Single Path One-Shot Neural Architecture Search with Uniform Sampling

Zichao Guo*, Xiangyu Zhang*, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, Jian Sun

Megvii Technology

Tsinghua University, Hong Kong University of Science and Technology

{guozichao, zhangxiangyu, hengwen, weiyichen, sunjian}@megvii.com

muhy17@mails.tsinghua.edu.cn, zliubq@connect.ust.hk

Qing Lyu

5/8/2019

Contribution

- proposed a Single Path One-Shot mode to achieve NAS on large dataset

NAS Background

- Nested Approaches
  - High computation cost
  - Used on small dataset (CIFAR 10)

$$w_a = \underset{w}{\operatorname{argmin}} \mathcal{L}_{\text{train}} \left(\mathcal{N}(a, w) \right), \tag{1}$$

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \text{ACC}_{\text{val}} \left(\mathcal{N}(a, w_a) \right), \tag{2}$$

NAS Background

- Weight Sharing Approaches
  - The architecture search space $\mathcal{A}$ is encoded in a supernet $\mathcal{N}(\mathcal{A}, W)$
  - convert the discrete architecture search space into a continuous one

$$(\theta^*, W_{\theta^*}) = \underset{\theta, W}{\operatorname{argmin}} \mathcal{L}_{train}(\mathcal{N}(\mathcal{A}(\theta), W)). \qquad (4)$$

- Challenges
  - the weights of the graph nodes in the supernet depend on each other and become deeply coupled during optimization
  - joint optimization of architecture parameter θ and weights W introduces further coupling

NAS Background

- One-shot Approaches
  - the supernet training and architecture search are decoupled, in two sequential steps

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \, \mathcal{L}_{\text{train}} \left(\mathcal{N}(\mathcal{A}, W) \right) . \tag{5}$$

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \text{ACC}_{\text{val}} \left(\mathcal{N}(a, W_{\mathcal{A}}(a)) \right) . \tag{6}$$

  - architecture weights are properly initialized
  - weights of the graph nodes in the supernet are still coupled

Method

- Single path one-shot supernet
  - Choice block search
  - Channel number search
  - Mixed-precision quantization search
- Uniform sampling
- Evolutionary Architecture Search

Method: objective function

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \mathcal{L}_{\text{train}} \left(\mathcal{N}(\mathcal{A}, W) \right) . \quad (5)$$

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \mathbb{E}_{a \sim \Gamma(\mathcal{A})} \left[\mathcal{L}_{\text{train}}(\mathcal{N}(a, W(a))) \right] , \quad (7)$$

where $\Gamma(\mathcal{A})$ is a prior distribution of $a \in \mathcal{A}$.

- In each step of optimization, an architecture a is randomly sampled
- Only weights W(a) are activated and updated

Method: single path and choice blocks

- Single path to reduce weights coupling

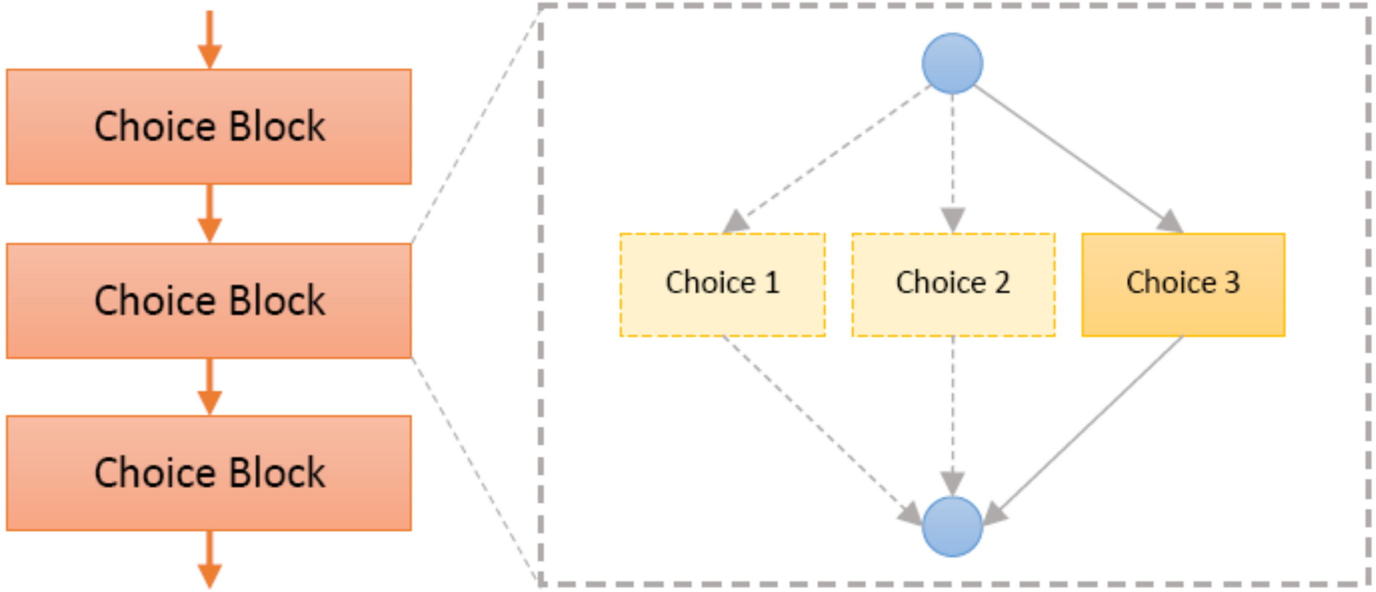
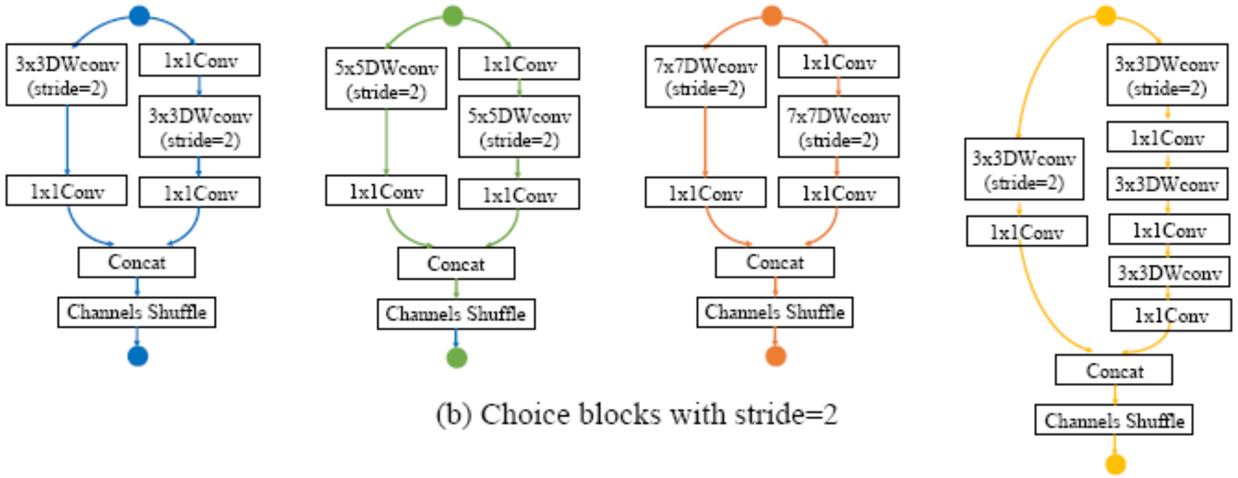


Figure 1. Architecture of a single path supernet. It consists of a series of *choice blocks*. Each has several *choices*. Only one choice is invoked at the same time.

Method: model and choice blocks

input shape	block	channels	repeat	stride
$224^2 \times 3$	3×3 conv	16	1	2
$112^2 \times 16$	CB	64	4	2
$56^2 \times 64$	CB	160	4	2
$28^2 \times 160$	CB	320	8	2
$14^2 \times 320$	CB	640	4	2
$7^2 \times 640$	1×1 conv	1024	1	1
$7^2 \times 1024$	GAP	-	1	-
1024	fc	1000	1	-

Table 2. Supernet architecture. *CB* – choice block. *GAP* – global average pooling. The "stride" column represents the stride of the first block in each repeated group.



(b) Choice blocks with stride=2

Method: channel number search and mixed-precision quantization search

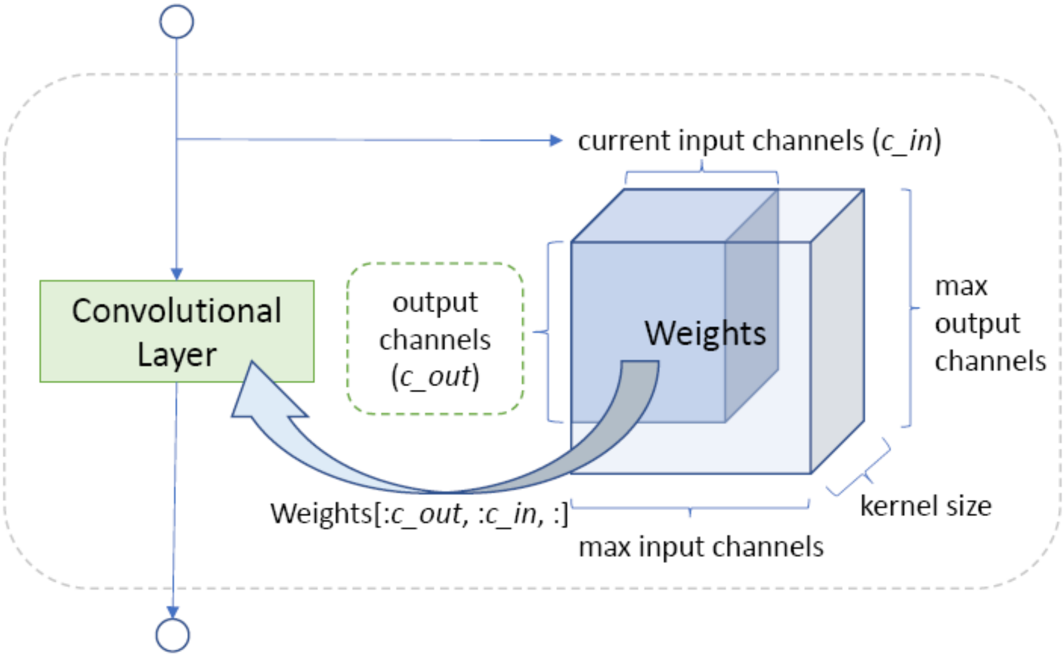


Figure 4. *Choice block* for channel number search.

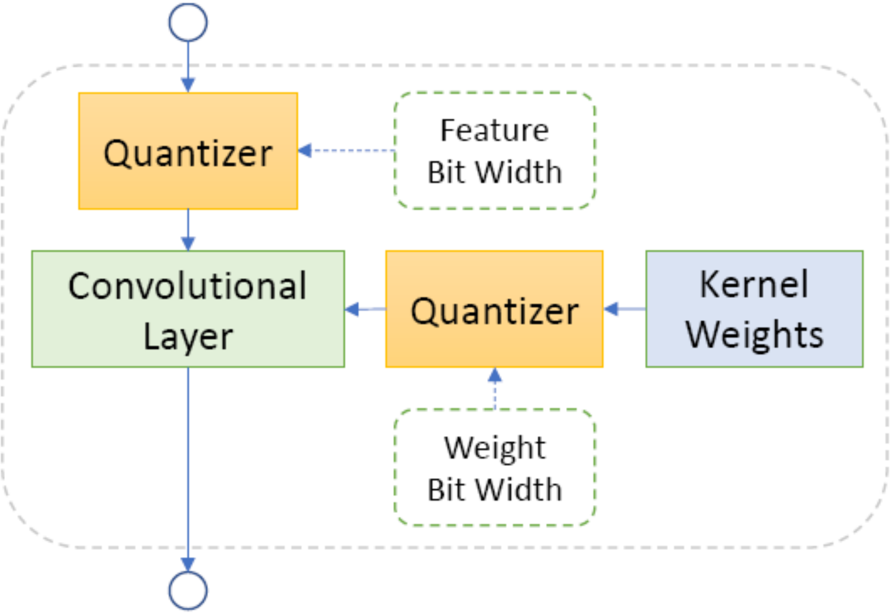
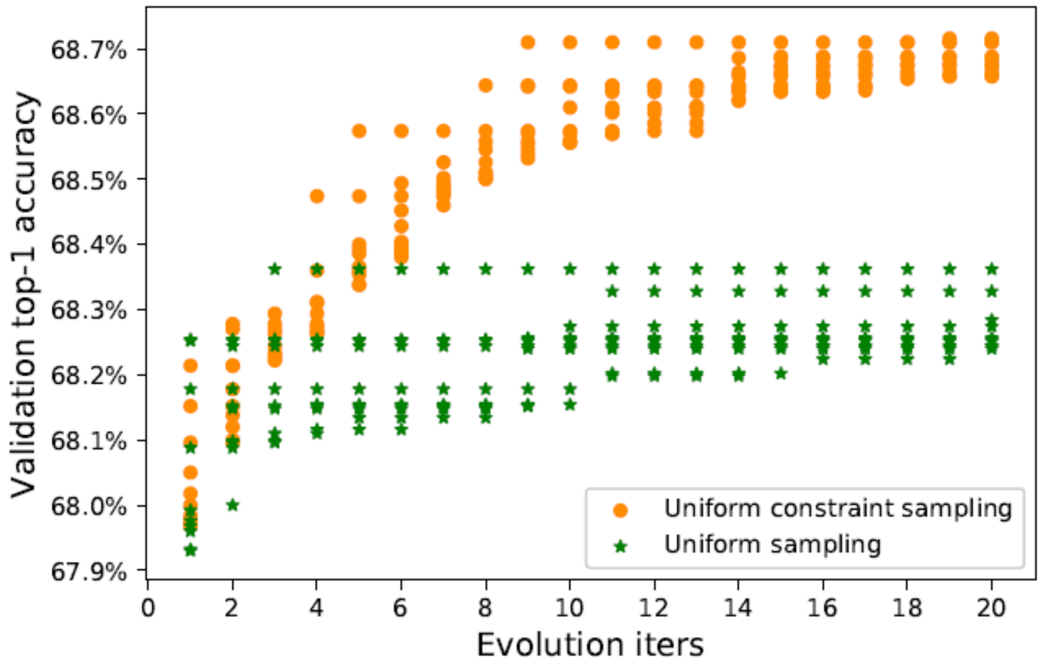


Figure 5. *Choice block* for mixed-precision quantization search.

Method: uniform sampling

- The prior distribution τ(A) could be important
- Empirically find that uniform sampling is good enough
- Treat all architectures equally



Method

- Evolutionary Architecture Search

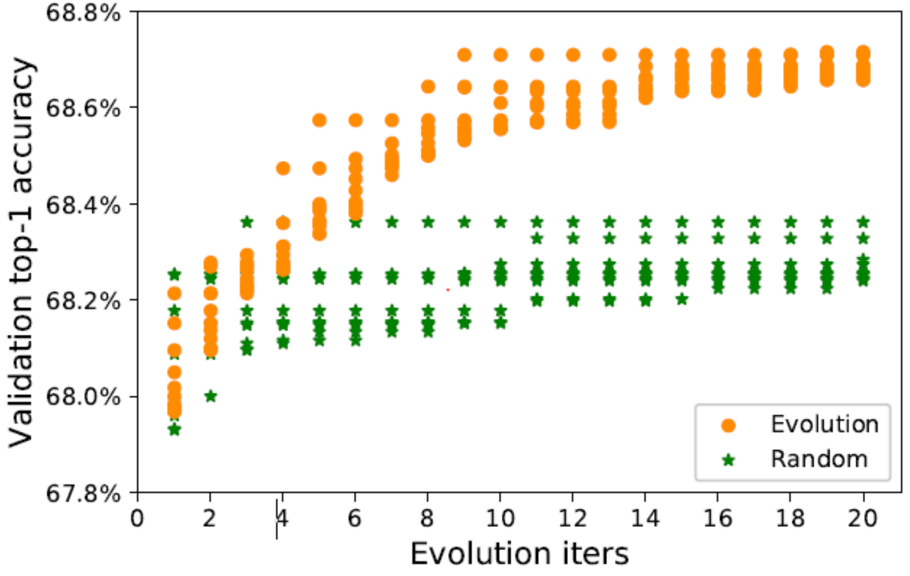


Figure 3. Evolutionary vs. random architecture search.

Algorithm 1 Evolutionary Architecture Search

Input: *supernet weights* $\mathrm{W}_{\mathcal{A}}$, *population size* P, *architecture constraints* $\mathcal{C}$, *max iteration* $\mathcal{T}$, *validation dataset* D_{val}

Output: *the architecture with highest validation accuracy under architecture constraints*

$\mathrm{P}_0 := Initialize\_population(P, \mathcal{C})$;
$n := P/2$; # Crossover number
$m := P/2$; # Mutation number
$prob := 0.1$; # Probability to mutate
$\mathrm{Topk} := \emptyset$;
for $i = 1 : \mathcal{T}$ **do**
    $\mathrm{ACC}_{i-1} := Inference(\mathrm{W}_{\mathcal{A}}, \mathrm{D}_{val}, \mathrm{P}_{i-1})$;
    $\mathrm{Topk} := Update\_Topk(\mathrm{Topk}, \mathrm{P}_{i-1}, \mathrm{ACC}_{i-1})$;
    $\mathrm{P}_{crossover} := Crossover(\mathrm{Topk}, n, \mathcal{C})$;
    $\mathrm{P}_{mutation} := Mutation(\mathrm{Topk}, m, prob, \mathcal{C})$;
    $\mathrm{P}_i := \mathrm{P}_{crossover} \cup \mathrm{P}_{mutation}$;
end for
return the entry with highest accuracy in Topk;

Result

- Dataset
  - ImageNet

- Training
  - The batch size is 1024
  - Supernet training for 120 epochs (150000 iterations)
  - The best architecture training for 240 epochs (300000 iterations)

- The size of the search space is 420

Result: building block search

model	FLOPs	top-1 acc(%)
all choice_3	324M	73.4
all choice_5	321M	73.5
all choice_7	327M	73.6
all choice_x	326M	73.5
random select (5 times)	~320M	~73.7
SPS + random search	323M	73.8
ours (fully-equipped)	319M	**74.3**

Table 3. Results of building block search. *SPS* – single path supernet (Sec. 3.2).

Result: channel search

Model	FLOPs	Top-1 acc(%)
all choice_3	324M	73.4
rand sel. channels (5 times)	$\sim$ 323M	$\sim$ 73.1
choice_3 + channel search	329M	**73.9**
rand sel. blocks + channels	$\sim$ 325M	$\sim$ 73.4
block search	319M	74.3
block search + channel search	328M	**74.7**
MobileNet V1 (0.75x) [10]	325M	68.4
MobileNet V2 (1.0x) [23]	300M	72.0
ShuffleNet V2 (1.5x) [17]	299M	72.6
NASNET-A [38]	564M	74.0
PNASNET [13]	588M	74.2
MnasNet [24]	317M	74.0
DARTS [15]	595M	73.1
Proxyless-R (mobile)* [4]	320M	74.2 (74.6)
FBNet-B* [26]	295M	74.1 (74.1)

Table 4. Results of channel search. * Performances are reported in the form "x (y)", where "x" means the accuracy retrained by us and "y" means accuracy reported by the original paper.

Result: compared with SOTA NAS methods

baseline network	FLOPs	latency	top-1 acc(%) baseline	top-1 acc(%) ours (same FLOPs)	top-1 acc(%) ours (same latency)
FBNet-A [26]	249M	13ms	73.0 (73.0)	**73.2**	**73.3**
FBNet-B [26]	295M	17ms	74.1 (74.1)	**74.2**	**74.8**
FBNet-C [26]	375M	19ms	74.9 (74.9)	**75.0**	**75.1**
Proxyless-R (mobile) [4]	320M	17ms	74.2 (74.6)	**74.5**	**74.8**
Proxyless (GPU) [4]	465M	22ms	74.7 (75.1)	**74.8**	**75.3**

Table 5. Compared with state-of-the-art *NAS* methods [26, 4] *using the same search space*. The latency is evaluated on a single *NVIDIA Titan XP* GPU, with $batchsize = 32$. Accuracy numbers in the brackets are reported by the original papers; others are trained by us. All our architectures are searched from the **same** supernet via evolutionary architecture optimization (see Sec. 3.4).

Result: mixed-precision quantization search

method	BitOps	top-1 acc(%)
ResNet-18	float point	70.9
2W2A	6.32G	65.6
ours	**6.21G**	**66.4**
3W3A	14.21G	68.3
DNAS [27]	15.62G	68.7
ours	**13.49G**	**69.4**
4W4A	25.27G	69.3
DNAS [27]	25.70G	**70.6**
ours	**24.31G**	70.5
ResNet-34	float point	75.0
2W2A	13.21G	70.8
ours	**13.11G**	**71.5**
3W3A	29.72G	72.5
DNAS [27]	38.64G	73.2
ours	**28.78G**	**73.9**
4W4A	52.83G	73.5
DNAS [27]	57.31G	74.0
ours	**51.92G**	**74.6**

Table 6. Results of mixed-precision quantization search. "kWkA" means k-bit quantization for all the weights and activations.

Result: search cost

Method	Proxyless	FBNet	Ours
GPU memory cost (8 GPUs in total)	37G	63G	24G
Training time	15 Gds	20 Gds	12 Gds
Search time	0	0	$<$1 Gds
Retrain time	16 Gds	16 Gds	16 Gds
Total time	31 Gds	36 Gds	29 Gds

Table 7. Search Cost. *Gds* - GPU days

Conclusion

- Propose a single path one-shot NAS approach that reduce weight coupling in the supernet
- comparing to previous works on a large dataset (ImageNet) verify that the proposed approach is SOTA in terms of
  - accuracy
  - memory consumption
  - training time
  - architecture search efficiency
  - flexibility