A. Detailed Experimental Settings

In this section, we will describe the detailed experimental settings of our MCT-NAS w.r.t. training and searching. In general, we used the same MCT during training and searching, and thus the constructed MCT with prioritized sampled paths can boost the search in terms of efficiency and search performance.

Supernet training with MCT-NAS. For ImageNet dataset, we use the same strategy follow [35, 9]. In detail, we use a batch size of 1024, and train the supernet using a SGD optimizer with 0.9 momentum. A cosine annealing strategy is adopted with an initial learning rate 0.12, which decays 120 epochs. For sampling architectures, we first warm up the supernet with 50% of overall training epochs (60 epochs) with uniform sampling. Then for the last 60 epochs, we sample subnets with FLOPs reduction, *i.e.*, the FLOPs of sampled subnets must be within a certain range of FLOPs budget(*i.e.*, $0.9\times \sim 1.0\times$). In the 20% of following training epochs (61-85 epochs), we also uniformly sample subnets to construct the MCT. Afterward, in the last 34 epochs (86-120), we sample subnets with MCT and UCT function defined in Eq. (6) and Eq. (8). In detail, we adopt $C_1 = 0.1$ and $C_2 = 0.2$ for Eq. (8) and Eq. (9). Besides, for NAS-Bench-Macro, we follow the same split ratios of using uniform sampling and MCTS as on ImageNet. And other training strategies are the same as retraining on CIFAR-10.

Searching optimal structure with MCT-NAS. We use hierarchical node selection with MCT for architecture search. For sampling a path (subnet), we select the optimal nodes hierarchically from the root node of MCT with a threshold constant n_{thrd} of 6. If the average number of visits of its child nodes is lower than the n_{thrd}, we randomly sample paths consisting of those child nodes and then evaluate the paths using a batch (*i.e.*, 128) of validation data until the threshold reached. After selecting the leaf nodes, the specific subnet (structure) is obtained; we then evaluate it with the full validation dataset. Moreover, we repeat to sample subnets with this process until we reach our predefined number of searches (*i.e.*, 20). Afterward, we select the structure with the best validation accuracy to train from scratch for evaluation.

Retraining of searched optimal structures. To train the obtained architectures from scratch, we follow previous works [25, 35, 2], the network is trained using RMSProp optimizer with 0.9 momentum, and the learning rate is increased from 0 to 0.064 linearly in the first 5 epochs with batch size 512, and then decays 0.03 every 2.4 epochs. Besides, the exponential moving average on weights is also adopted with a decay rate 0.9999.

B. Details of Path Sampling in Supernet Training

The supernet training in our framework can be divided into three stages with different subnet sampling strategies, including the supernet warm-up, the MCT warm-up, and sampling with MCTS. Firstly, in the supernet warm-up stage, we sample each path uniformly for a better search space exploration, which is the same as [9, 2]. Then, in the MCT warm-up stage, we also sample each path uniformly but discard those who do not meet the computation budget (*i.e.*, a certain range of FLOPs budget). Architectures with low computation budget are expected to have lower performance, which makes them less worthy of being optimized and evaluated. In this way, we can construct the MCT more efficiently. Finally, we sample paths according to the UCT function in Eq. (8) to train the supernet more efficiently with exploration-exploitation strategy in MCTS. Our iterative procedure of training supernet is presented in Algorithm 2.

Algorithm 2: Path Sampling in Supernet Training

Input: the ratio of iterations for supernet warm-up W_S, the ratio of iterations for MCT warm-up W_M, training dataset $\mathcal{D}_{tr}$ the maximum training iterations N, FLOPs budget $\mathcal{B}$.

```
while training epochs < N do
    if training epochs < W_S × N then
        randomly sample a path p in the supernet;
        optimize the path p with dataset D_tr;
    else
        if training epochs < (W_S + W_M) × N then
            random sample a path p;
            while FLOPs of path p not in the certain range of B do
                re-sample a new path p;
            end
        else
            sample a path p based on MCT with Hierarchical Node Selection as Algorithm 1 ;
        end
        optimize the path p with dataset D_tr;
        updating the nodes in MCT which corresponds to path p with the training loss;
    end
end
```

C. Details of Search Space on ImageNet Dataset

The macros-structure of the supernet used in our experiments on ImageNet is as described in Table 4. The choices for building blocks are as listed in Table 5.

Table 4. The macro-structure of the supernet in our experiments on ImageNet. The mean of each column is as follow: "n" is the number of such stacked building blocks; "input" is the size of input feature map; "block" is the type of building block; "channels" is the number of output channels, i.e. the number of filters; and "stride" is the stride of the first block among several repeated building blocks. The block type "Choice Block" means selecting a candidate from Table 5.

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

D. Details of Proposed Benchmark: NAS-Bench-Macro

Meaning of NAS-Bench-Macro. The key challenge of the one-shot NAS algorithm lies in the evaluation and ranking reliability of the supernet, which can be reflected by the ranking correlation between the evaluation performances of all architectures on supernet and their actual performances. There are already several NAS benchmarks for the DARTS-like micro search space. However, currently, there is no (as far as we know) public benchmark for one-shot macro search space that provides performances of all networks by retraining from scratch. Therefore, we construct NAS-Bench-Macro with the controlled size of the search space for the toy experiment of verifying our method's effectiveness in terms of searching efficiency and performance. Indeed, we construct NAS-Bench-Macro with the knowledge from MobileNetV2 search space but remove 7×7 convolution for simplifying.

Table 5. The operation candidates for the MobileNetV2-like building blocks used in Table 4. "ID" denotes an identity mapping.

block type	expansion ratio	kernel	SE
MB1_K3	1	3	no
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

D.1. Details of search space on NAS-Bench-Macro

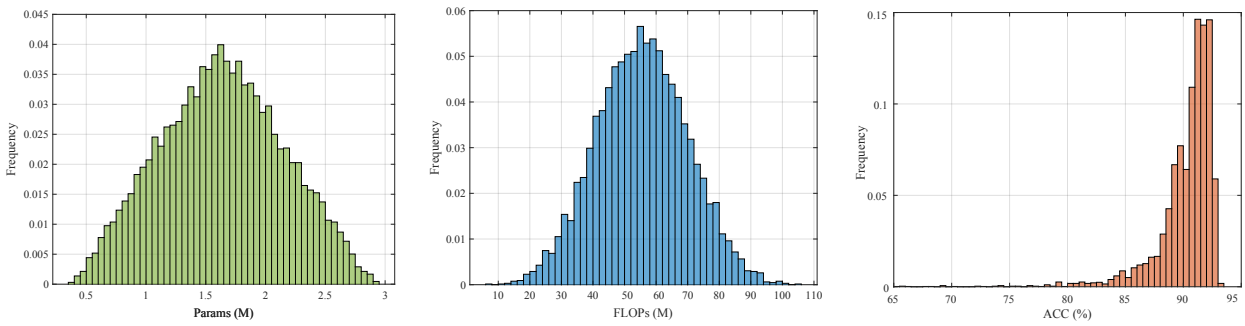
The macro structure of our search space is presented in Table 6.

Table 6. Macro structure of search space on NAS-Bench-Macro.

n	input	block	channel	stride
1	$32 \times 32 \times 3$	3×3 conv	32	1
2	$32 \times 32 \times 32$	Choice Block	64	2
3	$16 \times 16 \times 64$	Choice Block	128	2
3	$8 \times 8 \times 128$	Choice Block	256	2
1	$4 \times 4 \times 256$	1×1 conv	1280	1
1	$4 \times 4 \times 1280$	global avgpool	-	-
1	1280	FC	10	-

D.2. Statistics on CIFAR-10 training results

We analyze the distribution of parameters, FLOPs, and accuracies of all architectures in NAS-Bench-Macro, which are illustrated in Figure 7. From the results, we can infer that, although the FLOPs and parameters are distributed uniformly in the search space, but the accuracy performs differently, which might because when the capacity of model increases to a saturated level, the increment of capacity will not have remarkable performance increment; so the accuracies of a large number of architectures lie in a small range, this makes the NAS methods hard to rank those architectures accurately.



(a) Params (b) FLOPs (c) ACC

Figure 7. Histograms of architectures in NAS-Bench-Macro w.r.t. (a) Params, (b) FLOPs, and (c) ACC.

D.3. Rank correlation of parameters and FLOPs with accuracies on NAS-Bench-Macro

We measure the rank correlation between parameters and accuracies, FLOPs and accuracies, respectively. The results summarized in Table 7 show that the FLOPs has higher rank correlation coefficients than parameters. It indicates that the increment of FLOPs contributes more to accuracy than increasing parameters.

Table 7. Rank correlations of parameters and FLOPs with accuracies on NAS-Bench-Macro.

type	Spearman rho (%)	Kendall tau (%)
params	31.81	21.76
FLOPs	66.09	55.60

D.4. Visualization of best architecture

We visualize the best architecture of our NAS-Bench-Macro on CIFAR-10 dataset in Figure 8. The visualization shows that the architecture with the highest performance on CIFAR-10 tends to choose more large blocks (*MB6_K5*); however, an *Identity* block on the last layer is used probably for easing the optimization pressure and decreasing the total receptive field.

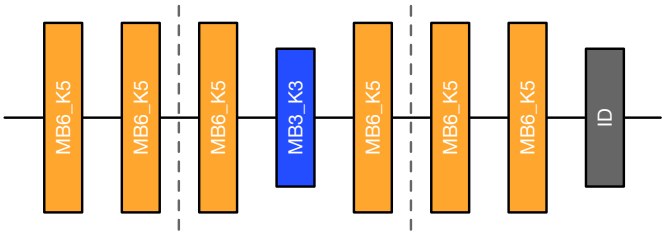


Figure 8. The best architecture on CIFAR-10 with 93.13% test accuracy, 2.0M parameters and 85.2M FLOPs.

E. Details of Rank Correlation Coefficient

In Section 4.1 and Section D.3, we rank the evaluation results of architectures on various validation sets with 1,000 and 50,000 samples. The obtained rankings are denoted as $\boldsymbol{r}$ and $\boldsymbol{s}$, respectively. To study the ranking consistency, we evaluate the correlation coefficient between them. Two metrics with different focuses are used, including the Kendall τ [15] and the Spearman ρ [21] correlation coefficient.

Firstly, the Kendall τ correlation coefficient focuses on the pairwise ranking performance. Considering a pair of index i and j such that $i < j$, if we have either both $\boldsymbol{r}_i > \boldsymbol{r}_j$ and $\boldsymbol{s}_i > \boldsymbol{s}_j$ or both $\boldsymbol{r}_i < \boldsymbol{r}_j$ and $\boldsymbol{s}_i < \boldsymbol{s}_j$, the sort order of $(\boldsymbol{r}_i, \boldsymbol{r}_j)$ and $(\boldsymbol{s}_i, \boldsymbol{s}_j)$ agree, and pairs $(\boldsymbol{r}_i, \boldsymbol{s}_i)$ and $(\boldsymbol{r}_j, \boldsymbol{s}_j)$ are said to be *concordant*. Otherwise, they disagree and the pairs are *disconcordant*. With the concordant and disconcordant pairs, we can formally define the Kendall τ correlation coefficient by

$$\tau_K = \frac{N_{\text{concordant}} - N_{\text{disconcordant}}}{N_{\text{all}}}, \tag{10}$$

where $N_{\text{concordant}}$ and $N_{\text{disconcordant}}$ are the numbers of concordant and disconcordant pairs, respectively, and $N_{\text{all}} = \binom{n}{2} = \frac{n(n-1)}{2}$ is the total number of possible pairs out of n overlapped elements. For efficient calculation, we can reformulate (10) into an explicit expression form:

$$\tau_K = \frac{2}{n(n-1)} \sum_{i<j} \text{sign}(\boldsymbol{r}_i - \boldsymbol{r}_j) \cdot \text{sign}(\boldsymbol{s}_i - \boldsymbol{s}_j), \tag{11}$$

where $\text{sign}(\cdot)$ is the sign function.

Secondly, the Spearman ρ correlation coefficient aims to evaluate to what degree a monotonic function fits the relationship between two random variables. If we consider $\boldsymbol{r}$ and $\boldsymbol{s}$ as two observation vectors of the random variables r and s, respectively, the Spearman ρ correlation coefficient can be calculated with the Pearson correlation coefficient:

$$\rho_S = \frac{\text{cov}(r, s)}{\sigma_r \sigma_s}, \tag{12}$$

where $\mathrm{cov}(\cdot,\cdot)$ is the covariance of two variables, and σ_r and σ_s are the standard deviations of r and s, respectively. In our experiment, the ranks are distinct integers. Therefore, Eq. (12) can be reformulated as:

$$\rho_S = 1 - \frac{6\sum_{i=1}^{n}(\boldsymbol{r}_i - \boldsymbol{s}_i)^2}{n(n^2-1)}, \tag{13}$$

where $n = 1000$ is the number of overlapped elements between $\boldsymbol{r}$ and $\boldsymbol{s}$.

F. More Ablation Studies

F.1. Comparison between different training epochs on supernet

In one-shot NAS, supernet as a performance evaluator highly correlates to the search performance. The more converged the supernet is, the more confident it will be on ranking subnets. However, increasing epochs of training also increase the computation cost; thus, we investigate the difference between different training epochs in this section. Concretely, we conduct experiments using our method with different training epochs on ImageNet, and measure their average validation accuracies and the corresponding train-from-scratch test accuracies of obtained architectures. The experiments are implemented with 330M FLOPs budget, and we search 20 subnets for each experiment.

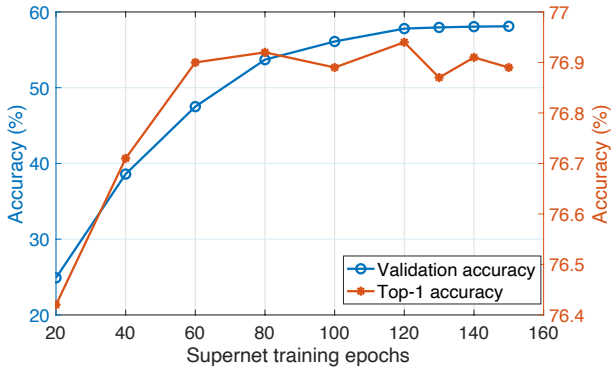


Figure 9. The Top-1 accuracy and average validation accuracy of searched architectures with different training epochs on supernet.

From the results illustrated in Figure 9, we find the average validation accuracy of searched structures benefits from the increase of training epochs, and it almost achieves the peak after training epochs $N = 120$; moreover, we can find that when the training epochs $N = 60$, the accuracy of obtained architecture is similar to the one with $N = 120$.

F.2. Effect of the threshold constant n_{thrd}

In MCT-NAS, the proposed hierarchical node selection aims to re-explore the less-visited nodes and evaluate the subnets more accurately with the defined threshold constant n_{thrd}. In detail, if the average number of visits of nodes in a layer is larger than n_{thrd}, we think it is promising to its reward, and thus the optimal node can be sampled with equation Eq. (6) and Eq. (9). To investigate the effect of the threshold constant n_{thrd} to the validation accuracy of searched subnets, we conduct experiments to search with different n_{thrd} on the same trained supernet of 330M FLOPs budget. In detail, we search 20 paths with different n_{thrd} and record the average validation accuracy as Figure 10.

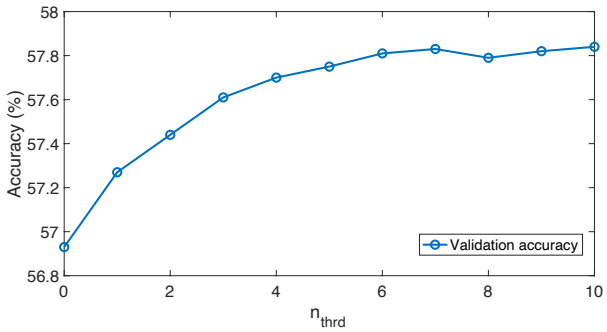


Figure 10. The validation accuracy of searched architectures with different threshold constant n_{thrd}.

Figure 10 shows that the larger n_{thrd} leads to higher average validation accuracy of searched structures. Concretely, the accuracy of searched paths gradually increased as n_{thrd} growing larger, which means that our proposed hierarchical node selection promotes to select for better paths. In addition, the performance tends to be stable when n_{thrd} grows larger than 6, which implies $n_{thrd} = 6$ is sufficient to search for optimal paths.

F.3. Trade-off between search number and n_{thrd}

During the search, the increments of search number and n_{thrd} both boost the search performance. However, under a certain upper limit of search cost, the search number should be in inversely proportional to the threshold constant n_{thrd}. Therefore, for a better trade-off between search number S_n and n_{thrd}, we conduct experiments with different combinations of S_n and n_{thrd}, which hold that

$$S_n \times n_{thrd} = C \tag{14}$$

where C is the target search cost coefficient, we implement the search with $C = 120$ on the same trained supernet and report the average validation accuracy of searched architectures as in Figure 11.

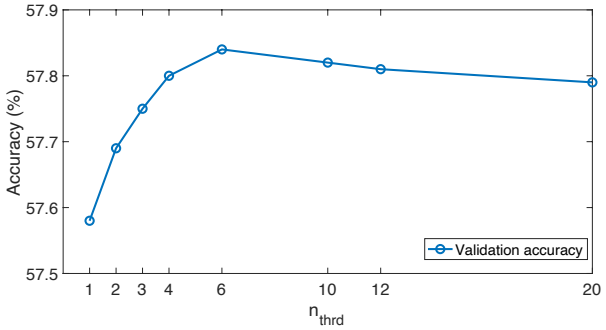


Figure 11. Average validation accuracy of searched architectures with different n_{thrd} under the sane search cost.

In detail, we notice that when n_{thrd} is small, the performance of the searched structure will increase with n_{thrd} growing larger; this is because a larger n_{thrd} promotes selecting nodes in MCT more accurately. Therefore, the performances of searched structures will be closer to the optimal one in the search space. However, when n_{thrd} goes beyond 6, the accuracy of selected structures tend to decrease slightly; since with a default search budget, larger n_{thrd} leads to a smaller search number, and thus it may hinder the search for the optimal structure.

F.4. Effect of MCT-NAS in supernet training

As illustrated in Algorithm 2, we investigate two kinds of warm-ups in MCT-NAS, *i.e.*, warm up of supernet and warm up of MCT. Besides, to investigate the ratio of iterations for these two terms (*i.e.*, W_S and W_M), we first conduct experiments with W_S by setting W_M to 0, *i.e.*, we randomly uniform sample paths to update supernet and then select paths by FLOPs reduction and start to update MCT, as the blue line shows in Figure 12. In detail, we achieve the best performance when W_S is set to 0.5. Afterward, we investigate the effect of W_M with W_S set to 0.5, as the red line shows in Figure 12. With W_S set to 0.2 (*i.e.*, 70% of training epochs), our searched structures achieve the best performance. As a result, with 50% of overall training epochs, our supernet can effectively help to rank the performance of different structures; and then with 20% training epochs, MCT is being well initialized and can be leveraged to effectively select paths in combination with UCT function (as defined in Eq. (6), (8)).

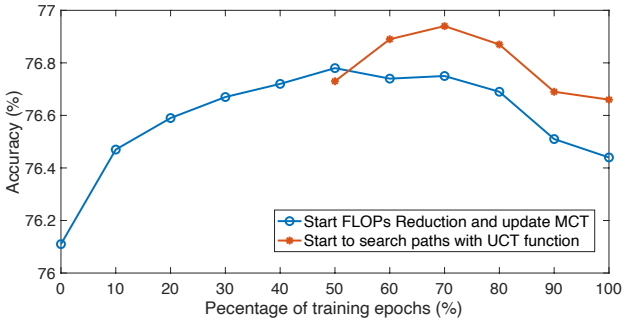


Figure 12. Effect of MCT-NAS in supernet training.

F.5. Effect of FLOPs reduction and updating MCT in training

To examine the effect of FLOPs reduction and updating MCT, we implement supernet training under different settings of the proposed strategies in training. The detailed results can be referred to as Table 8.

Table 8. The performance gain of each part in MCT-NAS with 330M FLOPs on MobileNet search space.

Training					Searching			Retraining	
Uniform Sampling	FLOPs Reduction	Update MCT in Training	UCT Search	Node Communication	Evolutionary Search	MCTS Search	Hierarchical Updates	Top-1	Top-5
✓					✓			75.94%	92.89%
✓						✓	✓	76.44%	93.15%
✓	✓					✓	✓	76.49%	93.19%
✓		✓				✓	✓	76.54%	93.23%
✓	✓	✓				✓	✓	76.65%	93.34%
✓	✓	✓				✓		76.21%	93.11%
✓	✓	✓	✓			✓		76.35%	93.17%
✓	✓	✓	✓	✓		✓		76.62%	93.32%
✓	✓	✓	✓	✓		✓	✓	76.94%	93.37%

In detail, we can see that with MCT sampler, FLOPs reduction (updating MCT in training) can lead to a 0.05%(0.1%) improvement on Top-1 accuracy (3-th and 4-th row in Table 8). Moreover, when use both strategies of FLOPs reduction and updating MCT in training (5-th row in Table 8), the performance of searched results can be boosted by 0.21%; Since with FLOPs Reduction, each sampled subnet is within the FLOPs budget, and thus can help to efficiently update MCT.

G. The Calculation of Search Cost in MCT-NAS

In this paper, we propose a hierarchical node selection method, which involves additional computation cost. For an accurate and fair comparison with other NAS methods, we calculate the total computed image number for the replacement of the search number. Many methods leverage the validation dataset [9, 35] (split from training) with 50k pictures to evaluate each path. However, in MCT-NAS, since we evaluate each path with a single batch of pictures, the number of pictures required for selecting a proper path is formulated as follows

$$\mathcal{N}_{path} = bs \times n_{thrd} \times \sum_{i=1}^{L} |\mathcal{O}_i| \tag{15}$$

where bs represents the batch size in search, and $|\mathcal{O}_i|$ indicates the number of candidate operations in layer i. Therefore, with $bs = 128$, $n_{thrd} = 6$, $L = 21$, and $|\mathcal{O}_i| = 13$, each path in MCT-NAS amounts to $5\times$ search cost compared to other search methods.

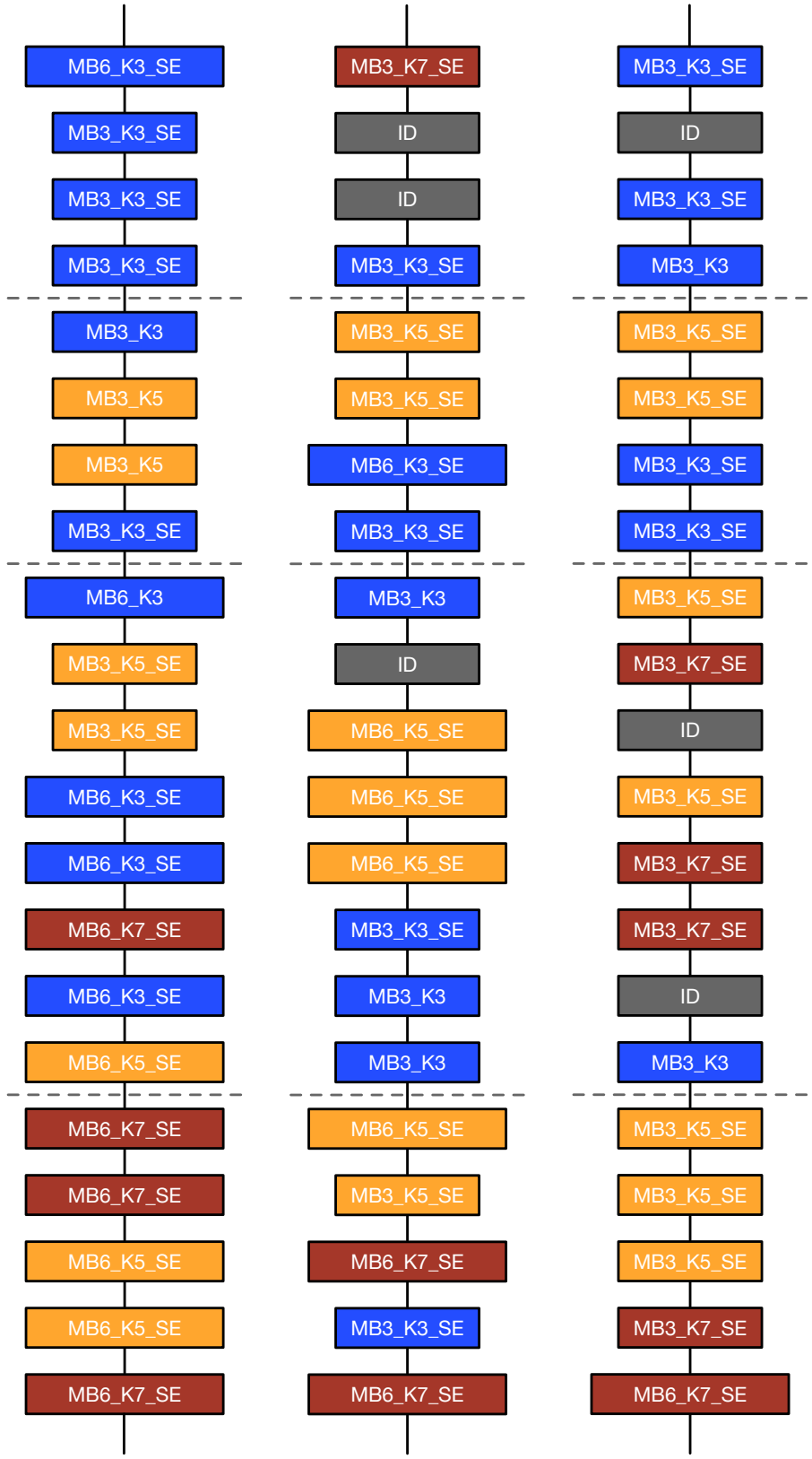
H. Visualization of Searched Architectures

We visualize the searched architectures by our MCTS-NAS method as Figure 13. In detail, we find that the searched optimal structures have the following three characteristics:

1. In the kernel level, the optimal structure generally tends to use more 5×5 or 7×7 convolutions at the layers close to the last layer, and the last layer is always with 7×7 kernel size.

2. In the width level, the last few operations of the searched structure are more likely to use a large expansion ratio, which is more evident with a larger FLOPs budget (*e.g.*, our 440M FLOPs structure uses an expansion ratio of 6 for the last 10 consecutive layers).

3. When the network budget is insufficient (*i.e.* 330M and 280M), the searched structure generally tends to use more ID operations at the layers close to the first layer to reduce FLOP consumption through fast downsampling.



(a) MCT-NAS-A (b) MCT-NAS-B (c) MCT-NAS-C

Figure 13. Visualization of architectures obtained by MCT-NAS in Table 2.