Appendix for CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS

Zixuan Zhou[13⋆], Xuefei Ning[12*], Yi Cai[1], Jiashu Han[1],
Yiping Deng[2], Yuhan Dong[3], Huazhong Yang[1], and Yu Wang[1†]

[1] Department of Electronic Engineering, Tsinghua University
[*]zhouzx21@mails.tsinghua.edu.cn, [*]foxdoraame@gmail.com,
[†]yu-wang@tsinghua.edu.cn
[2] Huawei TCS Lab
[3] Tsinghua Shenzhen International Graduate School

1 Additional Discussions on CLOSE and CLOSENet

1.1 Insights into the Improvements by Increasing Sharing Extent

Sec. 3.1 of the main paper shows that Supernet-2 (with larger sharing extent) significantly outperforms Supernet-1 (with vanilla sharing extent) in the early training stages. This observation is a bit counter-intuitive, since the previous studies have shown that a large sharing extent would aggravate the parameter coupling and multi-model forgetting phenomenon [1,8,5,3]. Therefore, it might be confusing where these improvements come from. To further understand our observation, we conduct a deeper inspection into the estimation results of the Supernet-1 and Supernet-2, and find out that there are two reasons underlying the observation.

First, a larger parameter sharing extent accelerates the training process of the supernet. This is due to the reduced number of the supernet's parameters. Previous studies reveal that a longer training can improve the ranking quality, yet brings extra computational costs [2,3]. In our case, the highly-shared supernet (Supernet-2) can achieve the same training performance under a smaller computational budget, thus improve the ranking quality.

Second, a large parameter sharing extent alleviates the well-known under-estimation phenomenon of larger architectures in vanilla one-shot estimations [2,3]. Following the visualization technique of a recent study [3], we divide the candidate architectures into five groups based on their complexities, and obtain the average ranking difference of architectures in each group. The ranking difference (RD) of an architecture is defined as the difference of its true ranking and its estimated ranking by supernets. A negative RD indicates the under-estimation of an architecture. As shown in Fig. A1, the average

⋆ Equal contribution.

RDs of the largest architectures in the top 20% (80% and 100% on the X-axis) in Supernet-2 are closer to zero than that in Supernet-1 on both NAS-Bench-201 and NAS-Bench-301. This indicates the alleviation of the under-estimation phenomenon of larger architectures.

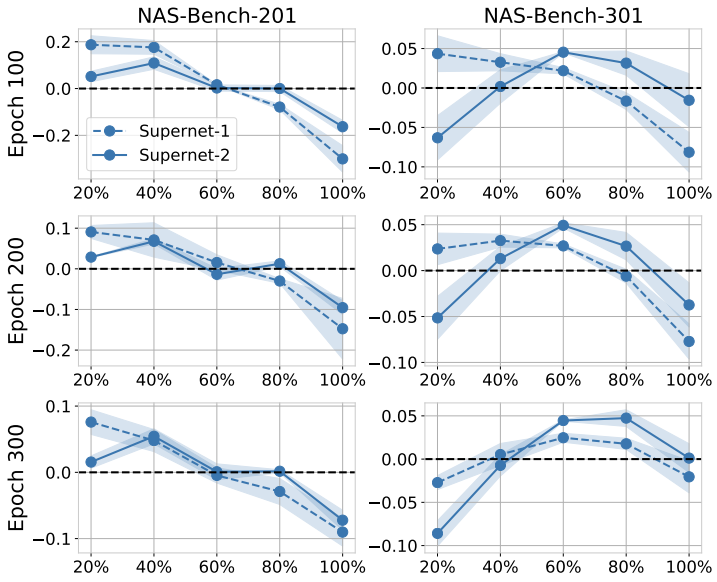


Fig. A1: Evaluation of the under-estimation phenomenon on NAS-Bench-201 and NAS-Bench-301. X-axis: Complexity groups. Y-axis: Average RD.

1.2 More Discussion on Using The GATE Module for The Dynamic Decision of Sharing Scheme

The "Strengths compared to Vanilla Supernets" section and Fig. 3 of the main paper have discussed how the dynamic decision of sharing scheme between architectures in CLOSENet can facilitate a more proper parameter sharing scheme between architectures. For example, recall that the two 1×1 convolutions in Fig. 3 (bottom) are equivalent in two isomorphic architectures despite having different positions. And the vanilla supernet uses different parameters for these two convolutions, while CLOSENet gives out a more proper sharing scheme to share the same GLOW block for them.

This strength comes from the dynamic decision design of the sharing scheme, and also partly from the GCN-based architecture embedder [4] adopted in our GATE module. As this architecture embedder conducts permutation-invariant

aggregations on the graph, it can naturally map the counterpart nodes in isomorphic architectures to the same node embeddings. Therefore, for the equivalent operations (i.e., the operations between the counterpart nodes) in two architectures, the MLP in the GATE module takes the same embeddings as input, and thus give out the same assignment.

The GCN-based architecture embedder adopted by CLOSENet, GATES [4], mimics the actual data processing to model the NN architecture. Corresponding to the computation flow of the cell-architecture in Sec. 3.2 of the main paper, the embedding of node j (denoted as E_j in Eq. 3 of the main paper) is defined as:

$$E_j = \sum_{i<j} \sigma(\text{OpEmb}(o^{(i,j)})W_o) \odot E_i W_x, \tag{1}$$

where σ is a sigmoid function, $\text{OpEmb}(o)$ gives out the embedding of this type of operation, and W_o and W_x denote two different linear transformation matrices. The embedding of the input node is randomly initialized.

1.3 CLOSENet in Non-topological Search Spaces

In non-topological search spaces (e.g., ResNet-like search space [6]), the computation blocks (i.e., operations) are put in sequential order, which is different from the topological structure in the generic search spaces. Therefore, we replace the normal GATE module with a simple but effective strategy that **assigns each GLOW block to some consecutive operations in an interval**. This assignment strategy comes from the intuition that the consecutive operations in the architectures have similar data processing functionality. Based on our analysis in Sec. 3.2 of the main paper, it is more reasonable to share the parameters of these operations. Since each operation has only one assigned block at the same time, the assignment intervals of the GLOW blocks are nonoverlapping.

When adding a new GLOW block, we propose to choose an existing block and divide its assignment interval down the middle. Then we assign the operations in one of the divided intervals to the new block. In this way, we can naturally apply the WIT to the new block to inherit the weights from the chosen one.

Fig. 5 in Sec. 4.1 of the main paper demonstrates that CLOSE consistently achieves higher ranking quality across the training process on the two non-topological search spaces (i.e., NDS ResNet and NDS ResNeXt-A).

1.4 Implementation of CLOSE

Alg. 1 shows the pipeline of using CLOSE to train CLOSENet. Specifically, we construct CLOSENet with one GLOW block at the beginning. Then, we gradually add blocks to reduce the sharing extent at the preset epochs. WIT is applied to initialize the parameters of the new block and MLP unit. In each training iteration, we randomly sample an architecture to update the parameters of CLOSENet, including GLOW blocks and the GATE module, through the Eq. 3 to Eq. 8 introduced in Sec. 3.2 of the main paper. SRT is applied to restart the learning rate when it becomes too small.

Algorithm 1 The Training Process of CLOSE on CLOSENet

Input:

D: Training data; T: Training epochs; A: Architecture search space;
S_{CL}: The set of switch points (epochs) of sharing extent
S_{LR}: The set of restart points (epochs) of learning rate

Training Process:

```
Construct a randomly initialized CLOSENet N_A with one GLOW block
for t = 1, ··· , T do
    if t ∈ S_CL then
        Adding a new GLOW block in N_A
        Using WIT to initialize the new block and GATE module
    end if
    if t ∈ S_LR then
        Using SRT to restart the learning rate and schedule
    end if
    for i = 1, ··· , I do
        Randomly sample an architecture a ∈ A
        Sample a batch of training data from D
        Update parameters in N_A with Eq. 3 ∼ Eq. 8
    end for
end for
```

Output: The well-trained CLOSENet N_A

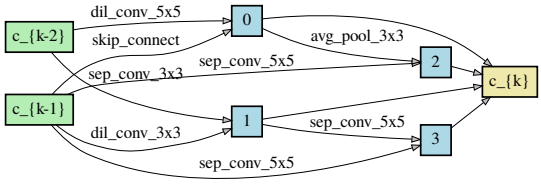
2 Detailed Configurations

2.1 Supernet Training

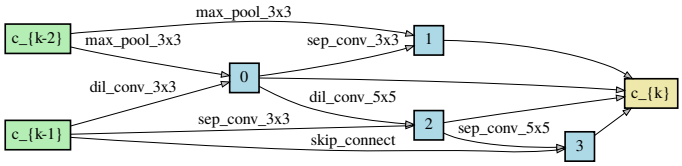
In our experiments, we use the same training configurations for vanilla one-shot supernets and CLOSENet. In detail, we train supernets via a SGD optimizer with momentum 0.9 and weight decay 5e-4 . The learning rate is set to 0.05 initially and decayed by 0.5 each time the supernet accuracy stops to increase for 30 epochs. In the training process, the dropout rate is set to 0.1, and the gradient norm is clipped to be less than 5. The batch size is set to 512. For each batch of examples, we randomly sample one architecture to update supernets' parameters. Besides, we set the training epochs of all supernets to 1000 epochs.

2.2 Architecture Search

We adopt CARS [7], an improved evolutionary approach, to search the optimal architectures in DARTS search space. The search process contains the supernet training stage and the architecture search stage. We set the total epochs to 400, and set the population size to 100. We first train the supernet for 100 epochs to warm up the parameters. Then, in the supernet training stage, we train the supernet for 5 epochs during one evolution iteration. In each mutation step during the architecture search stage, random mutation, random crossover and random sampling are conducted with a probability of 0.25, 0.25, and 0.5, respectively, following CARS [7]. Fig. A2 shows the discovered architectures.



(a) Normal Cell



(b) Reduction Cell

Fig. A2: The discovered cell architectures by CLOSE.

2.3 Training of the Discovered Architectures

On CIFAR-10, we stack the discovered architectures 20 times to construct the network, and set its initial channel number to 36. The network is trained for 600 epochs with batch size 128. We use a SGD optimizer with momentum 0.9 and weight decay 3e-4. The learning rate is decayed from 0.05 to 0.001 following a cosine schedule. The dropout rate is set to 0.1, and the gradient norm is clipped to be less than 5. Besides, the cutout augmentation with length 16, the path dropout of probability 0.2 and the auxiliary towers with weight 0.4 are used.

When transferring the discovered architectures to ImageNet, we stack 14 cells to construct the network, and set its initial channel number to 48. The network is trained for 300 epochs with batch size 256. The weight decay is set to 3e-5, and the learning rate is decayed from 0.1 to 0 following a cosine schedule. The path dropout technique is not used.

3 Additional Experiments

3.1 Investigation of The GATE Module

In this section, we conduct an experiment to further investigate the effectiveness and robustness of the GATE module in CLOSENet. Specifically, we randomly sample four pairs of architectures on NAS-Bench-301, where two of them have a big structure difference (labeled as 1 and 2), and the other two pairs (labeled as

3 and 4) have a similar structure. Fig. A3 shows the four pairs of architectures. We use a well-trained GATE module to obtain their assignment distribution following the Eq. 4 of the main paper. Then we calculate the KL divergence of the distribution between all the 3×3 convolutions in each pair of architectures and make a summation, which can reflect their assignment similarity. The results of these four pairs are 4.9847 ± 0.0121, 3.2797 ± 0.0385, 0.4680 ± 0.0091 and 0.0003 ± 0.0001. The stability of the KL divergence across different random seeds shows the robustness of GATE. Meanwhile, the larger KL divergence of pair 1 and 2 also demonstrates the GATE module can give out a proper assignment of blocks.

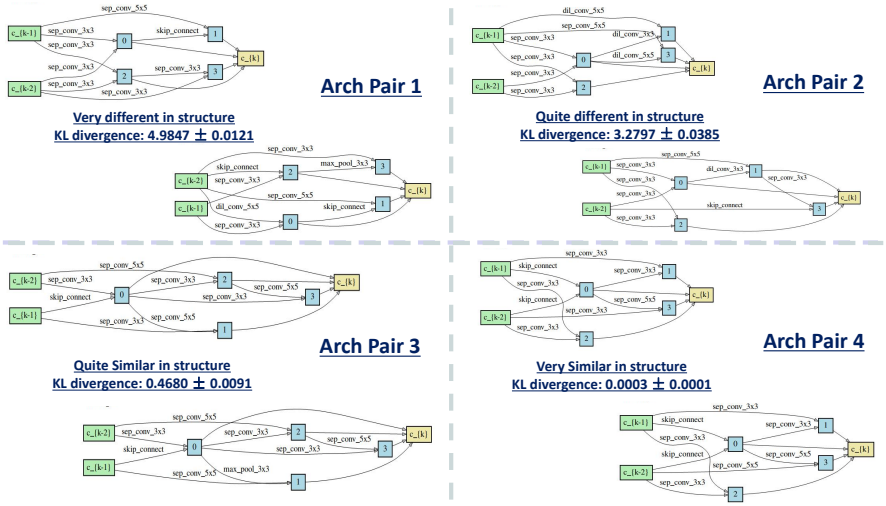


Fig. A3: The four pairs of the architectures we sample to investigate the GATE module on NAS-Bench-301. We demonstrate the normal cells of the architectures here. Architectures in Arch Pair 1 and Arch Pair 2 are different in structure, while architectures in Arch Pair 3 and Arch Pair 4 are similar.

3.2 Effectiveness of The WIT Technique

The results shown in Sec. 4.3 of the main paper demonstrate that WIT plays an important role in CLOSE. Here we provide a visualization to reveal its necessity more clearly. Fig. A4 shows that right after the curriculum (i.e., sharing extent) switch, randomly initializing the parameters of the new GLOW block and MLP unit significantly damages the ranking quality. On the contrary, WIT helps CLOSE to retain the high ranking quality.

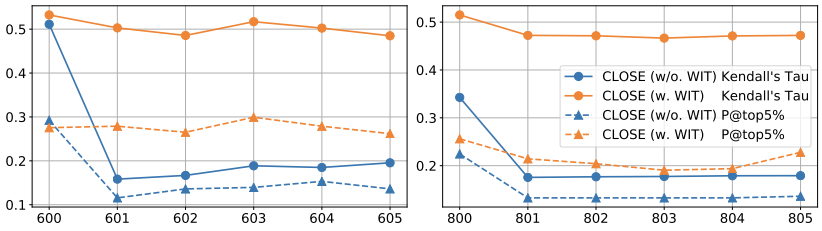


Fig. A4: The trend of the ranking quality right after switching the sharing extent (at 600 and 800 epoch) on NAS-Bench-301. X-axis: Training epochs. Y-axis: Ranking quality (Kendall's Tau or P@top5%).

3.3 A Simplified Version of CLOSE: CLOSE-S

In this section, we provide a simplified version of CLOSE (namely CLOSE-S). Based on the analysis in Sec. 3.1 of the main paper, we simply use the two sharing extents of Supernet-1 and Supernet-2 in two stages of the training process. Specifically, in the first stage (0 to 400 epoch), CLOSENet shares only one copy of parameters on all the edges in each cell-architecture. While in the second stage (400 to 1000 epoch), CLOSENet enables the operations on different edges to share different copies of parameters. The WIT and SRT are also adopted when switching the sharing scheme and extent at 400 epoch. In this way, we can easily apply CLOSE-S to train CLOSENet without the help of the GATE module, since the assignments of GLOW blocks are preset (but different) in these two stages.

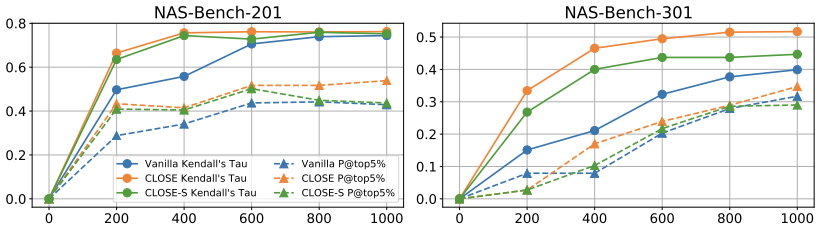


Fig. A5: Comparison of the vanilla one-shot supernet, CLOSE and CLOSE-S. X-axis: Training epochs. Y-axis: Ranking quality (Kendall's Tau or P@top5%).

As shown in Fig. A5, CLOSE-S achieves a higher KD and P@top5% than the vanilla supernet on two generic search spaces across the training process. Although it cannot reach the performances of CLOSE, CLOSE-S is easier to implement, thus can be adopted when the performance demand is not as strict.

References

1. Benyahia, Y., Yu, K., Smires, K.B., Jaggi, M., Davison, A.C., Salzmann, M., Musat, C.: Overcoming multi-model forgetting. In: International Conference on Machine Learning (ICML). pp. 594–603. PMLR (2019) 1
2. Luo, R., Qin, T., Chen, E.: Understanding and improving one-shot neural architecture optimization. CoRR **abs/1909.10815** (2019) 1
3. Ning, X., Tang, C., Li, W., Zhou, Z., Liang, S., Yang, H., Wang, Y.: Evaluating efficient performance estimators of neural architectures. In: Annual Conference on Neural Information Processing Systems (NIPS) (2021) 1
4. Ning, X., Zheng, Y., Zhao, T., Wang, Y., Yang, H.: A generic graph-based neural architecture encoding scheme for predictor-based nas. In: European Conference on Computer Vision (ECCV). pp. 189–204. Springer (2020) 2, 3
5. Niu, S., Wu, J., Zhang, Y., Guo, Y., Zhao, P., Huang, J., Tan, M.: Disturbance-immune weight sharing for neural architecture search. Neural Networks **144**, 553–564 (2021) 1
6. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10428–10436 (2020) 3
7. Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: Cars: Continuous evolution for efficient neural architecture search. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1829–1838 (2020) 4
8. Zhang, M., Li, H., Pan, S., Chang, X., Su, S.: Overcoming multi-model forgetting in one-shot nas with diversity maximization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7809–7818 (2020) 1