Supplementary Materials for
GreedyNASv2: Greedier Search with a Greedy Path Filter

A. Details of Search Spaces

A.1. Macro structures

In this paper, we conduct experiments on two macro structures of supernet, as presented in Table 6 and Table 7. The MobileNetV2 supernet is used for *MB-SE*, *MB-SE+MixConv*, and *MB-SE+MixConv+shuffle* search spaces, while the *Res-50-SE* adopts the same structure as ResNet-50 [5] in Table 7.

Table 6. Macro structure of our MobileNetV2 search space. *input* denotes the input feature size for each layer, *channels* means the output channels of the layer, *repeat* denotes the repeat times of stacking the same blocks, and *stride* is for the stride of first block when stacked for multiple times.

input	block	channels	repeat	stride
$224^2 \times 3$	3×3 conv	32	1	2
$112^2 \times 32$	MB1_K3	16	1	1
$112^2 \times 16$	Choice Block	32	4	2
$56^2 \times 32$	Choice Block	40	4	2
$28^2 \times 40$	Choice Block	80	4	2
$14^2 \times 80$	Choice Block	96	4	1
$14^2 \times 96$	Choice Block	192	4	2
$7^2 \times 192$	Choice Block	320	1	1
$7^2 \times 320$	1×1 conv	1280	1	1
$7^2 \times 1280$	global avgpool	-	1	-
1280	FC	1000	1	-

Table 7. Macro structure of our Res-50-SE search space.

input	block	channels	repeat	stride
$224^2 \times 3$	7×7 conv	64	1	2
$112^2 \times 64$	3×3 max pool	64	1	2
$56^2 \times 64$	Choice Block	256	3	1
$56^2 \times 256$	Choice Block	512	4	2
$28^2 \times 512$	Choice Block	1024	6	2
$14^2 \times 1024$	Choice Block	2048	3	2
$7^2 \times 2048$	global avg pool	-	1	-
2048	FC	1000	1	-

A.2. Candidate operations

The candidate operations in *Choice Block* of each supernet are summarized as follows.

• **MB-SE.** Following the previous NAS methods [13, 14, 17], we conduct the same searching operations in *MB-SE* search space, which consists of 13 MobileNetV2 [12] blocks with optional SE [6] module, as summarized in Table 8.

• **MB-SE+MixConv.** We design a new MobileNetV2 search space with additional MixConv blocks [15], which aims to mix the outputs of different kernel sizes (3×3, 5×5, and 7×7) of depth-wise convolution in MobileNetV2 block.

• **MB-SE+MixConv+Shuffle.** To further validate our performance on a larger search space, we add the ShuffleNetV2 [11] blocks in SPOS [4].

• **Res-50-SE.** We leverage the blocks in ResNet [5] and ResNeXt [16] to build our *Res-50-SE* search space, and all of them are equipped with additional SE modules. We design blocks with different kernel sizes (3, 5, and 7), and a *ratio* is used to control the intermediate number of channels, which has choices 0.5, 1.0, and 1.5, *e.g.*, 0.5 means that the number of intermediate channels is $0.5\times$ compared to the original one. The total number of candidate operations is 19, with an additional *ID* operation for layer removal.

The detailed settings of candidate operations are summarized in Table 8 and Table 9.

Table 8. Candidate operations in our mobile search spaces.

search space	block type	expansion ratio	kernel size	SE
-	MB1_K3	1	3	no
MB-SE	ID	-	-	-
	MB3_K3	3	3	no
	MB3_K5	3	5	no
	MB3_K7	3	7	no
	MB6_K3	6	3	no
	MB6_K5	6	5	no
	MB6_K7	6	7	no
	MB3_K3_SE	3	3	yes
	MB3_K5_SE	3	5	yes
	MB3_K7_SE	3	7	yes
	MB6_K3_SE	6	3	yes
	MB6_K5_SE	6	5	yes
	MB6_K7_SE	6	7	yes
MixConv	MB3_MIX	3	3+5+7	no
	MB6_MIX	6	3+5+7	no
	MB3_MIX_SE	3	3+5+7	yes
	MB6_MIX_SE	6	3+5+7	yes
Shuffle	Shuffle_3	-	3	yes
	Shuffle_5	-	5	yes
	Shuffle_7	-	7	yes
	Shuffle_x	-	3 + 3 + 3	yes

Table 9. Candidate operations in our Res-50-SE search space.

block type	basic block	ratio	kernel size	SE
ID	-	-	-	-
ResNet_K3_0.5×	ResNet	0.5	3	yes
ResNet_K3_1×	ResNet	1.0	3	yes
ResNet_K3_1.5×	ResNet	1.5	3	yes
ResNet_K5_0.5×	ResNet	0.5	5	yes
ResNet_K5_1×	ResNet	1.0	5	yes
ResNet_K5_1.5×	ResNet	1.5	5	yes
ResNet_K7_0.5×	ResNet	0.5	7	yes
ResNet_K7_1×	ResNet	1.0	7	yes
ResNet_K7_1.5×	ResNet	1.5	7	yes
ResNet_K3_1×	ResNeXt	1.0	3	yes
ResNeXt_K3_1.5×	ResNeXt	1.5	3	yes
ResNeXt_K5_0.5×	ResNeXt	0.5	5	yes
ResNeXt_K5_1×	ResNeXt	1.0	5	yes
ResNeXt_K5_1.5×	ResNeXt	1.5	5	yes
ResNeXt_K7_0.5×	ResNeXt	0.5	7	yes
ResNeXt_K7_1×	ResNeXt	1.0	7	yes
ResNeXt_K7_1.5×	ResNeXt	1.5	7	yes

B. Implementing Details of PU Learning

In the training of supernet, we train the path filter using VPU [1] every $t = 5$ epoch. At the first time we train it, the weights of the path filter are randomly initialized, then the following training fine-tunes the weights obtained in the previous training.

B.1. Complete learning objective in VPU

The core idea of VPU is the proposed variational loss as in Eq.(6). Besides, to further alleviate the over-fitting problem, VPU incorporates a MixUp [18] based consistency regularization term to the variational loss (Eq.(6)) as

$$\mathcal{L}_{\text{reg}}(\Phi) = \mathbb{E}_{\tilde{\Phi},\tilde{\boldsymbol{a}}}[(\log\tilde{\Phi} - \log\tilde{\Phi}(\tilde{\boldsymbol{a}}))^2], \tag{10}$$

with

$$\begin{aligned} &\gamma \overset{\text{iid}}{\sim} \text{Beta}(\sigma, \sigma), \\ &\tilde{\boldsymbol{a}} = \gamma \cdot \boldsymbol{a}' + (1-\gamma) \cdot \boldsymbol{a}'', \\ &\tilde{\Phi} = \gamma \cdot 1 + (1-\gamma) \cdot \Phi(\boldsymbol{a}''). \end{aligned} \tag{11}$$

Here $\tilde{\boldsymbol{a}}$ is an architecture generated by mixing randomly selected $\boldsymbol{a}' \in \mathcal{P}$ and $\boldsymbol{a}'' \in \mathcal{U}$, and $\tilde{\Phi}$ represents the *guessed* probability $\mathbb{P}(y = +1|\boldsymbol{a} = \tilde{\boldsymbol{a}})$ constructed by the linear interpolation of the true label and that predicted by Φ, σ is a hyper-parameter to control the MixUp percentage. Unlike the original MixUp on images, our architecture vector $\boldsymbol{a}$ is a tuple of discrete integers without semantic features, therefore, we conduct MixUp after the embedded features $\boldsymbol{A}$, *i.e.*,

$$\boldsymbol{A}_{\tilde{\boldsymbol{a}}} = \gamma \cdot \boldsymbol{A}_{\boldsymbol{a}'} + (1-\gamma) \cdot \boldsymbol{A}_{\boldsymbol{a}''}. \tag{12}$$

Complete form of loss function in VPU. The complete loss function to train our path filter is as below:

$$\mathcal{L}(\Phi) = \mathcal{L}_{\text{var}}(\Phi) + \lambda\mathcal{L}_{\text{reg}}(\Phi). \tag{13}$$

In our experiments, we set $\sigma = 0.3$ and $\lambda = 0.2$ following the original configurations in VPU.

C. Searching Results of Baseline NAS Methods on Res-50-SE Search Space

In this paper, we propose a new search space named *Res-50-SE* for searching ResNet-like models. Here we conduct experiments to compare our method with baselines SPOS [4] and GreedyNAS [17]. As the results summarized in Table 10, we can see that our *GreedyNASv2-L* obtains the highest accuracy with the minimal cost. Besides, for GreedyNAS, since each architecture has $\sim$ 4G FLOPs, the computation cost of multi-path sampling could be noticeably higher than the mobile search spaces, and the training cost is larger than GreedyNASv2 as a result.

Table 10. Evaluation results of *Res-50-SE* search space on ImageNet. The results of SPOS and GreedyNAS are obtained by our implementations.

Methods	Top-1 (%)	Top-5 (%)	FLOPs (M)	Params (M)	Training epochs	Training cost (GPU days)	Search number
SPOS [4]	80.6	95.1	4153	27.8	120	15.4	1000
GreedyNAS [17]	80.8	95.2	4125	28.1	49	11.3	1000
GreedyNASv2-L	**81.1**	95.4	4098	26.9	57	9	500

D. Transfer learning on object detection task

We transfer our searched models to verify the generalization performance on object detection task. Concretely, we train both two-stage Faster R-CNN with Feature Pyramid Networks (FPN) [8] and one-stage RetinaNet [9] networks on COCO dataset [10], and report the validation mAP in Table 11. Note that for fair comparisons, we train the networks using the default configurations in mmdetection [2], with only modifications on backbone models. The results show that our obtained models significantly outperform the baseline models.

Table 11. Evaluation results on COCO dataset.

Backbone	ImageNet Top-1 (%)	FPN mAP (%)	RetinaNet mAP (%)
ResNet-50	76.1	37.4	36.5
GreedyNASv2-L	81.1	41.7 (+4.3)	40.9 (+4.4)
MobileNetV2	72.0	32.1	30.5
GreedyNASv2-S	77.5	35.4 (+3.3)	34.9 (+4.4)

E. More Ablation Studies

E.1. Effects of path-level shrinkage and operation-level shrinkage

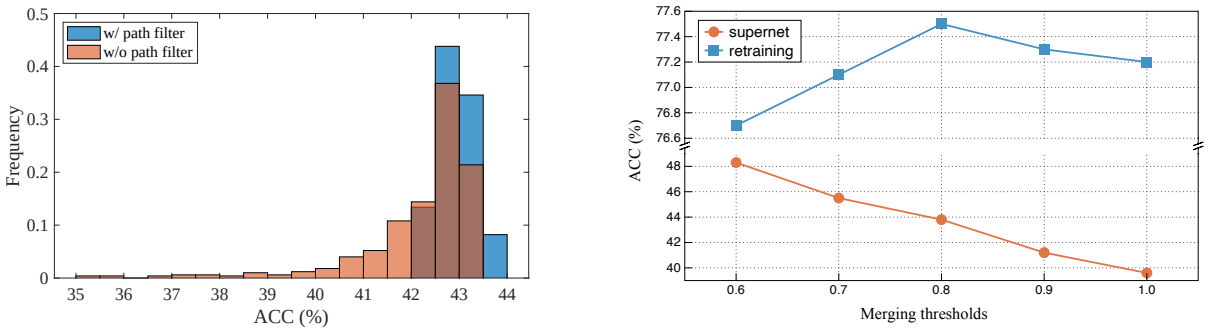
To validate the effectiveness of the proposed path-level shrinkage and operation-level shrinkage methods, we conduct experiments to ablate these two components in GreedyNASv2, as shown in Table 12.

Table 12. Effects of path-level and operation-level shrinkages on *MB-SE+MixConv+Shuffle (large)* search space.

Method	Path-level shrinkage	Operation-level shrinkage	ACC in retraining (%)	ACC on supernet (%)
SPOS [4]	-	-	75.5	33.4
GreedyNAS [17]	✔	-	76.5	35.1
GreedyNASv2	✔	✔	**77.5**	**43.8**
GreedyNASv2	✘	✔	76.8	42.1
GreedyNASv2	✔	✘	77.2	39.6

E.2. Effect of path filter in search

In GreedyNASv2, we use the learned path filter to filter the predicted weak paths during the search. Now we conduct experiments on *MB-SE+MixConv+Shuffle* search space to show the effectiveness of path filter in search. As the histogram of searched accuracies shown in Figure 7 (a), searching with a path filter can reject a larger number of weak paths and obtain higher accuracies, showing that the path filter can boost the searching efficiency.



(a) (b)

Figure 7. (a): Histogram of accuracies of searched paths on supernet with or without using path filter. (b): Supernet and retraining accuracies of different operation merging thresholds. Specifically, the threshold of 1.0 denotes no merging.

E.3. Effects of different operation merging thresholds

In our operation-level shrinkage, the operation pair with a similarity large than a certain threshold would be treated as similar pair; then, we will merge them into one operation and obtain a smaller search space as a result. Here we conduct experiments to show the performance of different merging thresholds. We train the *MB-SE+MixConv+Shuffle* supernet using GreedyNASv2 with merging thresholds 0.6, 0.7, 0.8, 0.9, and 1.0, respectively, and report the supernet and retraining accuracies of the searched models in Figure 7 (b). We can see that the smaller threshold would have more operations being merged, and thus the accuracy on supernet would be higher. However, too aggressive mergings (thresholds 0.6 and 0.7) would hurt the diversity of the search space; therefore, the performance of searched models would worsen.

E.4. Validate the correctness of operation similarity

To validate the correlation between our learned operation similarities and their corresponding evaluation performance, we conduct experiments to measure the rank correlation of evaluation performance in each operation pair. Concretely, we split the operation pairs on *MB-SE* search space into *similar*, *dissimilar*, and *random* set, the *similar* (*dissimilar*) set contains 10 pairs with highest (lowest) learned similarities, while the *random* set are built with randomly generated pairs. We first measure the rank correlation of each pair independently, then report their mean correlation as the correlation of the set. Specifically, for the measurement of the rank correlation of each pair, we randomly generate 100 paths containing the first operation, then evaluate their performance of validation set on a trained supernet, resulting in performance vector $\boldsymbol{x}$. For another operation, we use it to replace the first operation in generated paths and obtain performance vector $\boldsymbol{y}$. If these two operations in a pair have similar performance, vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ will obtain similar values for each element. We then use Spearman's [3] and Kenall's Tau [7] rank correlation to measure this similarity in performance. Note that we use the supernet learned by uniform sampling for fair evaluation without greedy biases.

As shown in Table 13, the learned similar pairs obtain very high similarities (rank correlations), indicating that our learned similarity can well reflect the similarity in performance; therefore, we can confidently leverage the learned similarities to merge operations.

Table 13. Rank correlations of the evaluation results of similar, dissimilar, and random pairs identified by the path filter on supernet.

Pairs	Mean similarity	Rank correlation (%)	
		Spearman's	Kendall's Tau
similar	0.9118	**99.26**	**93.68**
dissimilar	-0.2183	73.38	69.58
random	0.3412	83.62	78.31

F. Visualization of our searched architectures

Our searched *GreedyNASv2-S* and *GreedyNASv2-L* are visualized in Figure 8.

(a) GreedyNASv2-S (b) GreedyNASv2-L

Figure 8. Visualization of our searched architectures.

References

[1] Hui Chen, Fangqing Liu, Yin Wang, Liyue Zhao, and Hao Wu. A variational approach for learning from positive and unlabeled data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14844–14854. Curran Associates, Inc., 2020. 2

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 3

[3] Yadolah Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008. 4

[4] 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 *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1, 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[6] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1

[7] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 4

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 3

[11] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 1

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[13] Xiu Su, Tao Huang, Yanxi Li, Shan You, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Prioritized architecture sampling with monto-carlo tree search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10968–10977, 2021. 1

[14] Xiu Su, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. K-shot NAS: learnable weight-sharing for NAS with k-shot supernets. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 9880–9890. PMLR, 2021. 1

[15] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. In *BMVC*, page 74. BMVA Press, 2019. 1

[16] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 1

[17] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1999–2008, 2020. 1, 3

[18] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 2