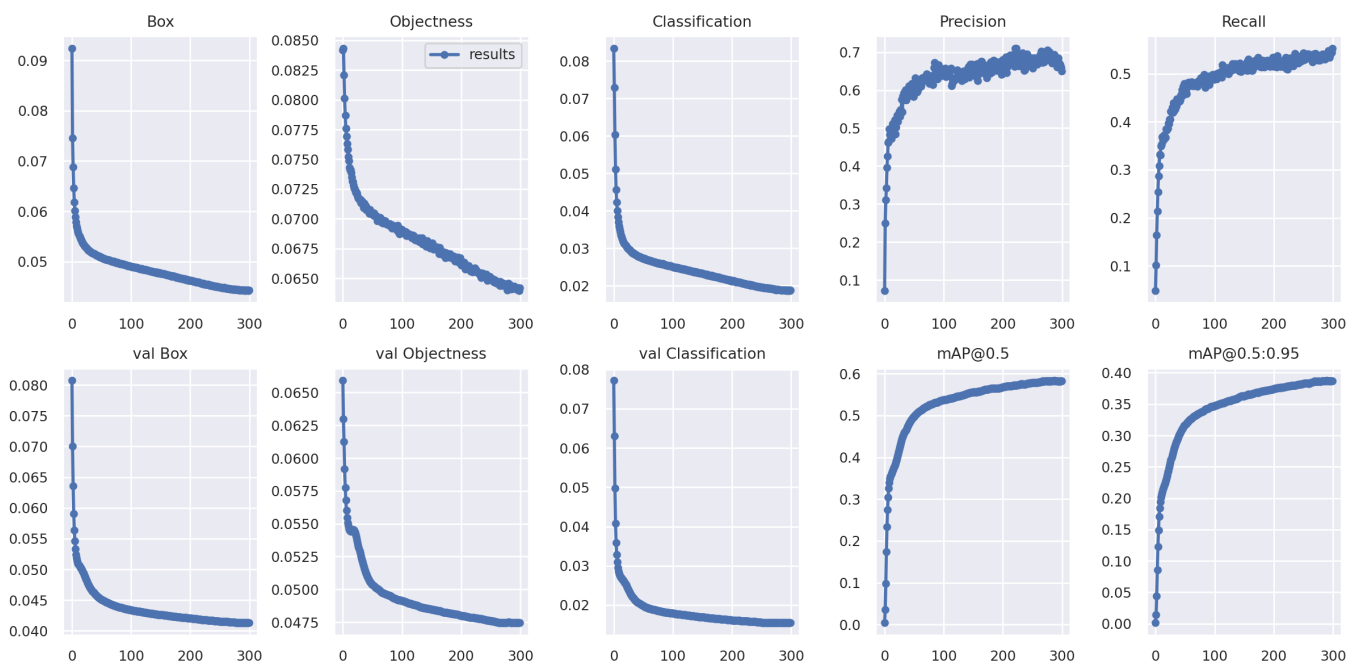


Figure 5. Illustration of the performance curves during the evaluation process. The searched model was trained for 300 epochs.

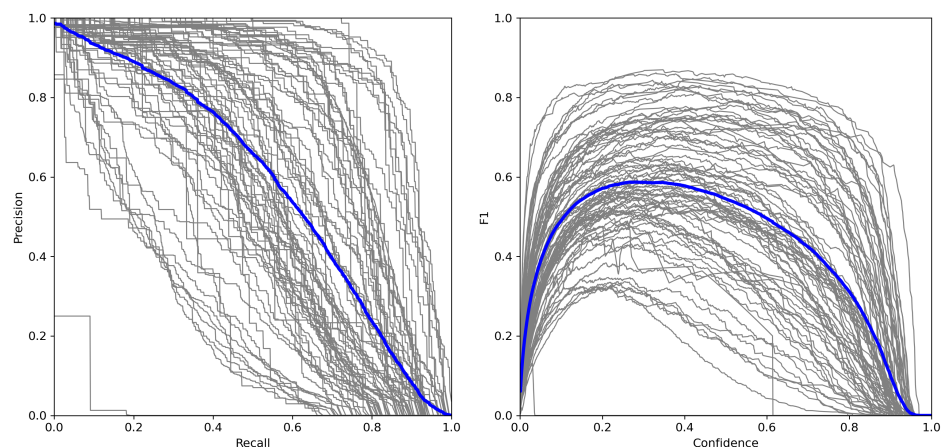


Figure 6. Curves of precision-recall and F1-score during the evaluation process. The searched model was trained for 300 epochs. Blue line denotes the averaged value among all 80 classes.

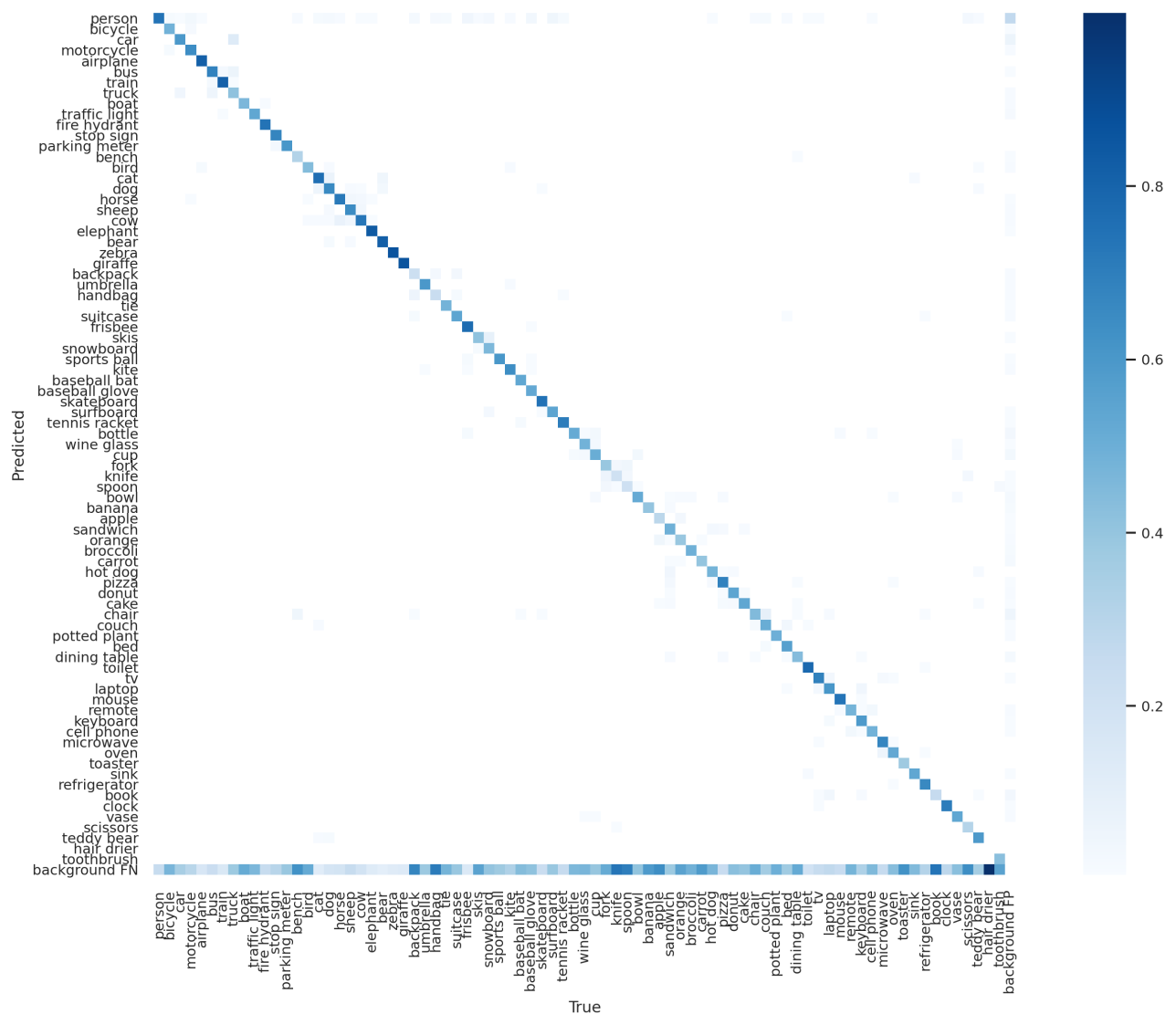


Figure 7. Confusion matrix among 80 classes of the searched model.

4.5. Comparison with Prior Methods

To show the effectiveness of our method, we compare the performance of searched architectures with other state-of-the-art works on the COCO test-dev dataset in Table 2. Our discovered models (FastDARTSDet) achieve the competitive and even better performance compared with the peer NAS methods. Specifically, FastDARTSDet with $N = 1$ achieves 36.4 AP with only 5.8 M parameters, outperforming EfficientDet-D0 by 2.6% AP. FastDARTSDet with $N = 2$ achieves 40.0% AP with 6.9 M parameters, surpassing EfficientDet-D1 by 0.4% AP with similar parameters. Moreover, compared to prior NAS methods for object detection, our method only requires 4.2 GPU-days, significantly faster than prior NAS methods. Additionally, most of the prior methods independently search either backbone (DetNAS, EfficientDet) or FPN model (NAS-FPN, NAS-FCOS, Auto-FPN), and only a few methods propose to jointly search both backbone and FPN (SM-NAS, Hit-Detector). On the one hand, compared to independent search method, our search space is much larger; on the other hand, compared to other joint search method, our method is much more efficient and faster.

Table 1. Performance of joint and independent search for backbone and FPN on MS-COCO validation set. The default architecture of backbone is the architecture by DARTS [22]. The default architecture of FPN is the architecture of PANet [60]. Specifically, the ‘default’ backbone utilize the discovered architecture searched on classification task, which cost 1.0 GPU-day. Consequently, for the setting of ‘default’ backbone and ‘default’ FPN, the search cost is 1.0 GPU-day. For the setting of ‘default’ backbone and ‘searched’ FPN, the search cost is 5.2 GPU-day (1.0 GPU-day to search default backbone and 4.2 GPU-days to search FPN). The best setting and the corresponding performance among each block is in bold.

Number of Normal Cell	Architecture		mAP	Number of Parameters	Search Cost GPU-Days
	Backbone	FPN			
N = 1	default	default	33.9	5.81 M	1.0
	searched	default	34.1	5.93 M	4.2
	default	searched	34.9	5.87 M	5.2
	searched	searched	36.4	5.76 M	4.2
N = 2	default	default	38.1	6.82 M	1.0
	searched	default	38.7	7.29 M	4.2
	default	searched	39.0	7.03 M	5.2
	searched	searched	39.9	6.94 M	4.2

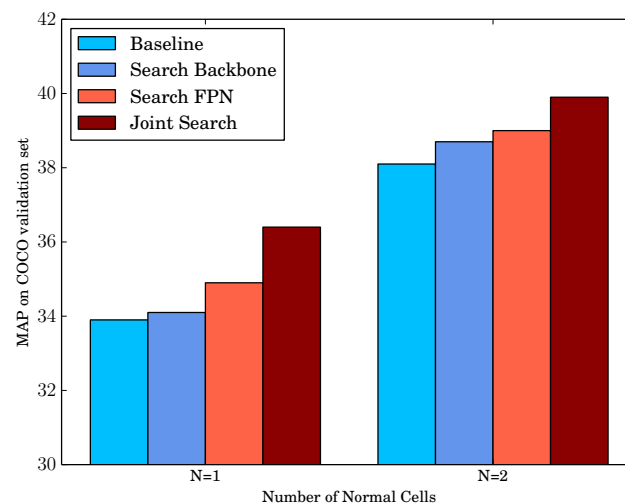


Figure 9. Performance of joint and independent search for backbone and FPN on MS-COCO validation set.

Table 2. Comparison with prior works on the COCO test-dev. Different blocks indicate models with various inference speeds and prediction performance. ‘*’: the unit of search cost is TPU-days, while the unit of other methods is GPU-days.

Method	Backbone	Resolution	FPS	#Params (M)	mAP (%)	AP ₅₀ (%)	AP ₇₅ (%)	AP _S (%)	AP _M (%)	AP _L (%)	Search Cost
EfficientDet-D0 [61]	Efficient-B0	BiFPN	512	3.9	33.8	52.2	35.8	12.0	38.3	51.2	-
NAS-FPN [33]	Res50	Searched	640	60.3	39.9	-	-	-	-	-	333 *
NAS-FCOS@128 [35]	Res50	Searched	1333 × 800	27.8	37.9	-	-	-	-	-	28
SpineNet-49S [62]	Searched	FPN	640	11.9	39.5	59.3	43.1	20.9	42.2	54.3	-
SM-NAS:E2 [36]	Search the combination		800 × 600	-	40.0	58.2	43.4	21.1	42.4	51.7	187
EfficientDet-D1 [61]	Efficient-B1	BiFPN	640	6.6	39.6	58.6	42.3	17.9	44.3	56.0	-
DetNAS [31]	Searched	FPN	1333 × 800	-	42.0	63.9	45.8	24.9	45.1	56.8	44
NAS-FPN [33]	Res50	Searched	1024	60.3	44.2	-	-	-	-	-	333 *
Auto-FPN [34]	Res50	Searched	800	32.6	40.5	61.5	43.8	25.6	44.9	51.0	16
NAS-FCOS@256 [35]	R-101	Searched	1333 × 800	57.3	43.0	-	-	-	-	-	28
SpineNet-49 [62]	Searched	FPN	640	28.5	42.8	62.3	46.1	23.7	45.2	57.3	-
SM-NAS:E3 [36]	Search the combination		800 × 600	-	42.8	61.2	46.5	23.5	45.5	55.6	187
Hit-Detector [63]	Searched	Searched	1200 × 800	27.1	41.4	62.4	45.9	25.2	45.0	54.1	-
OPA-FPN@64 [64]	Res50	Searched	1333 × 800	29.5	41.9	-	-	-	-	-	4
FastDARTSDet (N = 1)	Searched	Searched	640	5.8	36.4	53.4	38.0	18.6	39.4	45.3	4.2
FastDARTSDet (N = 2)	Searched	Searched	640	6.9	40.0	59.4	41.7	20.5	41.1	54.3	4.2

5. Conclusions and Outlook

We have presented an efficient and effective search approach to discover optimal architectures for object detection, which is a relatively less explored area. Unlike previous works searching either backbone or FPN structure alone, our method can simultaneously search backbone and FPN architecture. We adopt a differentiable-based algorithm in such a complex search space and propose a kernel reusing technique to speed up the search process stably. Our method can discover outstanding architectures in 4.2 GPU-days, whose efficacy has been demonstrated by extensive experiments. In particular, the discovered architecture achieves 40.0 AP on COCO test-dev with 6.9M parameters, and our light-weighted model achieves 36.4 AP on COCO test-dev with 5.8M parameters, competitive and even better than the state-of-the-art object detection NAS methods.

For future work, we would formulate the architecture search problem as a special case of combinatorial optimization on graphs [65], which can be readily connected to the recent advance in machine learning for combinatorial optimization [66]. In particular, it would be interesting to see if some useful architectures can be quickly searched from the architecture pool as more and more architectures have been found by experts and search algorithms. In this sense, architecture matching, or essentially graph matching especially across multiple graphs [67] to estimate the similarity, has been an important direction to explore as thus one need not search a brand new architecture from scratch. We have noted a recent line of research on using machine learning to achieve efficient and effective graph matching by using graph neural networks [68,69]. Readers are referred to [70] for more comprehensive review and we believe retrieval with fine search rather than search from scratch would be an important direction for future NAS research.

Author Contributions: Conceptualization, C.W. and J.Y.; methodology, C.W., X.G. and T.H.; software, C.W. and X.W.; validation, C.W., X.W., Y.W. and S.H.; formal analysis, C.W.; investigation, Y.W., S.H. and X.G.; resources, J.Y., T.H.; data curation, H.C.; writing—original draft preparation, C.W. and X.W.; visualization, Y.W. and S.H.; supervision, T.H. and J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by NSFC U19B2035, Interdisciplinary Program of Shanghai Jiao Tong University (YG2021QN67), the Shanghai Automotive Industry Science and Technology Development Foundation, CAAI-Huawei MindSpore Open Fund, and Chengdu Municipal Science and Technology Project 2019-YF08-00282-GX.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: MS-COCO dataset can be downloaded from <https://cocodataset.org/>.

Acknowledgments: Authors would also like to appreciate the Student Innovation Center of SJTU for providing GPUs.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. 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, Las Vegas, NV, USA, 27–30 June 2016.
2. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
3. Cao, J.; Mao, D.; Cai, Q.; Li, H.; Du, J. A review of object representation based on local features. *J. Zhejiang Univ. Sci. C* **2013**, *14*, 495–504. [\[CrossRef\]](#)
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
5. Kou, F.; Du, J.; Lin, Z.; Liang, M.; Li, H.; Shi, L.; Yang, C. A semantic modeling method for social network short text based on spatial and temporal characteristics. *J. Comput. Sci.* **2018**, *28*, 281–293. [\[CrossRef\]](#)

6. Li, W.; Sun, J.; Jia, Y.; Du, J.; Fu, X. Variance-constrained state estimation for nonlinear complex networks with uncertain coupling strength. *Digit. Signal Process.* **2017**, *67*, 107–115. [\[CrossRef\]](#)
7. Li, W.; Jia, Y.; Du, J.; Yu, F. Gaussian mixture PHD filter for multi-sensor multi-target tracking with registration errors. *Signal Process.* **2013**, *93*, 86–99. [\[CrossRef\]](#)
8. Li, Q.; Du, J.; Song, F.; Chao, W.; Liu, H.; Cheng, L. Region-based multi-focus image fusion using the local spatial frequency. In Proceedings of the 25th Chinese Control and Decision Conference, CCDC, Guiyang, China, 25–27 May 2013.
9. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures using Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
10. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
11. Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; Liu, C. Practical Block-Wise Neural Network Architecture Generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2423–2432. [\[CrossRef\]](#)
12. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Salt Lake City, UT, USA, 18–23 June 2018.
13. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via Parameter Sharing. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
14. Xue, C.; Wang, X.; Yan, J.; Li, C.G. A Max-Flow based Approach for Neural Architecture Search. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23 October 2022.
15. 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 International Conference on Machine Learning, ICML, Sydney, Australia, 6–8 August 2017.
16. Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; Kavukcuoglu, K. Hierarchical Representations for Efficient Architecture Search. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
17. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized Evolution for Image Classifier Architecture Search. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI, Honolulu, HI, USA, 27 January–1 February 2019; pp. 4780–4789.
18. Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzian, A.; Duffy, N.; et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 293–312.
19. Xie, L.; Yuille, A.L. Genetic CNN. In Proceedings of the IEEE International Conference on Computer Vision, ICCV, Venice, Italy, 22–29 October 2017; pp. 1388–1397. [\[CrossRef\]](#)
20. Elsken, T.; Metzen, J.H.; Hutter, F. Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
21. Bender, G.; Kindermans, P.; Zoph, B.; Vasudevan, V.; Le, Q.V. Understanding and Simplifying One-Shot Architecture Search. In Proceedings of the International Conference on Machine Learning, ICML, Stockholm, Sweden, 10–15 July 2018.
22. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search. In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
23. Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.J.; Tian, Q.; Xiong, H. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
24. Wang, X.; Xue, C.; Yan, J.; Yang, X.; Hu, Y.; Sun, K. MergeNAS: Merge Operations into One for Differentiable Architecture Search. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, IJCAI, Yokohama, Japan, 7–15 January 2021.
25. Bi, K.; Hu, C.; Xie, L.; Chen, X.; Wei, L.; Tian, Q. Stabilizing DARTS with Amended Gradient Estimation on Architectural Parameters. *arXiv* **2019**, arXiv:1910.11831.
26. Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; Hutter, F. Understanding and Robustifying Differentiable Architecture Search. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
27. Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA, 15–20 June 2019.
28. Xie, S.; Zheng, H.; Liu, C.; Lin, L. SNAS: Stochastic neural architecture search. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
29. Dong, X.; Yang, Y. Searching for a Robust Neural Architecture in Four GPU Hours. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA, 15–20 June 2019; pp. 1761–1770.
30. Chen, X.; Xie, L.; Wu, J.; Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1294–1303.
31. Chen, Y.; Yang, T.; Zhang, X.; Meng, G.; Xiao, X.; Sun, J. DetNAS: Backbone Search for Object Detection. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 6638–6648.

32. Jiang, C.; Xu, H.; Zhang, W.; Liang, X.; Li, Z. SP-NAS: Serial-to-Parallel Backbone Search for Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; pp. 11860–11869. [\[CrossRef\]](#)
33. Ghiasi, G.; Lin, T.; 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, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 7036–7045. [\[CrossRef\]](#)
34. Xu, H.; Yao, L.; Li, Z.; Liang, X.; Zhang, W. Auto-FPN: Automatic Network Architecture Adaptation for Object Detection Beyond Classification. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea, 27 October–2 November 2019; pp. 6648–6657. [\[CrossRef\]](#)
35. Wang, N.; Gao, Y.; Chen, H.; Wang, P.; Tian, Z.; Shen, C.; Zhang, Y. NAS-FCOS: Fast Neural Architecture Search for Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; pp. 11940–11948. [\[CrossRef\]](#)
36. Yao, L.; Xu, H.; Zhang, W.; Liang, X.; Li, Z. SM-NAS: Structural-to-modular neural architecture search for object detection. In Proceedings of the The Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.
37. Lin, T.; Maire, M.; Belongie, S.J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. *Microsoft COCO: Common Objects in Context*. ECCV; Lecture Notes in Computer Science; Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Zurich, Switzerland, 2014; Volume 8693, pp. 740–755. [\[CrossRef\]](#)
38. Yu, F.; Xian, W.; Chen, Y.; Liu, F.; Liao, M.; Madhavan, V.; Darrell, T. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *arXiv* **2018**, arXiv:abs/1805.04687.
39. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, 26 June–1 July 2016.
40. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:abs/1804.02767.
41. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
42. Jocher, G. YOLOv5 Documentation. Available online: <https://docs.ultralytics.com/> (accessed on 3 August 2020).
43. Wang, X.; Lin, J.; Yan, J.; Zhao, J.; Yang, X. EAutoDet: Efficient Architecture Search for Object Detection. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23 October 2022.
44. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
45. Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; Sun, J. Detnet: Design backbone for object detection. In Proceedings of the European Conference on Computer Vision, ECCV, Munich, Germany, 8–14 September 2018.
46. 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, CVPR, Honolulu, HI, USA, 21–26 July 2017.
47. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Washington, DC, USA, 23–28 June 2014.
48. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, ICCV, Santiago, Chile, 7–13 December 2015.
49. Yang, X.; Yan, J.; Feng, Z.; He, T. R3Det: Refined Single-Stage Detector with Feature Refinement for Rotating Object. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, Virtually, 2–9 February 2021.
50. Yang, X.; Yan, J.; Ming, Q.; Wang, W.; Zhang, X.; Tian, Q. Rethinking rotated object detection with gaussian wasserstein distance loss. In Proceedings of the International Conference on Machine Learning, ICML, Virtual, 18–24 July 2021.
51. Yang, X.; Yan, J. Arbitrary-Oriented Object Detection with Circular Smooth Label. In Proceedings of the European Conference on Computer Vision, ECCV, Glasgow, UK, 23–28 August 2020.
52. Yang, X.; Hou, L.; Zhou, Y.; Wang, W.; Yan, J. Dense Label Encoding for Boundary Discontinuity Free Rotation Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Virtual, 20–25 June 2021.
53. Yang, X.; Yan, J. On the Arbitrary-Oriented Object Detection: Classification based Approaches Revisited. *Int. J. Comput. Vis.* **2022**, *130*, 1340–1365. [\[CrossRef\]](#)
54. Yang, X.; Yang, J.; Yan, J.; Zhang, Y.; Zhang, T.; Guo, Z.; Sun, X.; Fu, K. SCRDet: Towards More Robust Detection for Small, Cluttered and Rotated Objects. In Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, Seoul, Korea, 27 October–2 November 2019.
55. Yang, X.; Yang, X.; Yang, J.; Ming, Q.; Wang, W.; Tian, Q.; Yan, J. Learning High-Precision Bounding Box for Rotated Object Detection via Kullback-Leibler Divergence. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18381–18394.
56. Yang, X.; Yan, J.; Liao, W.; Yang, X.; Tang, J.; He, T. SCRDet++: Detecting Small, Cluttered and Rotated Objects via Instance-Level Feature Denoising and Rotation Loss Smoothing. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *2022*, 1. [\[CrossRef\]](#) [\[PubMed\]](#)
57. Nayman, N.; Noy, A.; Ridnik, T.; Friedman, I.; Jin, R.; Zelnik-Manor, L. XNAS: Neural Architecture Search with Expert Advice. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1975–1985.
58. Zhou, H.; Yang, M.; Wang, J.; Pan, W. BayesNAS: A Bayesian Approach for Neural Architecture Search. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7603–7613.
59. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *TPAMI* **2020**, *42*, 318–327. [\[CrossRef\]](#) [\[PubMed\]](#)

60. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Salt Lake City, UT, USA, 18–22 June 2018. [\[CrossRef\]](#)
61. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 10778–10787.
62. Du, X.; Lin, T.-Y.; Jin, P.; Ghiasi, G.; Tan, M.; Cui, Y.; Le, Q.V.; Song, X. SpineNet: Learning scale-permuted backbone for recognition and localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 11589–11598.
63. 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 Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, 13–19 June 2020; pp. 11402–11411.
64. Liang, T.; Wang, Y.; Tang, Z.; Hu, G.; Ling, H. OPANAS: One-Shot Path Aggregation Network Architecture Search for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Virtual, 19–25 June 2021; pp. 10195–10203.
65. Wang, R.; Hua, H.; Liu, G.; Zhang, J.; Yan, J.; Qi, F.; Yang, S.; Yang, X. A Bi-Level Framework for Learning to Solve Combinatorial Optimization on Graphs. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 21453–21466.
66. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d’horizon. *Eur. J. Oper. Res.* **2021**, *290*, 405–421. [\[CrossRef\]](#)
67. Yan, J.; Cho, M.; Zha, H.; Yang, X.; Chu, M.S. Multi-Graph Matching via Affinity Optimization with Graduated Consistency Regularization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 1228–1242. [\[CrossRef\]](#)
68. Wang, R.; Yan, J.; Yang, X. Combinatorial Learning of Robust Deep Graph Matching: An Embedding based Approach. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *2020*, 1. [\[CrossRef\]](#)
69. Wang, R.; Yan, J.; Yang, X. Neural Graph Matching Network: Learning Lawler’s Quadratic Assignment Problem with Extension to Hypergraph and Multiple-graph Matching. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 5261–5279. [\[CrossRef\]](#)
70. Yan, J.; Yang, S.; Hancock, E. Learning Graph Matching and Related Combinatorial Optimization Problems. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI, Yokohama, Japan, 11–17 July 2020.