NAT: Neural Architecture Transformer for Accurate and Compact Architectures

Yong Guo*, Yin Zheng*, Mingkui Tan*†, Qi Chen,
Jian Chen†, Peilin Zhao, Junzhou Huang
South China University of Technology, Weixin Group, Tencent,
Tencent AI Lab, University of Texas at Arlington
{guo.yong, sechenqi}@mail.scut.edu.cn, {mingkuitan, ellachen}@scut.edu.cn,
{yinzheng, masonzhao}@tencent.com, jzhuang@uta.edu

In the supplementary, we provide the derivation of the objective function, the architecture representation method, more implementation details, and more comparison results of our paper [5]. For clarity, we organize our supplementary as follows. First, we give the derivation of our proposed objective function in Section 1. Second, we depict the details about architecture representation method of our NAT in Section 2. Third, we provide more implementation details about our model architecture in Section 3. Fourth, we report the quantitative results and corresponding visualizations w.r.t. the randomly sampled architecture and the effect of different graph representations in Section 4.

1 The Derivation of Objective Function

The objective function of NAT can be formulated as

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\beta\sim p(\cdot)}\left[\mathbb{E}_{\alpha\sim\pi(\cdot|\beta;\theta)}\left[R\left(\alpha,w\right)-R\left(\beta,w\right)\right]+\lambda H\big(\pi(\cdot|\beta;\theta)\big)\right] \\ &= \sum_{\beta} p(\beta)\left[\sum_{\alpha}\pi(\alpha|\beta;\theta)\big(R\left(\alpha,w\right)-R\left(\beta,w\right)\big)+\lambda H\big(\pi(\cdot|\beta;\theta)\big)\right]. \end{aligned} \tag{1}$$

The gradient $\nabla_\theta J(\theta)$ of the objective function w.r.t. θ is given by

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{\beta} p(\beta)\left[\sum_{\alpha}\nabla_\theta\pi(\alpha|\beta;\theta)\big(R\left(\alpha,w\right)-R\left(\beta,w\right)\big)+\lambda\nabla_\theta H\big(\pi(\cdot|\beta;\theta)\big)\right] \\ &= \sum_{\beta} p(\beta)\left[\sum_{\alpha}\pi(\alpha|\beta;\theta)\nabla_\theta\log\pi(\alpha|\beta;\theta)\big(R\left(\alpha,w\right)-R\left(\beta,w\right)\big)+\lambda\nabla_\theta H\big(\pi(\cdot|\beta;\theta)\big)\right] \\ &= \mathbb{E}_{\beta\sim p(\cdot)}\left[\mathbb{E}_{\alpha\sim\pi(\cdot|\beta;\theta)}\left[\nabla_\theta\log\pi(\alpha|\beta;\theta)\big(R\left(\alpha,w\right)-R\left(\beta,w\right)\big)\right]+\lambda\nabla_\theta H\big(\pi(\cdot|\beta;\theta)\big)\right] \\ &\approx \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\nabla_\theta\log\pi(\alpha_j|\beta_i;\theta)\big(R\left(\alpha_j,w\right)-R\left(\beta_i,w\right)\big)+\lambda\nabla_\theta H\big(\pi(\cdot|\beta_i;\theta)\big)\right]. \end{aligned} \tag{2}$$

2 Architecture Representation Methods of NAT

Deep neural networks (DNNs) have achieved remarkable success in many challenging tasks, including image classification [4, 6, 14, 15], face recognition [12, 13], brain signal processing [9, 10] many

*Authors contributed equally.
†Corresponding author.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

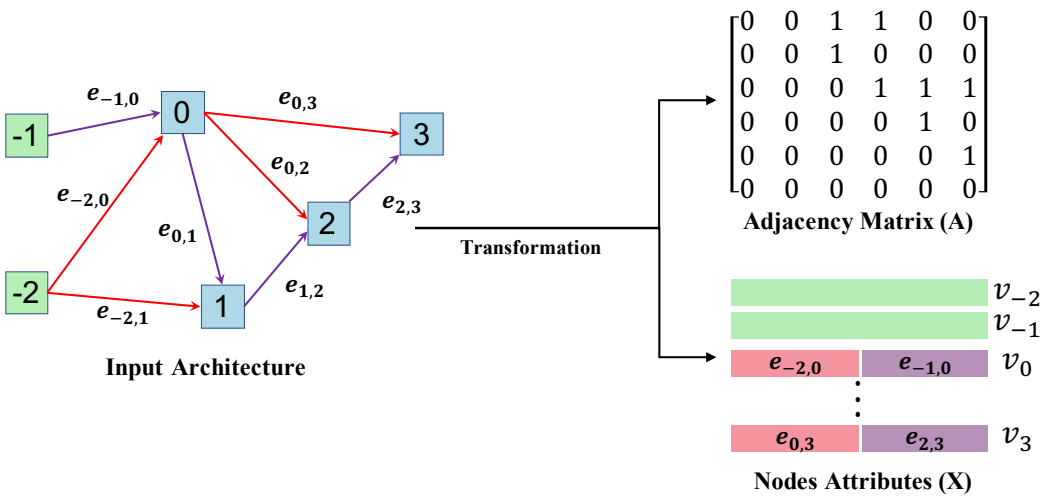

Figure 1: The architecture representation method of the proposed NAT. We define the operation connected to the input node with a smaller index as the first operation (red lines and blocks), and the other one as the second operation (purple lines and blocks).

Architectures	Original	NAT
#Params (M)	6.40±2.04	4.67±1.36
#MAdds (G)	1.07±0.32	0.79±0.21
Test accuracy (%)	95.83±1.08	**96.56±0.47**

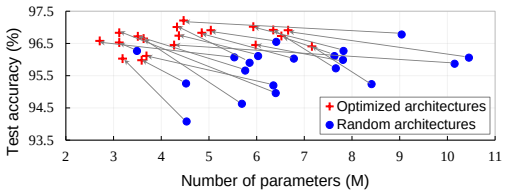

Figure 2: Effect of NAT on the average performance over 20 randomly sampled architectures on CIFAR-10 in terms of the number of parameters, the number of MAdds and test accuracy.

other areas [7, 1, 3, 2]. According to [16, 11, 8], we can represent an architecture α as a directed acyclic graph (DAG), *i.e.*, $\alpha = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes that denote the feature maps in DNNs and $\mathcal{E}$ is an edge set. To better exploit the adjacency information of operations in an architecture, we propose to exploit graph convolution network (GCN) to build the architecture optimization model.

In this paper, we develop a simple yet effective method to reconstruct node attributes as the combination of edge attributes. In the architecture graph, the intermediate nodes vary from each other due to the differences of their two input edges. We define the edge from the node with a small index as the first operation and the other one as the second operation. For any intermediate node v_k, we use the attributes of two input edges to uniquely represent it in the graph, *i.e.*, $v_k := (e_{ik}, e_{jk}), i \leq j$, where e_{ik} and e_{jk} are the first and the second operation respectively. To incorporate the information of the two input nodes, *i.e.*, v_{-2} and v_{-1}, we construct attribute vectors of each node with twice the length as that of an edge. Since all the architectures follow the same connection pattern in the output node, similar to ENAS [11] and DARTS [8], we omit the output node in our representation method. By combining the adjacency matrix and the representation of each node, the proposed representation method can uniquely identify a specific architecture.

3 More Implementation Details

The existing architectures can be classified into two categories, namely (i) loose-end architectures [11] and (ii) fully-concat architectures [8]. The loose-end architectures average all the nodes that are not selected as inputs to any other nodes and take the average value as the output (*e.g.*, ENAS, VGG and ResNet). The fully-concat architectures average all the intermediate nodes (*e.g.*, DARTS). One architecture transformer cannot be adapted to these two kinds architectures simultaneously. Thus, in

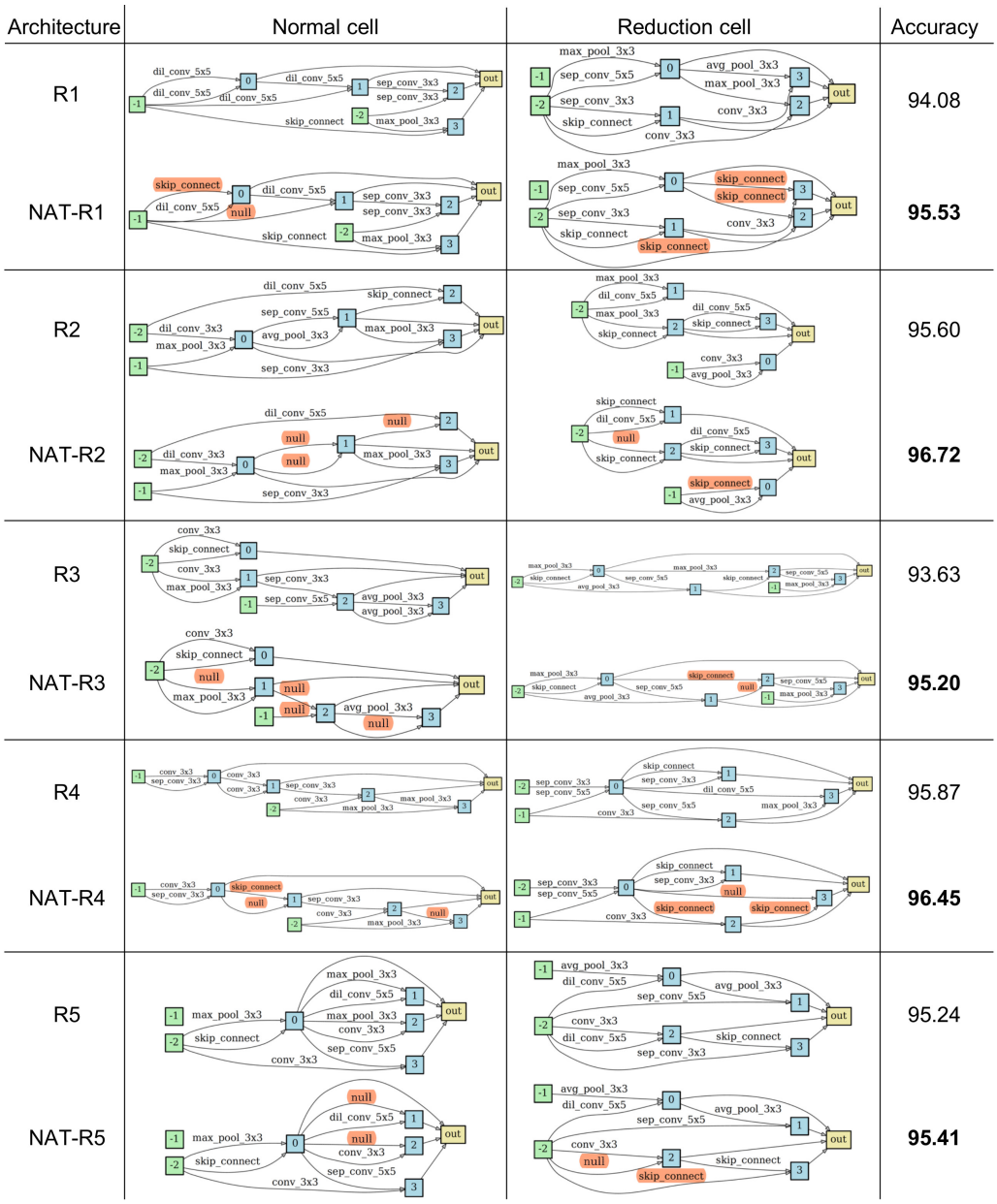

Figure 3: Architecture optimization results of several randomly sampled architectures on CIFAR-10.

this paper, we train two architecture transformers for the loose-end architectures and the fully-concat architectures separately. Furthermore, to cover most possible architectures and learn a general policy for NAT, we consider the original architectures with an operation set consisting of 9 very common operations, including identity, convolution with the kernel size 1×1 and 3×3, separable convolution with the kernel size 3×3 and 5×5, dilated separable convolution with the kernel size 3×3 and 5×5, max pooling, and average pooling. During training, we build the deep network by stacking 8 basic cells and train the transformer for 100 epochs. with the batch size of 64 and the initial channel number of 20.

4 More Results

4.1 Results on Randomly Sampled Architectures

In this section, we apply our NAT method to 20 architectures, which are randomly sampled from the whole architecture space. We train all the input architectures using momentum SGD with the batch size of 128 for 600 epochs. We set the initial learning rate to 0.05 and use a cosine scheduler to gradually decrease it to zero. Due to the large number of architectures, training all the models on ImageNet becomes extremely time-consuming and infeasible. Hence, we only report the results of these optimized architectures on CIFAR-10, which is a smaller dataset but can still demonstrate the effectiveness of our method.

From Figure 2, the architectures optimized by NAT outperform the original ones in terms of both model complexity and test accuracy. Thus, we draw a conclusion that the proposed method is able to produce more compact and accurate architectures compared to the original ones.

We take several architectures optimized by NAT as examples and shown them in Figure 3. From Figure 3, some computational modules in the randomly sampled architectures are replaced with skip connections or null operations in NAT based models while the accuracy improves. These results show that our NAT method is able to optimize different architectures to obtain better performance without introducing extra cost.

4.2 Effect of different graph representations for hand-crafted architectures.

Note that some nodes in the existing hand-crafted architecture only have one input. To make all the architectures share the same representation of DAG in which each node has two inputs, we add additional null operations to the nodes that has less than 2 input edges. Actually, there are multiple ways to add these null operations. Thus, a hand-crafted architecture can be transformed into different graph representations. For example, in Figure 5, the null operation of node 2 can be added to the edge from node -2 (ResnetV1), node 1 (ResNetV2) or node 0 (ResnetV3). To investigate the effect of different graph representations on NAT, we conduct more experiments on hand-crafted architectures (*e.g.*, VGG and ResNet20) in CIFAR-10. From the examples in Figure 4 and 5, our NAT based models consistently outperform the baseline models, which demonstrates the effectiveness of the proposed method.

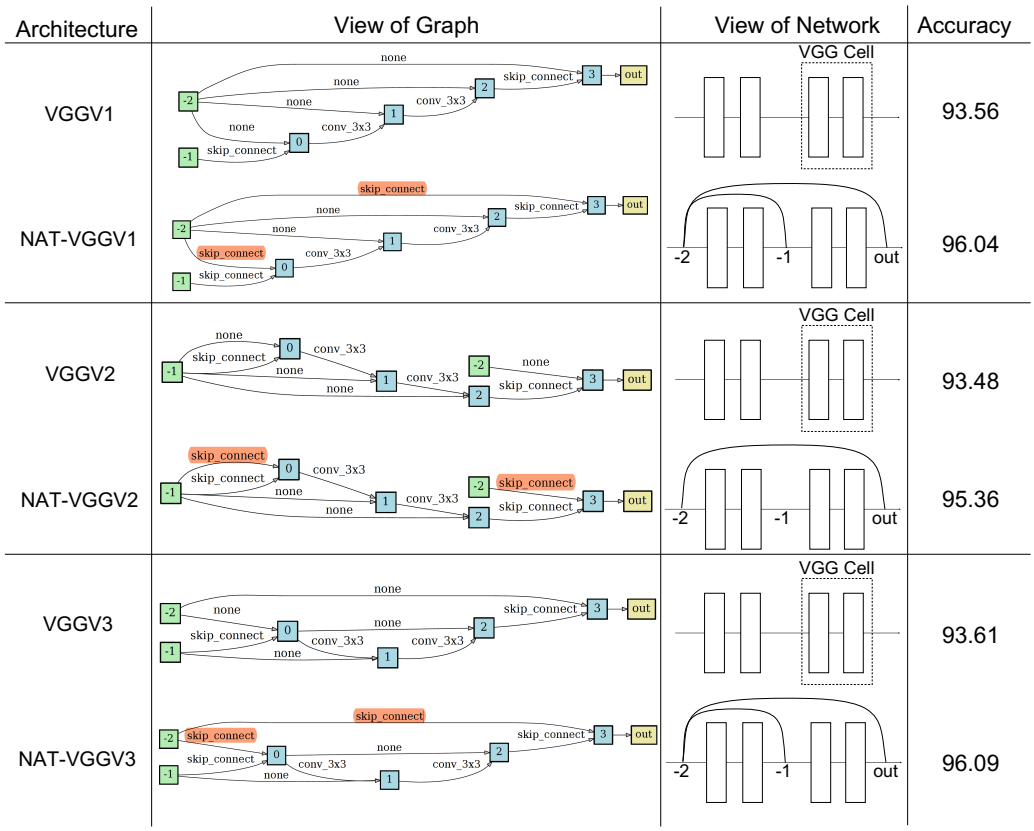

Figure 4: Optimization results with different graph representations of VGG on CIFAR-10.

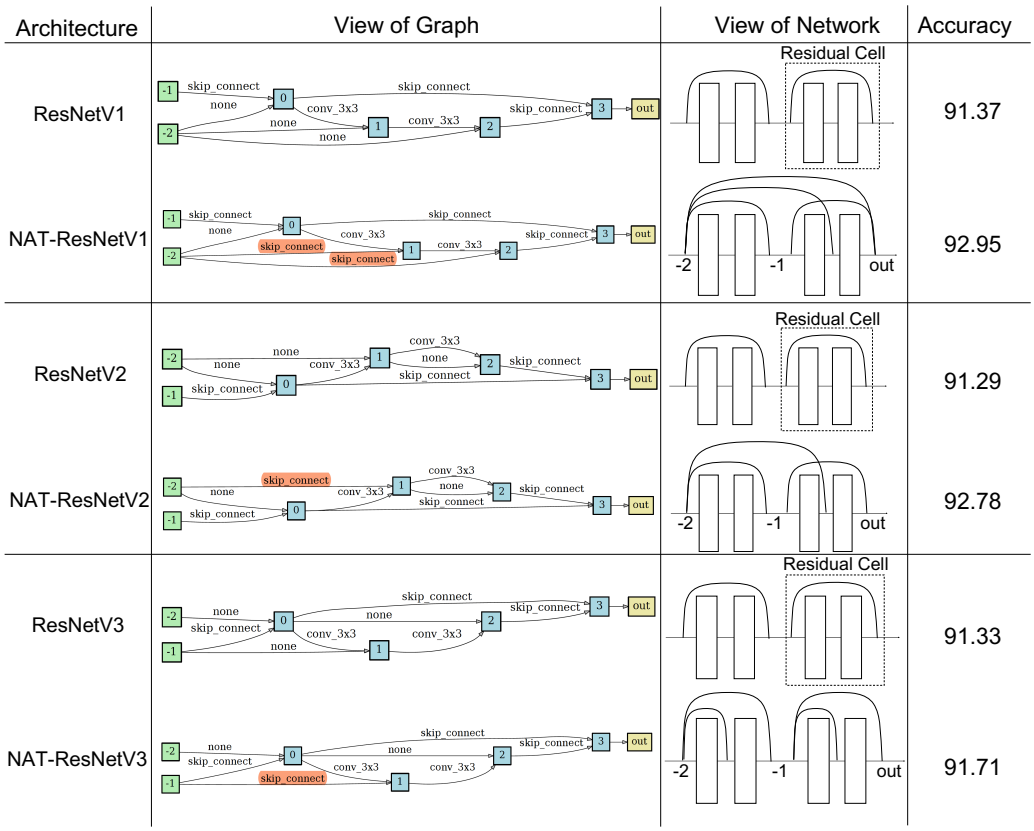

Figure 5: Optimization results with different graph representations of ResNet20 on CIFAR-10.

References

[1] J. Cao, Y. Guo, Q. Wu, C. Shen, J. Huang, and M. Tan. Adversarial learning with local coordinate coding. In *International Conference on Machine Learning*, pages 706–714, 2018.

[2] J. Cao, L. Mo, Y. Zhang, K. Jia, C. Shen, and M. Tan. Multi-marginal wasserstein gan. In *Advances in Neural Information Processing Systems*, 2019.

[3] Y. Guo, Q. Chen, J. Chen, Q. Wu, Q. Shi, and M. Tan. Auto-embedding generative adversarial networks for high resolution image synthesis. *IEEE Transactions on Multimedia*, 2019.

[4] Y. Guo, Q. Wu, C. Deng, J. Chen, and M. Tan. Double forward propagation for memorized batch normalization. In *AAAI Conference on Artificial Intelligence*, pages 3134–3141, 2018.

[5] Y. Guo, Y. Zheng, M. Tan, Q. Chen, J. Chen, P. Zhao, and J. Huang. Nat: Neural architecture transformer for accurate and compact architectures. In *Advances in Neural Information Processing Systems*, 2019.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[7] S. Lauly, Y. Zheng, A. Allauzen, and H. Larochelle. Document neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 18(1):4046–4069, 2017.

[8] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[9] C. S. Nam, A. Nijholt, and F. Lotte. *Brain–computer interfaces handbook: technological and theoretical advances*. CRC Press, 2018.

[10] J. Pan, Y. Li, and J. Wang. An eeg-based brain-computer interface for emotion recognition. In *2016 international joint conference on neural networks (IJCNN)*, pages 2063–2067. IEEE, 2016.

[11] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4095–4104, 2018.

[12] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A Unified Embedding for Face Recognition and Clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[13] Y. Sun, X. Wang, and X. Tang. Deeply Learned Face Representations are Sparse, Selective, and Robust. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900, 2015.

[14] Y. Zhang, H. Chen, Y. Wei, P. Zhao, J. Cao, X. Fan, X. Lou, H. Liu, J. Hou, X. Han, et al. From whole slide imaging to microscopy: Deep microscopy adaptation network for histopathology cancer image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 360–368. Springer, 2019.

[15] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.

[16] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.