Supplementary Materials for NAR-Former: Neural Architecture Representation Learning towards Holistic Attributes Prediction

1. Encoding for Other Information

Following the approach to encode operation types and topology information, other information necessary for downstream tasks can be similarly further encoded. Then each token is generated by concatenating the encoding of each part.

In the experiments on NNLQP [6], the architectures are whole deep neural networks(*e.g*. EfficientNet [11]), some of which may have more than 200 layers. In order to learn more complete and detailed representations of these large networks, in addition to the operation type and topology structure, we also encode the attribute parameters and the output tensor shape of the each operation. The specific contents of these two types of information are shown in Tab. 1. The operation attributes here refer to the parameters specified when defining an operation layer, such as kernel size, number of groups, and so on. Once the shape of the input data is determined, the output shape of each layer can be inferred. The output shape has a maximum of 4 parameters. However, after some layers (such as the flatted layer), it may only have 2 values, that is, the output is a batch of vectors rather than feature maps. For each node(*i.e*. layer), we use our real number tokenizer to map the node's information (operation type, self position, source nodes position, operation attributes, output shape) to vectors of length 32. The code of the node, which has a length of 480, is obtained by concatenating these vectors. When a layer node does not have some attributes or its output has less than 4 values, we pad zeros into its code to ensure uniform encoding lengths for all nodes. Considering that static properties (batch size, FLOPs, parameters, and memory access) have an impact on inference time, they are used for latency prediction together with the output of multi-stage fusion transformer.

Type	Content
Operation Attributes	Kernel Shape
	Stride
	Padding
	Dilation
	Groups
	Bias
	Perm
	Output Padding
Output shape	Batch Size
	Channel
	Height
	Width

Table 1. Details of other encoded information.

2. Implementation details

2.1. Setup on NAS-Bench-101

On NAS-Bench-101 [13], the setup of all experiments goes as follows. We employ AdamW [8] optimizer with default momentum parameters to train our NAR-Former. After the first 10% iteration warm-up, the learning rate linearly decays from initial value 0.0001 to 0. The batch size and number of epochs are set to 256 and 6000, respectively. For layer normalization and biases in all layers, weight decay is set to 0 and for other regular layers is 0.01. The dropout rate is 0.1. Inspired by ConvNeXt [7], we employ the Exponential Moving Average(EMA) [9] to alleviates overfitting.

NAS-Bench-101 provides the performance of each architecture on CIFAR-10 [5]. For the calculation of Kendall's Tau [10], the accuracy of each input architecture on the validation set is used as ground truth during training and validation, while the accuracy of each input architecture on the test set is used as ground truth during testing. We calculate the metrics on the test set respectively using the model of the last epoch, the best model, and the best EMA model on the validation set, and the highest one is reported.

2.2. Setup on NAS-Bench-201

On NAS-Bench-201 [4], the setup of all experiments goes as follows. AdamW [8] optimizer with default momentum parameters is used to train our NAR-Former. The learning rate follows a schedule that decays linearly to 0. The initial learning rate is 0.0001 reached by warming up with 10% iteration steps. The batch size and number of epochs are set to 256 and 4000, respectively. The weight decay is set to 0 for layer normalization and bias of all layers and 0.01 for other regular layers. The dropout rate is 0.1. We also use the Exponential Moving Average(EMA) [9] in experiments in this part.

NAS-Bench-201 contains the performance of each architecture on three different datasets: CIFAR-10, CIFAR-100 [5], and ImageNet-16-20 [2]. We only use the results on CIFAR-10 in our experiments. We follow the way in Sec. 2.1 to assign ground truth and report results.

2.3. Setup of Neural Architecture Search on DARTS

In this experiment, the structure of NAR-Former is the same as that used in experiments on NAS-Bench-101. In each round of evolution, the optimizer, weight decay strategy, dropout rate, and the change rules of the learning rate for predictor training are the same as those in Sec. 2.1, but the initial learning rate, batch size and the number of epochs are fixed to 0.001, 32, and 1000, respectively. The trained predictor was used to select the 10 most accurate architectures from the mutations.

We evaluate the searched architecture following [12]. The architecture to be evaluated is constructed using the searched normal cell and reduction cell and has 20 cells and 36 initial channels. This architecture then is trained from scratch on CIFAR-10 using a momentum SGD optimizer. The number of epochs is 600, and the batch size is set to 96. Additional training enhancements including cutout [3], drop path, and auxiliary towers are used.

2.4. Setup of Neural Architecture Search on MobileNet Space

We use the same search procedures and settings in OFA [1] to conduct the experiment of searching neural architectures on MobileNet space. The only difference is that we predict the accuracy of architecture using our NAR-Former, rather than the predictor adopted by OFA [1]. The architecture encoding is obtained through the tokenizer introduced in Sec. 3.2 of the paper, which has been slightly modified based on the MobileNet space. For each node token of the input architecture, we only encode the kernel size, width expansion ratio, self position of the current block. Instead of encoding the depth of the input architecture, we encode the size of the input image following the way of generating depth token. The structure of representation learning and attribute prediction is consistent with the NAR-Former we introduced in the paper.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[2] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 2

[3] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[4] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. 1

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 1, 2

[6] Liang Liu, Mingzhu Shen, Ruihao Gong, Fengwei Yu, and Hailong Yang. Nnlqp: A multi-platform neural network latency query and prediction system with an evolving database. In *51 International Conference on Parallel Processing - ICPP*, ICPP '22. Association for Computing Machinery, 2022. 1

[7] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[9] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 1, 2

[10] Pranab Kumar Sen. Estimates of the regression coefficient based on kendall's tau. *Journal of the American statistical association*, 63(324):1379–1389, 1968. 1

[11] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[12] Chen Wei, Chuang Niu, Yiping Tang, Yue Wang, Haihong Hu, and Jimin Liang. Npenas: Neural predictor guided evolution for neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[13] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019. 1