Lightweight Neural Architecture Search with Parameter Remapping and Knowledge Distillation

Hayeon Lee[*1] **Sohyun An**[*1] **Minseon Kim**[1] **Sung Ju Hwang**[1,2]

[1]KAIST, Seoul, South Korea
[2]AITRICS, Seoul, South Korea

Abstract Designing diverse neural architectures taking into account resource constraints or datasets is one of the main challenges in Neural Architecture Search (NAS). However, existing sample-based or one-shot NAS approaches require excessive time or computational cost to be used in multiple practical scenarios. Recently, to alleviate such issues, zero-shot NAS methods that are efficient proxies have been proposed, yet their performance is rather poor due to the strong assumption that they predict the final performance of a given architecture with random initialization. In this work, we propose a novel NAS based on block-wise parameter remapping (PR) and knowledge distillation (KD), which shows high predictive performance while being fast and lightweight enough to be used iteratively to support multiple real-world scenarios. PR significantly shortens training steps and accordingly we can reduce the required time/data for KD to work as an accurate proxy to just few batches, which is largely practical in real-world. In the experiments, we validate the proposed method for its accuracy estimation performance on CIFAR-10 from the MobileNetV3 search space. It outperforms all relevant baselines in terms of performance estimation with only 20 batches.

1 Introduction

Neural Architecture Search (NAS), which aims to automate the neural architecture design process to build a better model with stronger performance and higher efficiency for a given task, is born out of practical needs that reduce human efforts to design multiple models differently considering resource budgets and datasets in real-world scenarios. Recently, while many NAS methods [4, 42, 16, 19, 23, 29, 21, 38, 10] have been proposed and demonstrated their potential in benchmark settings, existing NAS frameworks suffer from being actively used for real-world tasks due to computationally expensiveness and time-consuming costs. For example, the search costs of sample-based NAS methods [35, 6] are 40,000 and 200 GPU hours for each device. Hardware-aware one-shot NAS methods [5, 33] alleviate such limitations by decoupling the supernet training process and the search process. We refer to those methods as the "one-shot NAS" because training of supernet [26], which is trained progressively while shrinking each dimension in the search space, requires constant one-time training cost. However, building a supernet still requires excessive computational cost and time such as 1,200 GPU hours on V100 GPU on a single dataset [5], which hinders their applicability to a new dataset. Moreover, training a supernet requires complicated techniques [5, 33] and the multi-model forgetting phenomenon can occur [26]. To further reduce the cost of architecture evaluation, zero-shot NAS methods [24, 1] that do not require training have been proposed. However, despite their high efficiency, the accuracy of prediction is very poor and most of them cannot capture the distributional distinctiveness of the various datasets [26].

To overcome such limitations, in this work, we propose a lightweight NAS approach based on block-wise parameter remapping (PR) and knowledge distillation (KD) for only in few batches, enabling rapid prediction of the performance of a given neural architecture using just few batches

[*] Equal Contribution

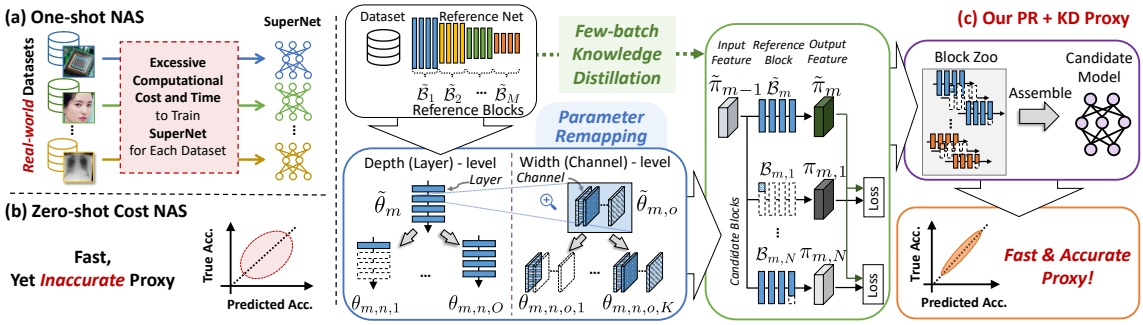

Figure 1: **(a)** While one-shot NAS frameworks train the supernet on each new dataset, such training is too difficult and requires a heavy cost to actively use for each individual in the real world. **(b)** Zero-shot NAS methods have shown high efficiency in the search process, yet their proxy performances need to be improved. **(c)** We design lightweight proxy by combining parameter remapping and knowledge distillation. The proposed proxy is fast enough and show accurate proxy performance to be used in practical scenarios.

(see Figure 1). Specifically, we first prepare reference blocks by learning a reference model consisting of reference blocks, on a target dataset. Then, we construct a block zoo via block-wise PR and KD from the reference to candidate blocks which are included in the search space. The former is performed by two-level steps such as layer-wise and channel-wise PR between each reference block and possible other candidate blocks. The latter is to make candidate blocks mimic each reference block via distilling knowledge of output feature map. During search process, candidate models are made by assembling the blocks provided from the block zoo. Finally, the accuracy of a candidate model with remapped and distilled parameters is used as a proxy to evaluate the final performance of the model. After selecting the final model based on the proposed proxy, we can obtain the model with high-performing parameters on target dataset rapidly though a few fine-tuning from distilled parameters instead of training a large number of parameters from random initialization. Notably, the PR from reference blocks before KD significantly improves the search speed compared to a proxy using KD only. Moreover, since the building block zoo is a one-time cost, the block zoo once created can be used repeatedly to support designing models suitable for different resource budgets.

The experimental results demonstrate that the proposed method shows a high-ranking correlation between the predicted values and the final performance of the architecture candidates in the MobileNetV3 search space, outperforming other baselines. We believe that the proposed method with high prediction performance in decent time is a new alternative bridge between the high cost of one-shot NAS and the poor performance of zero-shot NAS for real-world scenarios.

2 Related Work

NAS Methods Neural Architecture Search (NAS) suggests to automate designing the optimal architecture. Early NAS, which is based on reinforcement learning [42, 4, 41] or evolutionary algorithms [31, 20, 11, 30], was impossible to be applied without huge computing resources as the process of training and evaluating the sampled model was repeated. To overcome tremendous computing costs, two types of approaches are introduced : one-shot [2, 33] and zero-shot NAS methods [1, 9, 18, 22, 24, 27]. The former can reduce the evaluation cost by sharing parameters in the search space. However, they still take a lot of computational cost and time to train the supernet. Also, because the parameters are shared among all architectures in the search space, multi-model forgetting can occur [26]. The later zero-shot NAS, reduced both training and evaluation costs because it measures the performance of a model in random state. However, their proxies show

worse performance than the parameter size or FLOPs-based proxies in some search space [26]. In addition, there is a problem that the stability of the proxy is different for each search space.

Parameter Remapping Parameter remapping (PR) can reduce the tedious process caused by training a new model from scratch by reusing the parameters of the previously trained network. Some NAS methods [29, 11, 7, 34, 5] use parameter sharing to evaluate the architecture in the search space, which can be interpreted as PR on various levels, e.g., kernel size, depth, width, and resolution. Furthermore, other studies [8, 13, 12] utilize PR for effective initialization of the models. However, in previous studies, conducting PR was for succession from a supernet to a child model to evaluate the performance as it is, or for accelerating training of a new model. We present a novel method of taking advantage of PR as a proxy by applying PR before the knowledge distillation (KD) so that our proxy can approximate the actual performance of a model candidate within a few batches.

Block-wise Knowledge Distillation Knowledge distillation (KD) is proposed to transfer knowledge from a large teacher model to a smaller student model [3]. In order to make the student model imitate the teacher model, the student model is trained with soft labels [14], logits [36] or intermediate features [28, 32, 37, 39, 40] from the teacher model. To perform block-wise KD, [37] regards the model as a block-unit configuration and proposes a method for sequential KD for each block. Furthermore, to shorten the time required, [17] presents a parallel method for block-wise KD. However, the existing method is difficult to operate as a fast proxy because it takes a lot of time by performing KD. We address this problem by devising a new proxy that combines PR with KD.

3 Building a Lightweight Accuracy Proxy

In this section, we introduce a guideline to build our lightweight proxy to be used for the search process in several practical scenarios.

Preparing a Reference Model We first train the reference model $\tilde{\mathcal{F}}(\cdot;\tilde{\boldsymbol{\theta}})$ on a target dataset D, which the model is compositions of M blocks by $\tilde{\mathcal{F}}(x;\tilde{\boldsymbol{\theta}}) = \tilde{\mathcal{B}}_M \circ \cdots \circ \tilde{\mathcal{B}}_1(x;\tilde{\boldsymbol{\theta}}_1)$ where x is an input tensor (input image or intermediate feature) and $\mathcal{B}_i$ is the i-th block of the model. In this work, to maximize the effectiveness of KD in student networks, we regard the largest network we can get from the search space as the reference model.

Building a Block Zoo In the next two steps, we build a block zoo by making student blocks $\{\mathcal{B}_{m,n}(\cdot;\boldsymbol{\theta})\}_{n=1}^{N}$ mimic the trained reference blocks $\tilde{\mathcal{B}}_m$ for $m = 1,\ldots,M$ via PR and KD.

Block-wise Parameter Remapping In this step, we map the trained parameters $\tilde{\boldsymbol{\theta}}_m$ of the m-th reference block $\tilde{\mathcal{B}}_m$ to $\{\boldsymbol{\theta}_{m,n}\}_{n=1}^{N}$ of the candidate blocks $\{\mathcal{B}_{m,n}\}_{n=1}^{N}$. Unlike KD-only performance proxy [17, 25], we observe that including PR improves the performance and efficiency of the proxy (see Figure 2). As the number of layers and parameters (channels) for each layer between the reference and candidate block can be different, we consider layer-wise and channel-wise PR. In other words, a candidate block $\mathcal{B}_{m,n}(x) = \mathcal{O}_{m,n,O_{m,n}} \circ \cdots \circ \mathcal{O}_{m,n,1}(x;\boldsymbol{\theta}_{m,n,1})$ where the number of layers in $\mathcal{B}_{m,n}$ is $O_{m,n}$ and $\mathcal{O}_{m,n,o}$ is the o-th layer (operations) with parameter $\boldsymbol{\theta}_{m,n,o}$ in the n-th candidate block $\mathcal{B}_{m,n}$. We first consider layer-wise PR. For each block $\mathcal{B}_{m,n}$, we assume the number of layers $O_{m,n}$ in a candidate is always less than or equal to that of the reference block $O_{m,n} \leq \tilde{O}_m$. Then we remap parameters of each layer as follows:

$$\boldsymbol{\theta}_{m,n,o} = \tilde{\boldsymbol{\theta}}_{m,o} \quad \text{for} \;\; o = 1,\ldots,O_{m,n}. \tag{1}$$

Next, we consider channel-wise PR. Similar with the layer-wise PR, we assume that the number of kernels $K_{m,n,o}$ of each layer $\mathcal{O}_{m,n,o}$ in a candidate block is always less than or equal to that of the reference block $K_{m,n,o} \leq \tilde{K}_{m,o}$. We can remap the channels for each layer $\mathcal{O}_{m,n,o}$ as follows:

$$\boldsymbol{\theta}_{m,n,o,k} = \tilde{\boldsymbol{\theta}}_{m,o,k} \quad \text{for} \;\; k = 1,\ldots,K_{m,n,o}. \tag{2}$$

Table 1: Comparison of Spearman's rank correlation between the estimated and actual accuracies on CIFAR-10. B is the number of batches required to train the reference net (one-time cost). i is the number of candidate networks.

Type	Proxy Type	# of Training Batch	Final Accuracy of Scratch (SC)	Params. Remap.(PR)	Knowl. Distill (KD)
Baseline	Gradnorm [1]	$20i$	0.540	0.739	0.679
	Fisher [1]	$20i$	0.422	0.578	0.523
	Snip [1]	$20i$	0.408	0.610	0.573
	Synflow [1]	$20i$	0.208	0.449	0.416
	NASWOT [24]	$20i$	0.785	0.765	0.749
	FLOPs	-	0.328	0.490	0.556
	L2norm	$20i$	0.070	0.275	0.249
Ablation Study	Initial Accuracy of PR	-	0.275	0.214	0.343
	Initial Accuracy of KD	$B + 20i$	0.216	0.232	0.202
		$B + 650i$ (1 epoch)	0.786	0.861	0.853
Ours	**Initial Accuracy of PR + KD**	$B + 20i$	**0.797**	**0.893**	**0.887**

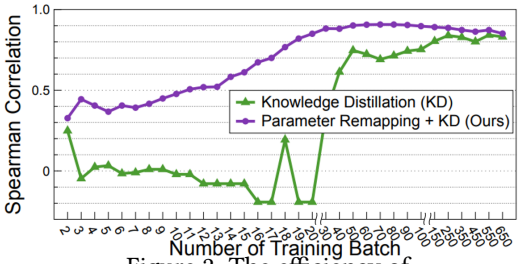

Figure 2: The efficiency of the proposed proxy (PR+KD).

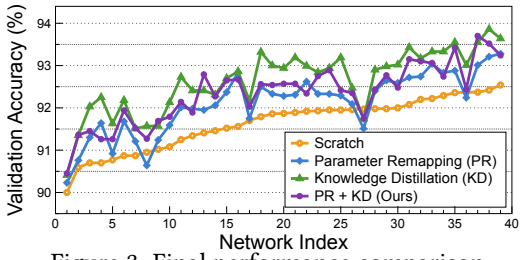

Figure 3: Final performance comparison dependent on initialization types.

Block-wise Few-batch Knowledge Distillation We conduct a distillation of block-wise feature maps by minimizing the Mean Square Error (MSE) loss between the reference block $\tilde{\mathcal{B}}_m(\cdot;\tilde{\boldsymbol{\theta}})$ and the parameter-remapped student block $\mathcal{B}_{m,n}(\cdot;\boldsymbol{\theta})$ in Equation (3). C is the number of channels in a feature map, $\boldsymbol{\theta}_{m,n}$ are the weights of the candidate block $\mathcal{B}_{m,n}$, $\tilde{\pi}_m = \tilde{\mathcal{B}}_m(\tilde{\pi}_{m-1};\tilde{\boldsymbol{\theta}}_m)$ is the target output feature of the reference block $\tilde{\mathcal{B}}_m$, $\pi_{m,n} = \mathcal{B}_{m,n}(\tilde{\pi}_{m-1};\boldsymbol{\theta}_{m,n})$ is the output of the n-th candidate block $\mathcal{B}_{m,n}$.

$$\mathcal{L}(\boldsymbol{\theta}_{m,n};\tilde{\pi}_{m-1},\tilde{\pi}_m) = \frac{1}{C}\sum_{c=1}^{C}\|\tilde{\pi}_m^c - \pi_{m,n}^c\|^2, \tag{3}$$

Now, we can perform training (block-wise KD) within just few batches T as follows:

$$\boldsymbol{\theta}_{m,n}^{(t+1)} = \boldsymbol{\theta}_{m,n}^{(t)} - \alpha\nabla_{\boldsymbol{\theta}_{m,n}^{(t)}}\mathcal{L}(\boldsymbol{\theta}_{m,n};\tilde{\pi}_{m-1}^{(t)},\tilde{\pi}_m^{(t)}) \quad \text{for } t = 1,\ldots,T, \tag{4}$$

where α is a learning rate. We conduct the distillation step with Equation (3) for all reference blocks $\{\tilde{\mathcal{B}}_m(\cdot;\tilde{\boldsymbol{\theta}})\}_{m=1}^{M}$. The total number of components in the block zoo is $M \times N$.

Scoring Model Candidates We can assemble any candidate model $\mathcal{F}_i(\cdot;\boldsymbol{\theta})$ by taking the blocks corresponding to a configure of the model from the block zoo $\{\mathcal{B}_{m,n}(\cdot;\boldsymbol{\theta})\}$ for $m = 1\ldots M$ and $n = 1\ldots N$. We use an average accuracy of the given model $\mathcal{F}_i(\cdot;\boldsymbol{\theta})$ computed on T batches that were used to train blocks, as the proxy for the final accuracy of the model. Moreover, we can obtain the final model by fine-tuning and it outperforms the baselines trained from scratch (see Figure 3).

4 Experiment

We validate the performance of our proxy on the accuracy prediction of architectures on CIFAR-10. We consider **MobileNetV3** [15, 5, 25] search space which supports many CNNs of different sizes that is tuned for mobile applications. Specifically, our search space consists of 5 stages, and in

each stage, the number of layers ranges across $\{1, 2, 3, 4\}$, the kernel size is 3, and the channel expansion ratio should be chosen from $\{2, 4, 6, 8, 10\}$. This leads the search process to a choice out of $(5^1 + 5^2 + 5^3 + 5^4)^5 \approx 10^{15}$.

Effectiveness and Efficiency of the Proposed Proxy In Table 1, we report the Spearman's rank correlation (higher the better) between the estimated accuracies and three types of actual accuracies when 40 neural architectures are trained after random initialization (SC), parameter remapping (PR), and knowledge distillation (KD), respectively. We compare against two types of baselines: 1) a proxy predictor with FLOPs and 2) zero-shot proxies [1, 24] such as Gradnorm, NASWOT, etc. For fair comparison with our method, the results of zero-shot proxies are obtained using the average zero-shot proxy values measured with those same T batches in Equation (4) for a given model. The results show that zero-shot proxies are better than FLOPs based proxy, yet their prediction performances are still low. Surprisingly, our proxy achieves the best Spearman's rank correlations of 0.797, 0.893, and 0.887 on SC, PR, and KD, respectively. This shows the clear advantage of our method, as the superior estimation accuracy with decent sample-efficiency in practical scenarios.

Ablation Study We analyze the effect of the PR and KD combination in Table 1. We observe that using both PR and KD (PR + KD) largely outperforms the others that consider only a single component. Correlation values of PR-only proxy are less than 0.343 on three final accuracies and those of KD-only (20 batches) proxy are less than 0.232. Even if we use more training batches such as 650, KD-only proxy still underperforms the proposed proxy. In Figure 2, we further demonstrate the effect of the number of training batches on the performance of the accuracy proxy. In particular, when the number of training batches is 15 or more, our PR + KD proxy achieve over 0.6 correlation on 40 neural architectures. Contrarily, KD-only proxy shows very poor performance with few training batches (<20). Here, we show that PR largely helps our proxy to predict the performance more accurately even in the early batches by utilizing the trained parameters.

Final Performance with Initialization Type In Figure 3, we analyze the effect of fine-tuning the pre-trained models with the proposed method. Most of fine-tuning knowledge-distilled models (KD) outperform models that are trained from the random initialization (SC). Initializing with PR or PR + KD also shows better performance compared to SC, and comparable performance to KD-only.

5 Conclusion

We proposed a simple-yet-effective accuracy proxy that combines parameter remapping (PR) and knowledge distillation (KD) which allows NAS to estimate accuracy more precisely with the fast time speed. Specifically, we build a block zoo by remapping the parameters from trained model on target dataset and distilling its features block-wisely. The PR significantly reduces the number of training steps as just few steps during the KD. Training accuracy of models that is assembled with blocks from block zoo is used as our proxy. Moreover, the model is also used as a pre-trained weight that makes model to obtain higher performance at the end. We validated our method by measuring its accuracy estimation performance on CIFAR-10. Our method outperforms the baselines with high predictive performance within only 20 training batches.

6 Limitations and Broader Impact Statement

Limitations While we have observed that the PR from the reference to the candidate block from the front of the layers and channels of the reference are sufficient to obtain a good proxy, we believe that deciding which channel parameters of the reference is mapped to those of the candidate block is important to get better proxy. In addition, more flexible search can be possible by performing PR and KD from a reference block with a smaller size to the blocks with larger size.

Broader Impact Since our method requires only a few batches to evaluate each network, we can largely reduce the waste of energy consumption and CO2 emissions. Since the proposed method is fast, lightweight, and high performing, it can have a huge impact as people can easily use it to design a model optimized for their own dataset.

Ackowledgement This work was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

[1] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane. Zero-cost proxies for lightweight nas. *International Conference on Learning Representations (ICLR)*, 2021.

[2] Y. Akimoto, S. Shirakawa, N. Yoshinari, K. Uchida, S. Saito, and K. Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *International Conference on Machine Learning (ICML)*, 2019.

[3] J. Ba and R. Caruana. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27, 2014.

[4] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

[5] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.

[6] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[7] S. Chen, Y. Chen, S. Yan, and J. Feng. Efficient differentiable neural architecture search with meta kernels. *arXiv preprint arXiv:1912.04749*, 2019.

[8] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *International Conference on Learning Representations (ICLR)*, 2016.

[9] W. Chen, X. Gong, and Z. Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *International Conference on Learning Representations (ICLR)*, 2021.

[10] X. Chen, R. Wang, M. Cheng, X. Tang, and C.-J. Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2021.

[11] T. Elsken, J. H. Metzen, and F. Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *International Conference on Learning Representations (ICLR)*, 2019.

[12] J. Fang, Y. Chen, X. Zhang, Q. Zhang, C. Huang, G. Meng, W. Liu, and X. Wang. Eat-nas: Elastic architecture transfer for accelerating large-scale neural architecture search. *Science China Information Sciences*, 64(9):1–13, 2021.

[13] J. Fang, Y. Sun, Q. Zhang, K. Peng, Y. Li, W. Liu, and X. Wang. Fna++: Fast network adaptation via parameter remapping and architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):2990–3004, 2020.

[14] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[15] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[16] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[17] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, and X. Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1989–1998, 2020.

[18] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition. *International Conference on Computer Vision (ICCV)*, 2021.

[19] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[20] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations (ICLR)*, 2018.

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

[22] V. Lopes, S. Alirezazadeh, and L. A. Alexandre. Epe-nas: Efficient performance estimation without training for neural architecture search. In *International Conference on Artificial Neural Networks*, pages 552–563. Springer, 2021.

[23] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu. Neural architecture optimization. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[24] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley. Neural architecture search without training. In *International Conference on Machine Learning (ICML)*, pages 7588–7598. PMLR, 2021.

[25] B. Moons, P. Noorzad, A. Skliar, G. Mariani, D. Mehta, C. Lott, and T. Blankevoort. Distilling optimal neural networks: Rapid search in diverse spaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

[26] X. Ning, C. Tang, W. Li, Z. Zhou, S. Liang, H. Yang, and Y. Wang. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] D. S. Park, J. Lee, D. Peng, Y. Cao, and J. Sohl-Dickstein. Towards nngp-guided neural architecture search. *arXiv preprint arXiv:2011.06006*, 2020.

[28] N. Passalis and A. Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284, 2018.

[29] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning (ICML)*, 2018.

[30] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence (AAAI)*, 2019.

[31] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning (ICML)*, 2017.

[32] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *International Conference on Learning Representations (ICLR)*, 2015.

[33] M. Sahni, S. Varshini, A. Khare, and A. Tumanov. Compofa: Compound once-for-all networks for faster multi-platform deployment. *International Conference on Learning Representations (ICLR)*, 2021.

[34] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 481–497. Springer, 2019.

[35] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[36] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep and convolutional? *arXiv preprint arXiv:1603.05691*, 2016.

[37] H. Wang, H. Zhao, X. Li, and X. Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *International Joint Conferences on Artificial Intelligence Organization (IJCAI)*, pages 2769–2775, 2018.

[38] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

[39] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.

[40] Z. Zhang, G. Ning, and Z. He. Knowledge projection for deep neural networks. *arXiv preprint arXiv:1710.09505*, 2017.

[41] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018.

[42] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017.

7 Reproducibility Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.

   (b) Did you describe the limitations of your work? [Yes] See Section 6.

   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work has no potential negative social impact.

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.

   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3.

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [No] As our work is ongoing, the code, data, and instructions have not been released yet. We plan to release them along with our main paper.

   (b) Did you include the raw results of running the given instructions on the given code and data? [N/A] The Figure 2, 3 and the Table 1 that are generated based on the raw results are included.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [N/A]

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [N/A]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] Our experiments were conducted on publicly available datasets (Cifar10) and we specified training details (e.g., search spaces, the number of training batches, etc.).

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] We compared different methods exactly on the same search space, dataset, set of architecture candidates, and hyperparameters. See section 4.

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] See section 4.

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes] We used the same evaluation protocol with a proxy score averaged for 20 training batches with regard to the architecture candidates we specified.

   (i) Did you compare performance over time? [No]

   (j) Did you perform multiple runs of your experiments and report random seeds? [No]

(k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We have not experimented repeatedly yet, but so far we have not had any error bars to report on.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [No] We did not use any other benchmarks for in-depth evaluations.

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We will include them in the main paper.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We used publicly available dataset (CIFAR10) and we cited all the creators who provided the code used to measure the baselines in **References**.

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]