DetNAS: Neural Architecture Search on Object Detection

Yukang Chen[1†], Tong Yang[2†], Xiangyu Zhang[2], Gaofeng Meng[1], Chunhong Pan[1], Jian Sun[2]
[1]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences*
[2]*Megvii Inc*
{*yukang.chen, gfmeng, chpan*}*@nlpr.ia.ac.cn* {*yangtong, zhangxiangyu, sunjian*}*@megvii.com*

Overview

- The performance of object detectors highly rely on features extracted by backbones. Most works on object detection directly use networks designed for classification as backbone feature extractors, e.g., ResNet.
- Image classification focus on ”what” the main object of an image is, while object detection aims at finding ”where” and ”what” each object instance in an image.
- The architectures optimized on image classification can not guarantee the performance on object detection.

Neural architecture search

- NAS has achieved great progress on image classification and semantic segmentation.
- Some NAS work directly applies the architecture searched on CIFAR-10 classification on object detection, which is sub-optimal.

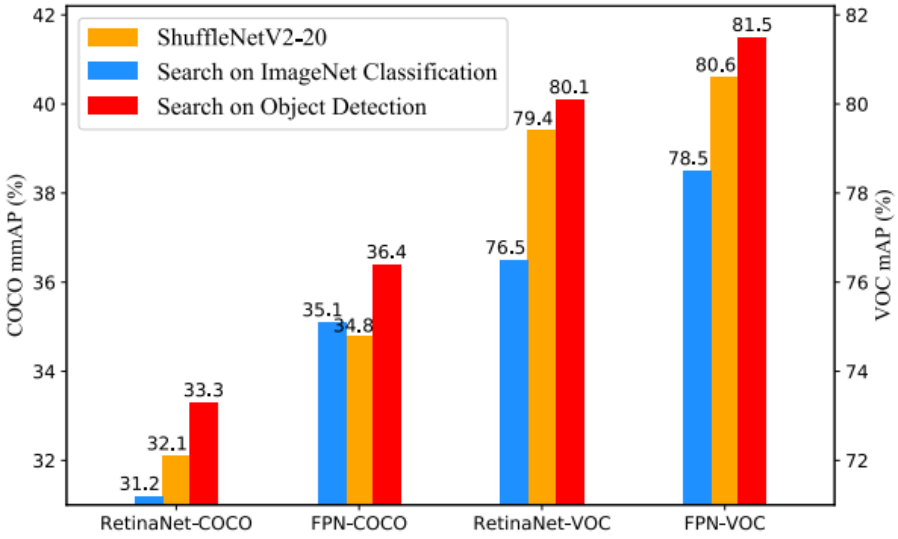

Figure 1. Backbone comparisons in various detectors and datasets. Architectures searched on the target task achieves consistent improvements over the hand-craft baseline and the ImageNet classification searched counterparts. More details are given in Table 2.

DetNAS

- DetNAS is to learn a backbone network for object detection task, and conducts neural architecture search directly on target tasks.
- To improve computation efficiency, we formulate this problem into searching the optimal path in a large graph or supernet.
- Briefly speaking, DetNAS consists of three steps:
  - training a supernet that includes all sub-networks in search space;
  - searching for the sub-network with the highest performance on the validation set with EA;
  - retraining the result network and evaluating it on the test set.

Overall pipeline

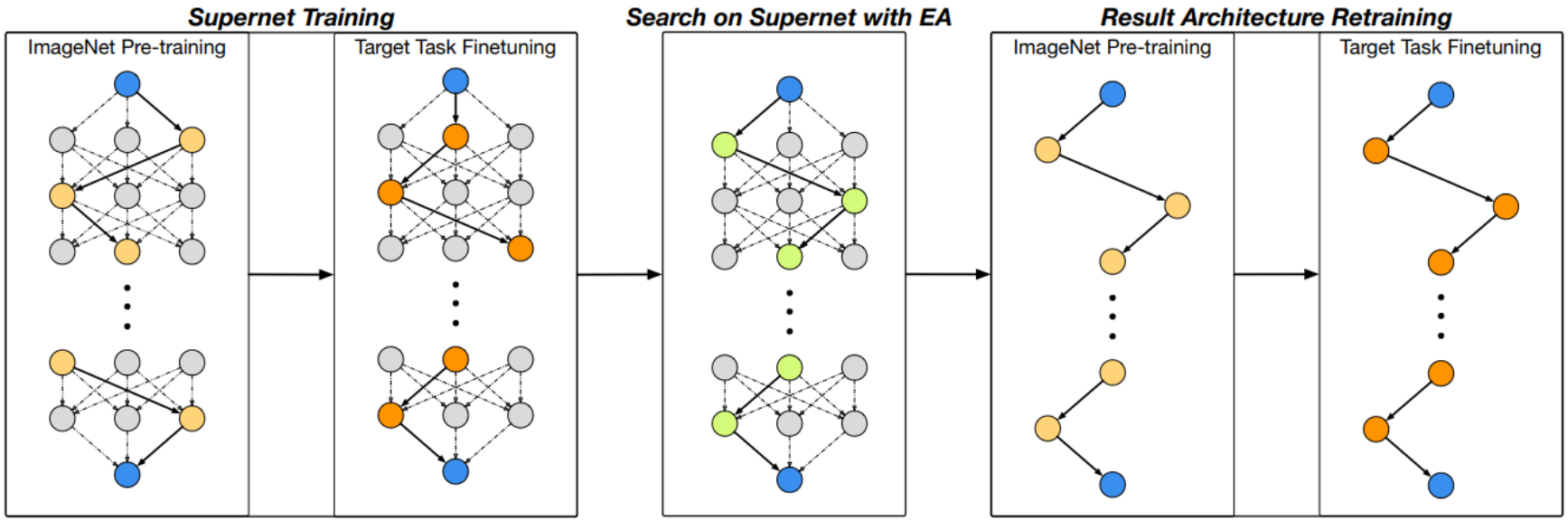

Figure 2. The pipeline of DetNAS. It involves three steps: supernet training, search on supernet with EA and result architecture retraining. The training step includes both ImageNet pre-training and target task finetuning. If we search in the from scratch scheme, no ImageNet pre-training needs during supernet training and the iterations on target task finetuning would be doubled. More details are listed in Table 1.

Search Space Supernet

- Training object detectors is known to be computationally expensive.
- The search space is formulated into a large supernet graph to relief the issue of computation cost.

Algorithm 1: SuperNet Training

Input : Search space $\mathbb{S}$, Detector $\mathcal{D}$,
ImageNet pretrain iterations $\mathrm{I_P}$,
Detection finetuning iterations $\mathrm{I_F}$,
ImageNet pretrain training set $\mathbb{T}_{Pre}$,
Detection finetuning training set $\mathbb{T}_{Det}$.

```
S ← initialize(𝕊)
i ← 0
while i < I_P do
    θ ← random-path(𝕊)
    S ← training(S(θ), 𝕋_Pre)
    i ← i + 1
end
while i < I_P + I_F do
    θ ← random-path(𝕊)
    S ← training(𝒟(S(θ)), 𝕋_Det)
    i ← i + 1
end
```

Output: The trained supernet model S

Search Space Supernet

- Key points:
  - The training process is path-wise In each iteration, only one single path is sampled for forward and back propagation. No gradient or weight update acts on other paths or nodes in the supernet graph.
  - Synchronized Batch Normalization (SyncBN) is necessary for finetuning on detection datasets instead of common Batch Normalization (BN).

Algorithm 1: SuperNet Training

Input : Search space $\mathbb{S}$, Detector $\mathcal{D}$,
ImageNet pretrain iterations $\mathrm{I_P}$,
Detection finetuning iterations $\mathrm{I_F}$,
ImageNet pretrain training set $\mathbb{T}_{Pre}$,
Detection finetuning training set $\mathbb{T}_{Det}$.

$S \leftarrow$ `initialize(`$\mathbb{S}$`)`
$i \leftarrow 0$
while $i < \mathrm{I_P}$ **do**
  $\theta \leftarrow$ `random-path(`$\mathbb{S}$`)`
  $S \leftarrow$ `training(`$S(\theta)$`,` $\mathbb{T}_{Pre}$`)`
  $i \leftarrow i + 1$
end
while $i < \mathrm{I_P} + \mathrm{I_F}$ **do**
  $\theta \leftarrow$ `random-path(`$\mathbb{S}$`)`
  $S \leftarrow$ `training(`$\mathcal{D}(S(\theta))$`,` $\mathbb{T}_{Det}$`)`
  $i \leftarrow i + 1$
end

Output: The trained supernet model S

Search Space Supernet

Table 2. ShuffleNetv2 search space backbone

Stage	Operation	c_1	n_1	c_2	n_2	s
0	Conv3×3-BN-ReLU	16	1	48	1	2
1	Shuffle block (search)	64	4	96	8	2
2	Shuffle block (search)	160	4	240	8	2
3	Shuffle block (search)	320	8	480	16	2
4	Shuffle block (search)	640	4	960	8	2

* Each stage has n blocks. All blocks in one stage has output channels c. The first block of each stage has a stride s and all others use stride 1.

- The ShuffleNetv2 block is a efficient and lightweight convolution architectures. It involves channel split and shuffle operation.
- Search space 1: 4 stages, 20 ShuffleNetv2 blocks, 4 choices/block, 4^{20}~10^{12}possible architectures

Search Strategy

1-3: A population of networks P is initialized randomly. Each individual P in the population P consists of its architecture P.θ and its fitness P.f.

5-11: After initialization, they start to evaluate the individual architecture *P.ϑ* to obtain its fitness *P.f* on the validation set $V_{det.}$ **(6)**Before that, they collect the batch normalization statistics for each network on a small subset of training set T_{Dets}.

12-16: Among these evaluated networks, we select the top $|P|$ as parents to generate child networks.
(14-15) This step involves two common transformations, mutation and crossover.

Algorithm 2: Search on Supernet with EA

Input : Trained supernet model S, Detector $\mathcal{D}$, Population size $|\mathbf{P}|$, Parent size $|\mathcal{P}|$, Detection validation set $\mathbb{V}_{Det}$, Small subset of detection training set $\mathbb{T}_{Det_S}$, Evolution iterations I_{E}, FLOPs constraint η.

1 $\mathbf{P}^{(0)} \leftarrow$ `random-initialize`$(|\mathbf{P}|, \eta)$
2 $j \leftarrow 0$
3 $f_{best}, \theta_{best} \leftarrow (0, \varnothing)$
4 **while** $j < \mathrm{I}_{\mathrm{E}}$ **do**
5 **for** $P \in \mathbf{P}^{(j)}$ **do**
6 $S(P.\theta) \leftarrow$ `collectBN`$(\mathcal{D}(S(P.\theta)), \mathbb{T}_{Det_S})$
7 $P.f \leftarrow$ `evaluate`$(\mathcal{D}(S(P.\theta)), \mathbb{V}_{Det})$
8 **if** $P.f > f_{best}$ **then**
9 $f_{best}, \theta_{best} \leftarrow (P.f, P.\theta)$
10 **end**
11 **end**
12 $\mathcal{P} \leftarrow$ `select-topk`$(\mathbf{P}^{(j)}, |\mathcal{P}|)$
13 $j \leftarrow j + 1$
14 $\mathbf{P}_{mut} \leftarrow$ `mutate`$(\mathcal{P}, \frac{|\mathbf{P}|}{2}, \eta)$
15 $\mathbf{P}_{crs} \leftarrow$ `crossover`$(\mathcal{P}, \frac{|\mathbf{P}|}{2}, \eta)$
16 $\mathbf{P}^{(j)} \leftarrow \{\mathbf{P}_{mut}, \mathbf{P}_{crs}\}$
17 **end**

Output: The best architecture θ_{best}

Search Strategy

- Key points:
  - The search process is also pathwise. To evaluate each path, validation data is only passed through the specific path.
  - Before the path is evaluated on the validation set, training data needs to be passed on the path for several iterations. This involves only forward propagation and no gradient backward. It aims to compute batch statistics.

Algorithm 2: Search on Supernet with EA

Input : Trained supernet model S, Detector $\mathcal{D}$, Population size $|\mathbf{P}|$, Parent size $|\mathcal{P}|$, Detection validation set $\mathbb{V}_{Det}$, Small subset of detection training set $\mathbb{T}_{Det_S}$, Evolution iterations $\mathrm{I_E}$, FLOPs constraint η.

$\mathbf{P}^{(0)} \leftarrow$ `random-initialize`$(|\mathbf{P}|, \eta)$
$j \leftarrow 0$
$f_{best}, \theta_{best} \leftarrow (0, \varnothing)$
while $j < \mathrm{I_E}$ **do**
  for $P \in \mathbf{P}^{(j)}$ **do**
    $S(P.\theta) \leftarrow$ `collectBN`$(\mathcal{D}(S(P.\theta)), \mathbb{T}_{Det_S})$
    $P.f \leftarrow$ `evaluate`$(\mathcal{D}(S(P.\theta)), \mathbb{V}_{Det})$
    if $P.f > f_{best}$ **then**
      $f_{best}, \theta_{best} \leftarrow (P.f, P.\theta)$
    end
  end
  $\mathcal{P} \leftarrow$ `select-topk`$(\mathbf{P}^{(j)}, |\mathcal{P}|)$
  $j \leftarrow j + 1$
  $\mathbf{P}_{mut} \leftarrow$ `mutate`$(\mathcal{P}, \frac{|\mathbf{P}|}{2}, \eta)$
  $\mathbf{P}_{crs} \leftarrow$ `crossover`$(\mathcal{P}, \frac{|\mathbf{P}|}{2}, \eta)$
  $\mathbf{P}^{(j)} \leftarrow \{\mathbf{P}_{mut}, \mathbf{P}_{crs}\}$
end

Output: The best architecture θ_{best}

Batch Normalization

- Typically, the parameters of BN, during fine-tuning, are fixed as the pre-training batch statistics.
- In DetNAS, the supernet is trained in a path-wise manner. The batch statistics of one node are obviously not same on different paths.
- When finetuning the supernet on the detection dataset, they replace the conventional BN with SyncBN to compute batch statistics across multiple GPUs. This enables the supernet finetuning by increasing the effective batch size for BN.

Results

Table 3. Results of different backbones

Backbone	ImageNet Classification		Object Detection with FPN on COCO					
	FLOPs	Accuracy	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet-50	3.8G	76.15	36.7	58.4	39.6	21.1	39.8	48.1
ResNet-101	7.6G	**77.37**	39.4	61.2	43.4	22.6	42.9	51.4
DetNet-59	4.8G	76.50	37.9	60.1	41.2	22.7	41.2	48.3
ShuffleNetV2-40	**1.3G**	77.18	39.2	60.8	42.4	23.6	42.3	52.2
DetNASNet	**1.3G**	77.20	**40.0**	61.5	43.6	23.3	42.5	53.8

Table 4. Comparisons in ShuffleNetV2 search space.

	FPN		RetinaNet	
	VOC (mAP, %)	COCO (mmAP, %)	VOC (mAP, %)	COCO (mmAP, %)
ShuffleNetV2-20	$80.6_{(73.1)}$	$34.8_{(73.1)}$	$79.4_{(73.1)}$	$32.1_{(73.1)}$
ClsNASNet	$78.5_{(74.3)}$	$35.1_{(74.3)}$	$76.5_{(74.3)}$	$31.2_{(74.3)}$
Random	$80.9\pm 0.2_{(73.9\pm0.2)}$	$35.9\pm0.3_{(73.9\pm0.2)}$	$79.0\pm0.7_{(73.9\pm0.2)}$	$32.5\pm0.4_{(73.9\pm0.2)}$
DetNAS-S	$81.1_{(73.8)}$	$\mathbf{35.9}_{(74.3)}$	$79.9_{(74.2)}$	$32.8_{(74.2)}$
DetNAS-P	$\mathbf{81.5}_{(74.0)}$	$\mathbf{36.4}_{(74.0)}$	$\mathbf{80.1}_{(74.1)}$	$\mathbf{33.3}_{(73.9)}$

* The subscript numbers are the ImageNet classification accuracy of the pretrained models. All models in this figure are trained with hyper-parameters described in Section 4.1. Their FLOPs are all similar and under 300M.

Summary

- The authors propose DetNAS to search architectures for backbones of detectors.
- The authors prove the effectiveness of DetNAS with multidimensional experiments. It works in different detectors (two-stage FPN and one-stage RetinaNet), various datasets (COCO and VOC) and different scheme (ImageNet pre-training and training from scratch).
- The searched networks have consistently better performance than the architectures searched from ImageNet on the target task. DetNASNet, the main result architecture, achieves 40.0 mAP on COCO and outperforms ResNet-101 with 5x less FLOPs complexity.