Supplementary Materials for "Contrastive Neural Architecture Search with Neural Architecture Comparators"

Yaofo Chen[1,2*], Yong Guo[1*], Qi Chen[1], Minli Li[1], Wei Zeng[3], Yaowei Wang[2†], Mingkui Tan[1,4†]
[1]South China University of Technology, [2]Peng Cheng Laboratory, [3]Peking University,
[4]Key Laboratory of Big Data and Intelligent Robot, Ministry of Education
{sechenyaofo, guo.yong, sechenqi, seminli_li}@mail.scut.edu.cn,
weizeng@pku.edu.cn, wangyw@pcl.ac.cn, mingkuitan@scut.edu.cn

In the supplementary, we provide a detailed proof of Proposition 1, more experimental results and more details on the CTNAS method. We organize our supplementary as follows.

- In Section A, we provide the detailed proof for Proposition 1.
- In Section B, we describe how to construct the training data for CTNAS.
- In Section C, we show more training details of CTNAS.
- In Section D, we detail the evaluation method for the searched architectures.
- In Section E, we provide more details on the three considered search spaces.
- In Section F, we give more details on the evaluation metric Kendall's Tau.
- In Section G, we give more experimental results on DARTS search space.
- In Section H, we show the visualization results of the searched architectures.

A. Proof of Proposition 1

In this section, we provide the detailed proof of Proposition 1. We first introduce a useful theorem.

Let $\mathcal{X}$ be a set of samples to be ranked. Given $x, x' \in \mathcal{X}$ with the labels $\mathcal{R}_x$ and $\mathcal{R}_{x'}$, a score function $f : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$, the pairwise ranking loss is defined as $\ell_0(x, x', \mathcal{R}_x, \mathcal{R}_{x'}) = \mathbb{1}\{(\mathcal{R}_x - \mathcal{R}_{x'}) f(x, x') \leq 0\}$. The expected true risk is defined as $R_0(f) = \mathbb{E}[\ell_0]$. Since the indicator function is non-differential, directly optimize ℓ_0 is non-trivial. In practice, the pairwise ranking problem aims to learn a ranking function by optimizing a surrogate loss function. Let $\Phi(\cdot)$ be some measurable function. A surrogate loss can be defined as $\ell_\Phi(x, x', \mathcal{R}_x, \mathcal{R}_{x'}) = \Phi(-\mathrm{sgn}(\mathcal{R}_x - \mathcal{R}_{x'}) \cdot f(x, x'))$. The expected surrogate risk is defined as $R_\Phi(f) = \mathbb{E}[\ell_\Phi]$. The following theorem [14] analyzes the statistical consistency of the surrogate loss ℓ_Φ with respect to the true loss ℓ_0.

Theorem 1 *Let $R_0^* = \inf_f R_0(f)$ and $R_\Phi^* = \inf_f R_\Phi(f)$. Then for all functions f, as long as Φ is convex, we have*

$$R_0(f) - R_0^* \leq \Psi^{-1}(R_\Phi(f) - R_\Phi^*), \tag{1}$$

where

$$\Psi(x) = H^-(\frac{1+x}{2}) - H^-(\frac{1-x}{2}),$$

$$H^-(\rho) = \inf_{\alpha:\alpha(2\rho-1)\leq 0} (\rho\Phi(-\alpha) + (1-\rho)\Phi(\alpha)).$$

*Authors contributed equally.
†Corresponding author.

The above theorem states that if an algorithm can effectively minimize the right-hand side of the inequality (1) to approach zero, then the left-hand side will also approaches zero. In other words, if $\Phi(\cdot)$ is a convex function, the surrogate loss ℓ_Φ is statistically consistent with the true loss ℓ_0. Based on Theorem 1, we then prove Proposition 1.

Proposition 1 *Let $f : \Omega \times \Omega \rightarrow \mathcal{R}$ be some measurable function, $\sigma(\cdot)$ be the sigmoid function, and $\alpha, \alpha' \in \Omega$. The surrogate loss $\mathcal{L}(f; \alpha, \alpha', y) = \mathbb{E}\left[\ell(\sigma \circ f(\alpha, \alpha'), y)\right]$ is consistent with $\mathcal{L}_0(f; \alpha, \alpha', \mathcal{R}_\alpha, \mathcal{R}_{\alpha'}) = \mathbb{E}\left[\mathbb{1}\{(\mathcal{R}_\alpha - \mathcal{R}_{\alpha'}) f(\alpha, \alpha') \leq 0\}\right]$.*

Proof Given $y = \mathbb{1}\{\mathcal{R}_\alpha - \mathcal{R}_{\alpha'} \geq 0\}$ and $\sigma(x) = \frac{1}{1+e^{-x}}$, we have the following equation,

$$
\begin{aligned}
\mathcal{L}(f; \alpha, \alpha', y) =& \mathbb{E}\left[-y \log \sigma(f(\alpha, \alpha')) - (1-y) \log(1 - \sigma(f(\alpha, \alpha'))\right] \\
=& \mathbb{E}\left[-y \log \sigma(f(\alpha, \alpha')) + (1-y)(f(\alpha, \alpha') - \log \sigma(f(\alpha, \alpha')))\right] \\
=& \mathbb{E}\left[(1-y) f(\alpha, \alpha') + \log(1 + \exp(-f(\alpha, \alpha')))\right] \\
=& \mathbb{E}\left[\log(1 + \exp(-\mathrm{sgn}(\mathcal{R}_\alpha - \mathcal{R}_{\alpha'}) \cdot f(\alpha, \alpha')))\right] \\
=& \mathbb{E}\left[\Phi(-\mathrm{sgn}(\mathcal{R}_\alpha - \mathcal{R}_{\alpha'}) \cdot f(\alpha, \alpha'))\right],
\end{aligned}
\tag{2}
$$

where $\Phi(x) = \log(1 + \exp(x))$. Since $\Phi(x)$ is a convex function, according to Theorem 1, the surrogate loss $L(f; \alpha, \alpha', y)$ is consistent with $L_0(f; \alpha, \alpha', \mathcal{R}_\alpha, \mathcal{R}_{\alpha'})$.

B. More Details on Data Construction

As mentioned in Section 4, the training of NAC relies on a set of training data. Here, we provide more details on the training data of NAS-Bench-101 search space [21], MobileNetV3-like search space [8] and DARTS search space [13].

Data Construction in NAS-Bench-101 Search Space. To train the proposed CTNAS model, we use the architectures and the corresponding performance as training data in the NAS-Bench-101 search space. Google has released 423k architectures with corresponding performance on CIFAR-10 in the NAS-Bench-101 search space. The performance includes the training accuracy, validation accuracy, test accuracy, training time and the number of trainable parameters. We randomly sample 423 architectures (0.1% of the whole architectures) as the training set and 100 architectures as the validation set. For training the proposed CTNAS, we construct the training architecture pair (α, α') by randomly sampling from the training architecture set. The label of the architecture pair (α, α') is computed from the validation accuracy of architecture α and α'.

Data Construction in MobileNetV3-like Search Space. To obtain the training data pairs for CTNAS in the MobileNetV3-like search space, we train a supernet with a progressive shrinking strategy following the setting in [1]. We then randomly sample 1000 architectures in the search space and apply the weights from the supernet ($1.0\times$) to these architectures. Then we evaluate the architectures on 10000 validation images sampled from the training set of ImageNet to obtain the validation accuracy. The evaluation takes less than 0.1 GPU days. Finally, we compute the label of the training pairs for NAC with the validation accuracy of the architectures.

Data Construction in DARTS Search Space. Training CTNAS requires some architectures with the corresponding performance. To this end, we first train a supernet for 120 epochs with a uniform sampled strategy [5]. Following DARTS [13], we train the supernet on 50% of the official training set of CIFAR-10 and evaluate them for the validation accuracy on the remain 50% using standard data augmentation techniques [7]. The batch size and initial learning rate are set to 96 and 0.01, respectively. Then, we randomly sample 1000 architectures in the search space. The validation accuracy of each sampled architecture is computed by inheriting weights from the supernet. The label of the training pairs for NAC is computed from the average validation accuracy of the last 10 epochs in the supernet training.

C. More Training Details of CTNAS

In this section, we show more training details of CTNAS while searching in the NAS-Bench-101 [21] and MobileNetV3-like search space [8]. We follow the settings in [10] to build our NAC model. Specifically, we first use two GCN layers to extract features of two input architecture. Then, we concatenate the features and send them to a fully-connected (FC) layer to produce the probability of the first architecture being better than the other one. Since the number of nodes of the architecture graph may be less than 7 in NAS-Bench-101 search space, we pad the size of the adjacency matrix to 7 with zero paddings.

In the training, we train the CTNAS model for 10k iterations. We use Adam with a learning rate of 2×10^{-4} and a weight decay of 5×10^{-4} as the optimizer. The training batch size of NAC is set to 256. To explore more training data, we randomly sample 512 architecture pairs every iteration and add the top 256 architecture pairs with the predicted labels according to comparison probability into the training data batch (See Algorithm 3). We update the baseline architecture every 1000 iteration. We add the controller's sample entropy to the reward, which is weighted by 5×10^{-4}.

D. More Details on Architecture Evaluation

In this section, we elucidate the details of evaluation method for the searched architectures in NAS-Bench-101 [21], MobileNetV3-like [8] and DARTS [13] search space.

More Evaluation Details in NAS-Bench-101 search space. NAS-Bench-101 has released the performance of all the architectures in its search space, including the training accuracy, the validation accuracy and the test accuracy. Following the settings in NAS-Bench-101, we report the average test accuracy of the searched architecture over 3 different runs.

More Evaluation Details in MobileNetV3-like search space. We evaluate the architectures searched in MobileNetV3-like search space on ImageNet. Following the setting in [13], the number of multiply-adds (MAdds) in the searched architectures is restricted to be less than 600M. Following the setting in [1], we directly apply the weights form the supernet (1.0$\times$) to the searched architectures, and then evaluate them on the validation set of ImageNet [3] without finetune.

More Evaluation Details in DARTS search space. To evaluate the searched architectures, we train them from scratch on CIFAR-10. We build the final convolution network with 18 learned cells, including 16 normal cells and 2 reduction cells. The two reduction cells are put at the $1/3$ and $2/3$ depth of the network, respectively. The initial number of the channels is set to 44. Following the setting in [13], we train the model for 600 epochs using the batch size of 96. The training images are padded 4 pixels on each side. Then the padded images or their horizontal flips are randomly cropped to the size of 32×32. We use cutout [4] with a length of 16 in the data augmentation. We use an SGD optimizer with a weight decay of 3×10^{-4} and a momentum of 0.9. The learning rate starts from 0.025 and follows the cosine annealing strategy with a minimum of 0.001. Additional enhancements include path dropout [11] of probability of 0.2 and auxiliary towers [19] with a weight of 0.4.

E. More Details on Search Space

In this paper, we consider three search spaces, namely the NAS-Bench-101 search space [21], MobileNetV3-like search space [8] and DARTS search space [13]. We show the details of these search spaces below.

NAS-Bench-101 Search Space. NAS-Bench-101 defines a search space that is restricted to small topology structures, usually called cells. Each cell contains 7 nodes at most, including IN and OUT nodes, which represent the input and output tensors of the cell. The remaining nodes can be selected from 3×3 convolution, 1×1 convolution and 3×3 max pooling. The number of connections in a cell is limited to no more than 9. The whole convolution network is stacked with 3 blocks followed by a global average pooling and a fully-connected layer. Each block is stacked with 3 cells followed by a downsampling layer, where the image height and width are halved via max-pooling and the channel count is doubled. The initial layer of the model is a stem consisting of a 3×3 convolution with 128 output channels.

MobileNetV3-like Search Space. We consider the search space proposed by MobileNetV3. The model is divided into 5 units with gradually reduced feature map size and increased channel number. Each unit consists of 4 layers at most where only the first layer has stride 2 if the feature map size decrease. All the other layers in the units have stride 1. In our experiments, we search for the number of layers in each unit (chosen from $\{2, 3, 4\}$), the kernel size in each layer (chosen from $\{3, 5, 7\}$) and the width expansion ratio in each layer (chosen from $\{3, 4, 6\}$).

DARTS Search Space. We consider a cell-based search space proposed by DARTS [13], which includes two cell types, namely the normal cell and reduction cell. The normal cell keeps the spatial resolution of feature maps. The reduction cell reduces the height and width of feature maps by half and doubles the number of channels. Each cell contains 7 nodes, including 2 input nodes, 4 intermediate nodes and 1 output node. The output node is the concatenation of the 4 intermediate nodes. There are 8 candidate operations between two nodes, including 3×3 depthwise separable convolution, 5×5 depthwise separable convolution, 3×3 max pooling, 3×3 average pooling, 3×3 dilated convolution, 5×5 dilated convolution, identity, and none.

F. More Details on Kendall's Tau

To evaluate the performance estimators, we use Kendall's Tau (KTau) [18] to compute the ranking correlation between the predicted performance and the performance obtained by training from scratch over a set of architectures. Given n architectures $\{\alpha_i\}_{i=1}^n$, there are $\binom{n}{2} = \frac{n(n-1)}{2}$ architectures pairs. The KTau τ can be computed by

$$\begin{aligned}\tau = \binom{n}{2}^{-1} \sum_{i<j} \big[&\mathrm{sgn}(f(\alpha_i) - f(\alpha_j)) \cdot \\ &\mathrm{sgn}(\mathcal{R}(\alpha_i, w_{\alpha_i}) - \mathcal{R}(\alpha_j, w_{\alpha_j})) \big],\end{aligned} \tag{3}$$

where $\mathrm{sgn}(\cdot)$ is a sign function, $f(\cdot)$ is a performance predictor, $\mathcal{R}(\cdot, w)$ is the performance obtained by training from scratch. The value of τ represents the correlation between the ranking of the predicted performance and the performance obtained by training from scratch. A higher KTau means the predicted performance ranking is more accurate.

We compare the KTau of different NAS methods in the NAS-Bench-101 search space. The results are shown in the Table A. Our proposed CTNAS achieves higher KTau (0.751) than the considered methods, including ReNAS [22], SPOS [5] and FairNAS [2]. These results mean that the performance ranking predicted by our method is very close to that of training from scratch. In the search process, the higher KTau means a more accurate reward, often resulting in better search performance.

Table A. Comparisons of the KTau in the NAS-Bench-101 search space for different NAS methods.

Method	ReNAS [22]	SPOS [5]	FairNAS [2]	CTNAS (Ours)
KTau	0.634	0.195	-0.232	**0.751**

G. More Results on DARTS Search Space

Implementation Details. We train a supernet for 120 epochs with a uniform sampled strategy [5]. We randomly sample 1000 architectures in the DARTS search space [13], and obtain their validation accuracy by inheriting weights from the supernet to construct the data for NAC training. We train the controller for 10k iteration with a batch size of 1 following the settings in [16]. We add the controller's sample entropy to the reward, which is weighted by 5×10^{-4}.

Comparisons with State-of-the-art Methods. We compare the searched performance of our proposed CTNAS with the state-of-the-art methods in DARTS [13] search space. From Table B, the architecture searched by CTNAS outperforms both manually designed and automatically searched architectures. We also compare the searched architecture with the best one in 1000 sampled architectures. The searched architecture achieves higher average accuracy (97.41% *vs.* 97.33%), which demonstrates the effectiveness of the proposed CTNAS. Moreover, our CTNAS takes 0.2 GPU days to train the supernet and 0.1 GPU day for the search phrase, which is much faster than the considered NAS methods. The reason is that our NAC has a much lower time cost while evaluating architectures (See results in Sec. 6.1 in the paper).

Table B. Comparisons with state-of-the-art models on CIFAR-10. "cutout" indicates evaluating the architecture using cutout [4] data argumentation. "–" means unavailable results.

Architecture	Test Accuracy (%)	#Params. (M)	Search Cost (GPU days)
DenseNet-BC [9]	96.54	25.6	–
PyramidNet-BC [6]	96.69	26.0	–
Random search baseline	96.71 ± 0.15	3.2	–
NASNet-A + cutout [24]	97.35	3.3	1,800
NASNet-B [24]	96.27	2.6	1,800
NASNet-C [24]	96.41	3.1	1,800
AmoebaNet-A + cutout [17]	96.66 ± 0.06	3.2	3,150
AmoebaNet-B + cutout [17]	96.63 ± 0.04	2.8	3,150
SNAS [20]	97.02	2.9	1.5
ENAS + cutout [16]	97.11	4.6	0.5
NAONet [15]	97.02	28.6	200
GHN [23]	97.16 ± 0.07	5.7	0.8
PNAS + cutout [12]	97.17 ± 0.07	3.2	–
DARTS + cutout [13]	97.24 ± 0.09	3.4	4
Best in Sampled Architectures + cutout	97.33 ± 0.07	3.7	–
CTNAS + cutout (Ours)	$\mathbf{97.41} \pm \mathbf{0.04}$	3.6	**0.3**

H. Visualization Results of Searched Architectures

We show the visualization results of searched architectures of three consider search space in Figure A, B and C, respectively. In Figure A, we show the best architecture searched by CTNAS in NAS-Bench-101 search space, which is the third best architecture in the whole search space. In Figure B, we show the resulting architecture searched in MobileNetV3-like search space, which achieves 77.3% top-1 accuracy and 93.4% top-5 accuracy on ImageNet. These architectures searched by CTNAS outperform existing human-designed architectures and automatically searched architectures. The visualization results of the normal cell and reduction cell searched by CTNAS in DARTS search space are shown in Figure C, which achieves the average accuracy of 97.41% on CIFAR-10.

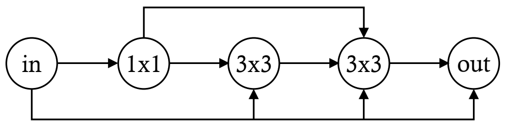


Figure A. The best architecture searched by CTNAS in NAS-Bench-101 search space.

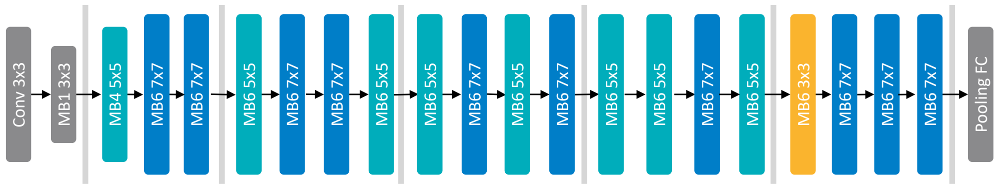
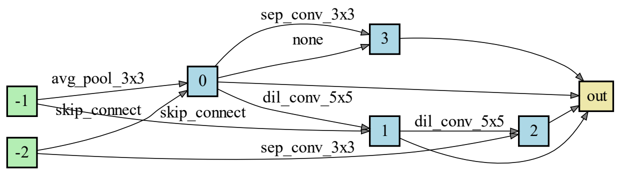
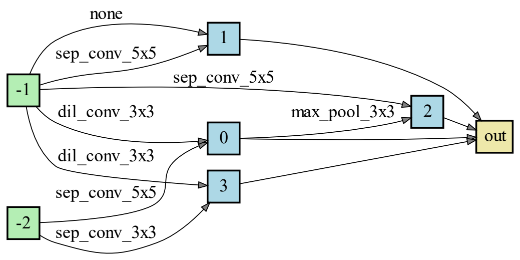


Figure B. The architecture searched by CTNAS in MobileNetV3-like search space.



(a) Normal cell.



(b) Reduction cell.

Figure C. The architecture of the convolutional cells found by CTNAS in DARTS search space. Following [13], we represent the convolutional cell as a directed acyclic graph, which consists of 7 nodes, including 2 input nodes (green boxes), 4 intermediate nodes (blue boxes) and 1 output node (yellow box). Each labeled edge in the cell denotes an operation (*e.g.*, 3×3 separable convolution). The unlabeled edges denote the feature average operation over all intermediate nodes to obtain the output node.

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. In *International Conference on Learning Representations*, 2020.

[2] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

[5] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *The European Conference on Computer Vision*, 2020.

[6] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 6307–6315, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *The IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[11] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations*, 2017.

[12] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *The European Conference on Computer Vision*, pages 19–35, 2018.

[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[14] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[15] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems*, pages 7827–7838, 2018.

[16] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018.

[17] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[18] Pranab Kumar Sen. Estimates of the regression coefficient based on Kendall's Tau. *Journal of the American Statistical Association*, 63(324):1379–1389, 1968.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[20] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

[21] 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, 2019.

[22] Yixing, Yunhe Wang, Kai Han, Shangling Jui, Chunjing Xu, Qi Tian, and Chang Xu. ReNAS: Relativistic evaluation of neural architecture search. *arXiv preprint arXiv:1910.01523*, 2019.

[23] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2018.

[24] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.