Self-Supervised Representation Learning for Evolutionary Neural Architecture Search – Supplementary Material

Chen Wei, Yiping Tang, Chuang Niu, Haihong Hu, Yue Wang, Jimin Liang

I. The Operation Position Analyze of Path-based Encoding Scheme

Figure S1 shows two neural network architectures from the NASBench-101 search space, and the mean percentage test accuracy of the two neural architectures are 91.2% (Figure S1a) and 90.4% (Figure S1b), respectively.

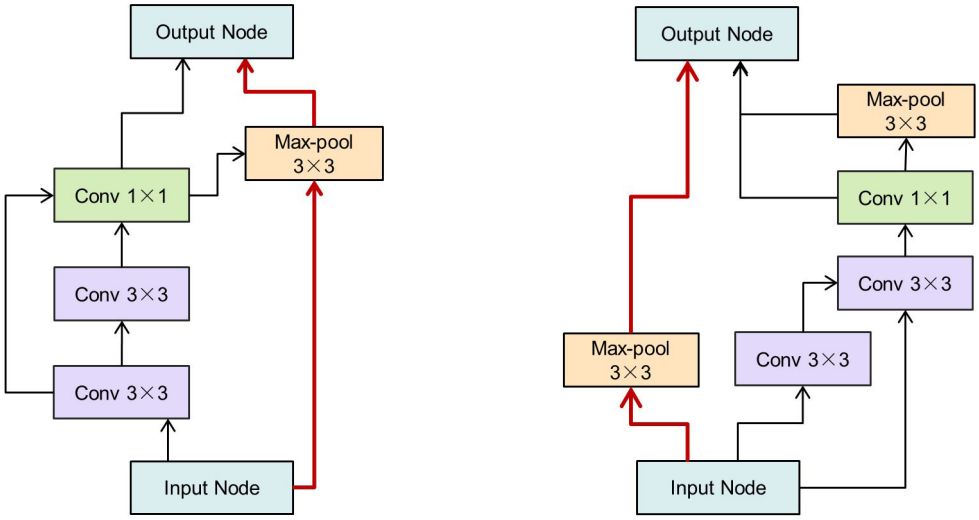

FIGURE S1: Two different neural architectures from NASBench-101.

The two neural architectures in Figure S1 have the same path-based encoding, as shown in Figure S2. The red line paths in Figure S1a and Figure S1b indicate the same input-to-output path that only contains a max-pooling 3×3 operation. Although the red line paths in the two neural architectures are identical, the position of the max-pooling 3×3 operation are different. In Figure S1a, the input of the max-pooling 3×3 operation is the input node and the convolution 1×1 operation, while in Figure S1b the input of the max-pooling 3×3 operation is the input node. The ignorance of the position of operations in the neural architecture causes the path-based encoding method to map the two different neural architectures in Figure S1 into the same encoding vector.

II. Position-aware Path-based Encoding with more architecture properties

To demonstrate the flexibility of the proposed position-aware path-based encoding scheme, we extend it to include the input resolution and kernel numbers of neural architectures. The new encoding scheme is illustrated in Figure S3, taking the NASBench-101 as an example. The NASBench-101 contains three types of operation nodes (convolution 3×3, convolution 1×1, and max-pooling 3×3), which are encoded by three-dimensional one-hot vectors, as described previously. Four different kernel values (256, 512, 1024, and 2048) and four input resolutions (32, 64, 128, and 256) are considered. Each property is encoded with a four-dimensional one-hot encoding vector. Therefore, an operation node can be represented by a seven-dimensional vector, where the first three dimensions represent the operation type and the last four dimensions represent the number of kernels. If an operation node does not contain the kernel number property, a four-dimensional all-zero vector is used to fill the operation property. An input-to-output path is encoded in a similar way to the original position-aware path-based encoding scheme. The final encoding of the neural architecture is formed by concatenating the vector of input resolution encoding and that of all the input-to-output paths.

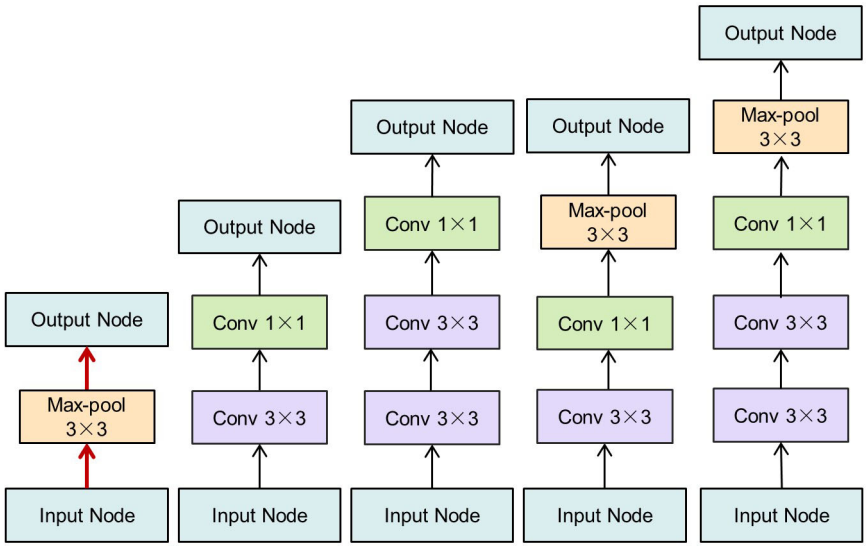

FIGURE S2: Input-to-output paths of the two neural architectures in Figure S1.

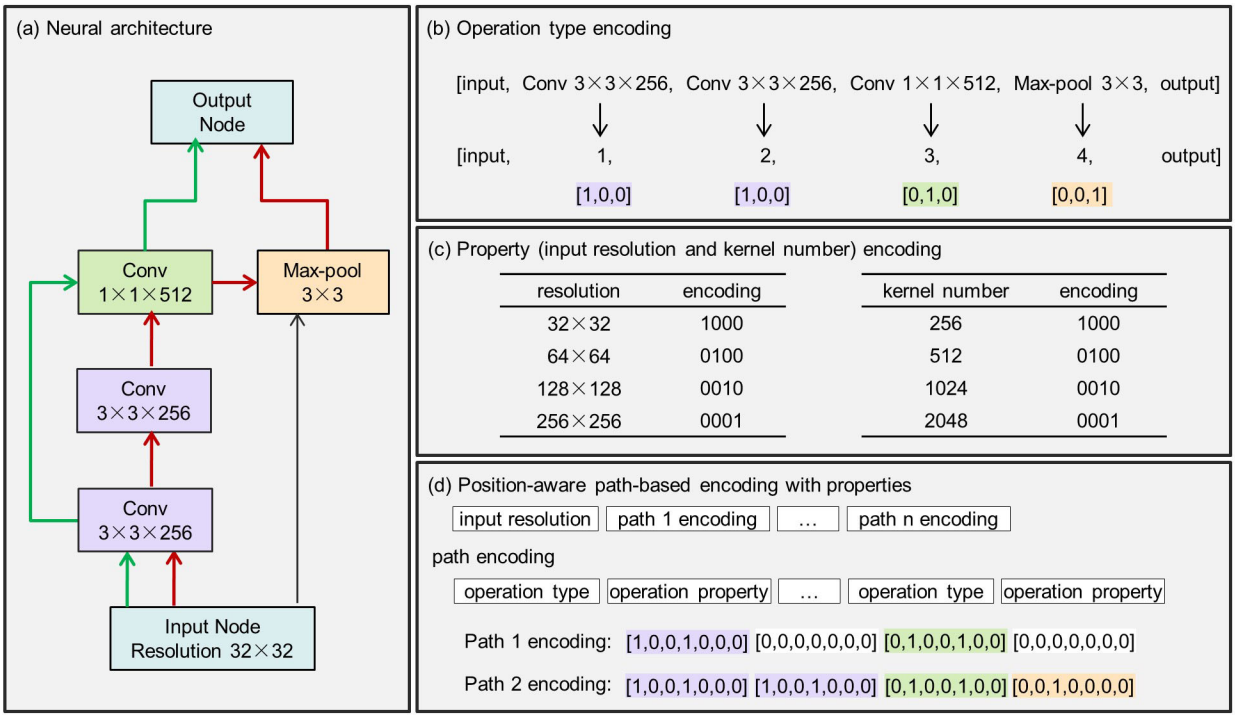

FIGURE S3: Overview of the position-aware path-based encoding with more architecture properties. (a) A neural architecture in the NASBench-101 search space, to which we explicitly add attributes such as input resolution and number of kernels. The green and red lines indicate the two input-to-output paths. (b) Unique indices and operation type encodings for operations in the neural architecture. (c) Encoding scheme of input resolution and kernel number. (d) Position-aware path-based encoding with properties is composed of input resolution encoding followed by several operation node encodings.

III. An Illustration of the Central Contrastive Learning

As illustrated in Figure S4, the objective of the central contrastive learning is to aggregate the positive green features to the center vector $\mathbf{e}_c$ and push the negative orange features far away from the center. The central vector $\mathbf{e}_c$ is calculated by averaging all the green feature vectors and can be formulated as

$$\mathbf{e}_c = \frac{1}{3}\sum_{i=1}^{3}\mathbf{e}_{pi}. \tag{1}$$

The center contrastive loss corresponding l is defined as

$$l = l_c + 0.5 \times l_{reg}, \tag{2}$$

where l_c and l_{reg} are defined in Eq. 3 and Eq. 4, respectively.

$$l_c = \sum_{i=0}^{3} -\log \frac{\exp(sim(\mathbf{e}_{pi}, \mathbf{e}_c))}{\exp(sim(\mathbf{e}_{pi}, \mathbf{e}_c)) + \sum_{k=1}^{5} \exp(sim(\mathbf{e}_{nk}, \mathbf{e}_c))}. \tag{3}$$

$$l_{reg} = \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \mathbb{1}_{[j \neq i]} \mathbf{e}_{pi}^{\top} \mathbf{e}_{pj}. \tag{4}$$

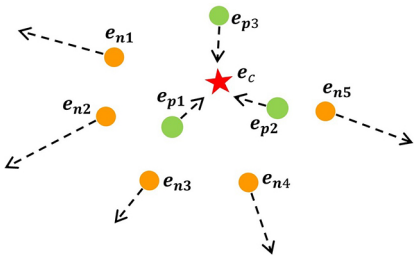

FIGURE S4: An illustration of the central contrastive learning.

IV. More Experiments About Fixed Budget NPENAS

The comparison of NPENAS-NP and NPENAS-NP-FIXED, NPENAS-SSRL and NPENAS-SSRL-FIXED, and NPENAS-SSCCL and NPENAS-SSCCL-FIXED on the NASBench-101 search spaces are illustrated in Figure S5. The pairwise comparison plots are attached with error bars to reflect the stability of algorithms, and the error bars represent the 30% and 70% percentile range.

As shown in Figure S5, the performance of NPENAS-SSRL and NPENAS-SSCCL is comparable with their corresponding fixed budget version, but NPENAS-NP-FIXED decreases significantly compared to NPENAS-NP. By comparing the error bars in the plots, NPENAS-SSCCL and NPENAS-SSCCL-FIXED are more stable than the other algorithms.

Figure S6-S8 show the mean performance of each algorithm on NASBench-201 on CIFAR10, CIFAR-100, and ImageNet-16-120. On this relatively small search space, NPENAS-NP-FIXED, NPENAS-SSRL-FIXED, and NPENAS-SSCCL-FIXED perform comparably to their full-budget versions on CIFAR-10 and ImageNet-16-120. On CIFAR-100, the algorithms' fixed budget versions perform slightly better than their full budget versions. Figure S6 and Figure S7 also show that the 30% and 70% percentile ranges converge to a single line when the search budget is greater than 60, which indicates that all the algorithms are fairly stable.

The above results illustrate that self-supervised pre-training can significantly reduce the number of training neural architectures of neural predictors and confirm the necessity of self-supervised pre-training for improving the performance of NAS.

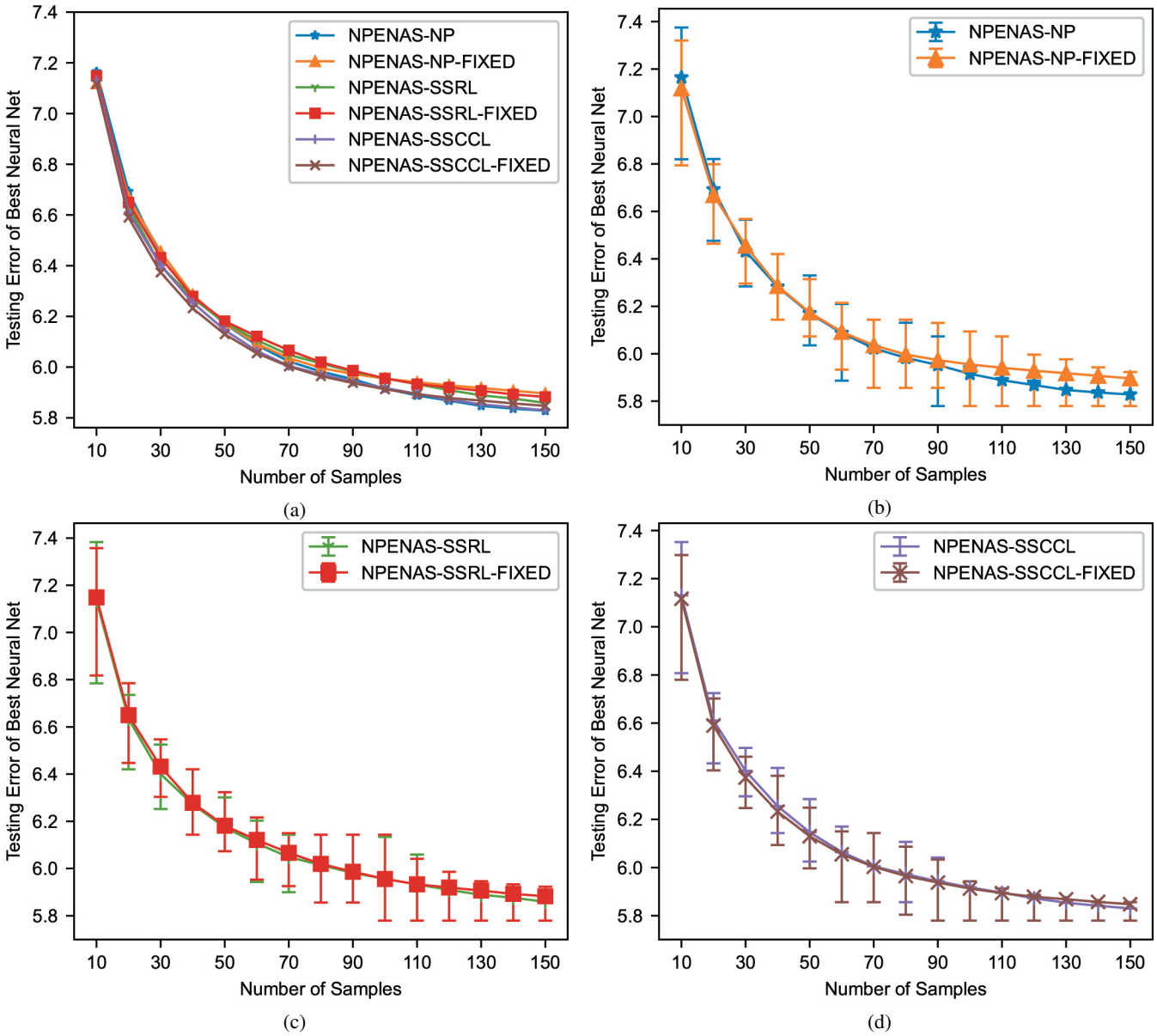

FIGURE S5: Comparison of the fixed budget NPENAS algorithms on NASBench-101. (a) The mean plots of algorithms under different search budgets. (b) The mean plots of NPENAS-NP and NPENAS-NP-FIXED with error bars. (c) The mean plots of NPENAS-SSRL and NPENAS-NP-SSRL with error bars. (d) The mean plots of NPENAS-SSCCL and NPENAS-SSCCL-FIXED with error bars.

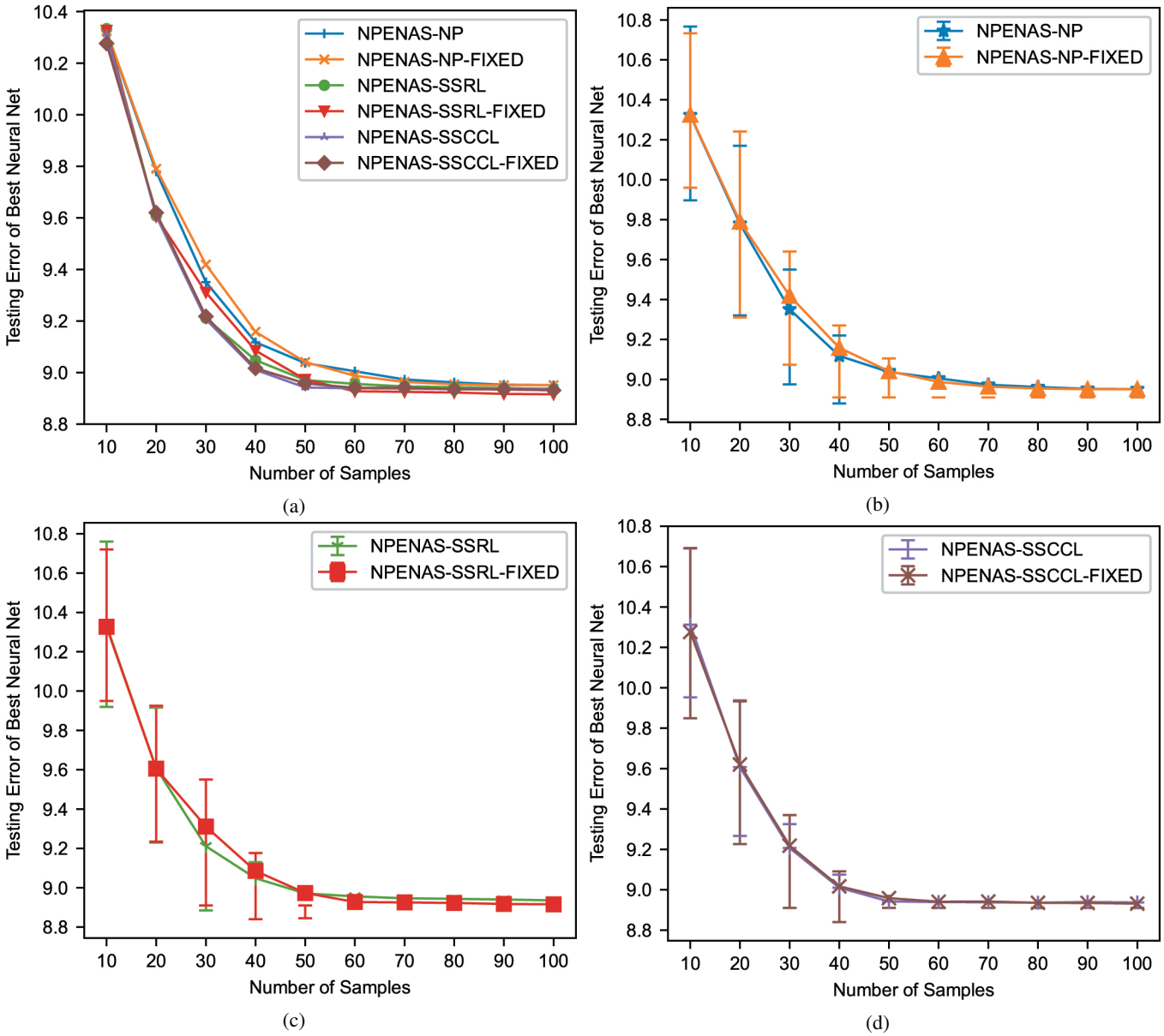

FIGURE S6: Comparison of the fixed budget NPENAS algorithms on NASBench-201 on CIFAR-10. (a) The mean plots of algorithms under different search budgets. (b) The mean plots of NPENAS-NP and NPENAS-NP-FIXED with error bars. (c) The mean plots of NPENAS-SSRL and NPENAS-NP-SSRL with error bars. (d) The mean plots of NPENAS-SSCCL and NPENAS-SSCCL-FIXED with error bars.

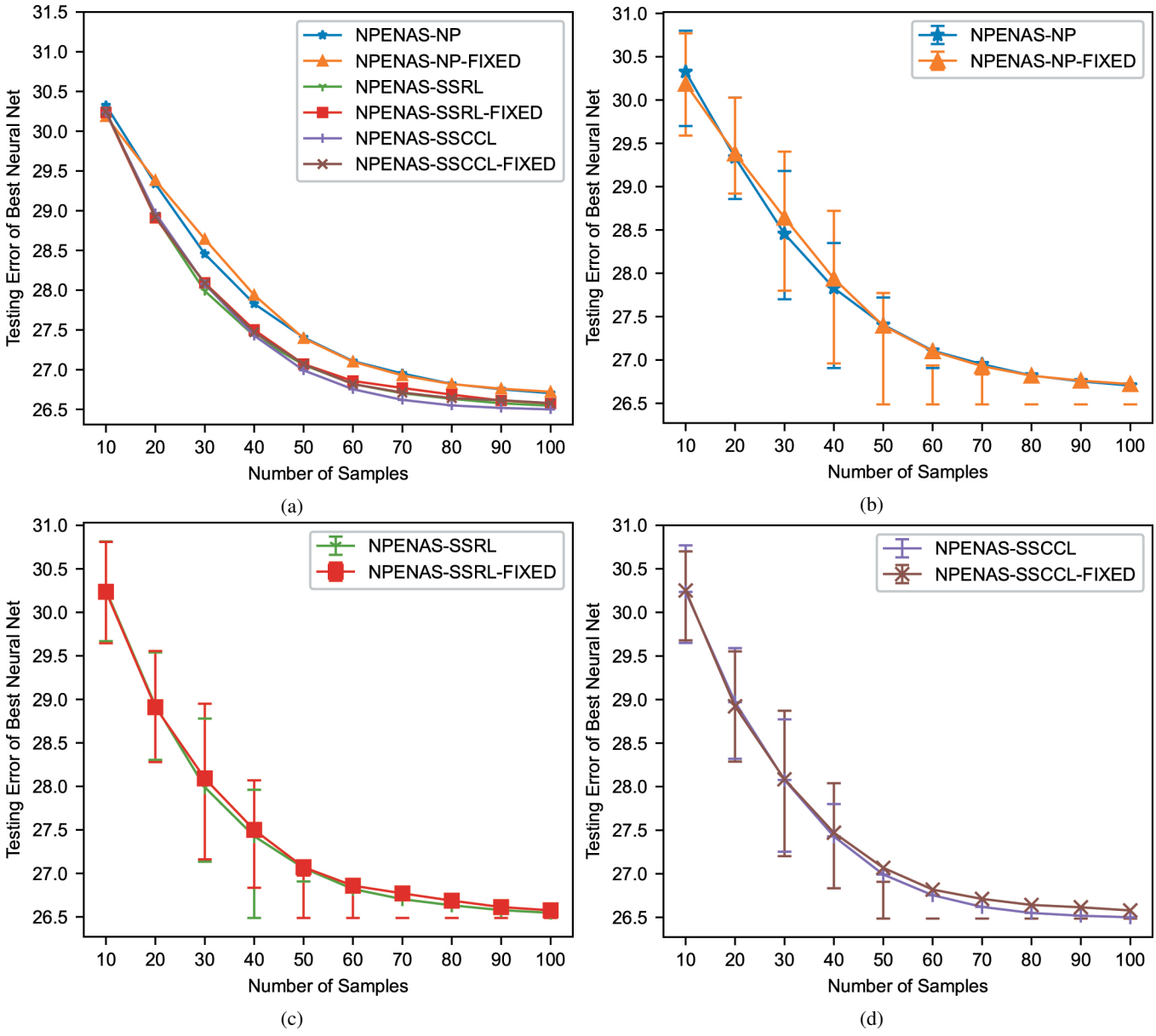

FIGURE S7: Comparison of the fixed budget NPENAS algorithms on NASBench-201 on CIFAR100. (a) The mean plots of algorithms under different search budgets. (b) The mean plots of NPENAS-NP and NPENAS-NP-FIXED with error bars. (c) The mean plots of NPENAS-SSRL and NPENAS-NP-SSRL with error bars. (d) The mean plots of NPENAS-SSCCL and NPENAS-SSCCL-FIXED with error bars.

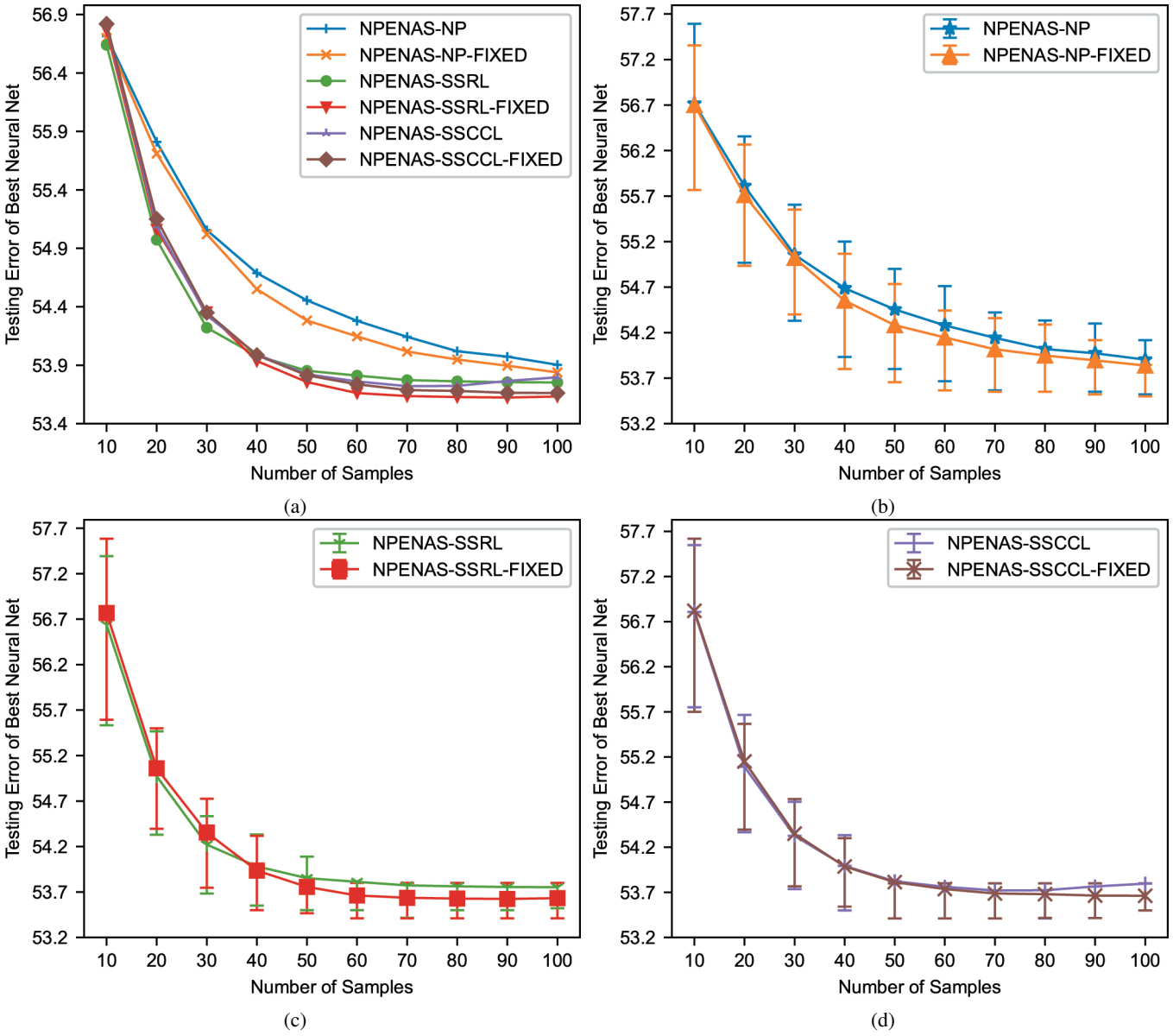

FIGURE S8: Comparison of the fixed budget NPENAS algorithms on the NASBench-201 on ImageNet-16-120. (a) The mean plots of algorithms under different search budgets. (b) The mean plots of NPENAS-NP and NPENAS-NP-FIXED with error bars. (c) The mean plots of NPENAS-SSRL and NPENAS-NP-SSRL with error bars. (d) The mean plots of NPENAS-SSCCL and NPENAS-SSCCL-FIXED with error bars.