Special Paper
방송공학회논문지 제26권 제7호, 2021년 12월 (JBE Vol. 26, No. 7, December 2021)
https://doi.org/10.5909/JBE.2020.26.7.855
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

Training-Free Hardware-Aware Neural Architecture Search with Reinforcement Learning

Linh Tam Tran[a)] and Sung-Ho Bae[a)‡]

Abstract

Neural Architecture Search (NAS) is cutting-edge technology in the machine learning community. NAS Without Training (NASWOT) recently has been proposed to tackle the high demand of computational resources in NAS by leveraging some indicators to predict the performance of architectures before training. The advantage of these indicators is that they do not require any training. Thus, NASWOT reduces the searching time and computational cost significantly. However, NASWOT only considers high-performing networks which does not guarantee a fast inference speed on hardware devices. In this paper, we propose a multi objectives reward function, which considers the network's latency and the predicted performance, and incorporate it into the Reinforcement Learning approach to search for the best networks with low latency. Unlike other methods, which use FLOPs to measure the latency that does not reflect the actual latency, we obtain the network's latency from the hardware NAS bench. We conduct extensive experiments on NAS-Bench-201 using CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets, and show that the proposed method is capable of generating the best network under latency constrained without training subnetworks.

Keywords: Neural Architecture Search, Hardware-Aware Neural Architecture Search, NAS without Training

Ⅰ. Introduction

Deep Neural Networks (DNNs) have shown remarkable performance on various computer vision tasks such as image classification[1,2], object detection[3,4], and image segmentation[5]. However, designing a good DNN is time-consuming and requires a lot of expert knowledge. Thus, researchers shift their goal to designing algorithms that search for neural networks, leading to Neural Architecture Search (NAS). Although NAS can design good networks, it requires a huge amount of memory and time on GPUs. To make NAS applicable for real-world problems, NAS Without Training (NASWOT) is introduced[6]. The motivation behind NASWOT is to leverage an indicator to predict the performance of networks before training. Since this

a) Department of Computer Science and Engineering, Kyung Hee University
‡ Corresponding Author : Sung Ho Bae
E-mail: shbae@khu.ac.kr
Tel: +82-31-201-2593
ORCID: https://orcid.org/0000-0002-3389-1159

※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2018R1C1B3008159). Also, this research was a result of a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT' and NIPA.
· Manuscript received October 25, 2021; Revised December 9, 2021; Accepted December 9, 2021.

process does not require any training, NASWOT can find the best network within minutes.

Hardware-aware Neural Architecture Search[7,8] can be considered as a sub-field of NAS where it focuses on hardware devices such as mobiles, FPGA, Raspberry Pi, and Edge GPU. Despite using different methods for searching, they share the same paradigm which is incorporating the actual latency to the loss function or the reward function. However, the major drawback of these methods is that they require a lot of computational resources which is unaffordable for real-world applications. For example, MnasNet[8] requires 91K GPU hours to complete the search phase. FBNet[7] takes 216 GPU hours to find the best network.

To solve this problem, this paper proposes a training-free hardware-aware NAS which leverages indicators to guide the search under latency constrained. We use the Reinforcement Learning (RL) approach to generate architectures which have high performance with low latency. We develop a new reward function which considers the network's latency during searching. We evaluate the performance of the proposed method using various training-free indicators on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets under NAS-Bench-201 search space. We conduct extensive experiments using three hardware devices, namely Edge GPU, Raspi 4, and Pixel 3, and show that using the proposed reward formulation, we can search for the best network under hardware constrained without any training.

II. Related works

In this section, we briefly discuss several works on NASWOT, hardware-aware NAS algorithms, and datasets for benchmarking NAS algorithms.

1. Neural architecture search without training

NASWOT[6] has been proposed recently to tackle the resource demand for conventional NAS approaches by predicting the performance of networks before training via some indicators. We adopt five training-free indicators used in previous works, namely Correlation of Jacobian (CJ), the Number of Linear Region (NLR), the Condition Number of Neural Tangent Kernel (CNNTK), Output Sensitivity (OS), and Fusion Indicator (FI).

Correlation of Jacobian (CJ): Joseph Mellor, et al[6] proposed a training-free indicator for NAS. They find that the performance of network is positively related to the correlation of Jacobian matrix with augmented input X. Let f be a neural network and X_n be the samples which are generated by applying Cutout[9] to X. The Jacobian matrix for X is calculated as:

$$J=\left(\frac{\partial f(X_1)}{\partial X_1},\frac{\partial f(X_2)}{\partial X_2},...,\frac{\partial f(X_n)}{\partial X_n}\right).$$

The score for CJ is defined as:

$$S_{CJ}=\sum_{i,j}1(0<(\sum_J)_{i,j}<\beta),$$

where β is a pre-defined threshold. The intuition behind CJ is that if the correlation of Jacobian for different input X is low, the network should have high performance and vice versa.

The Number of Linear Regions (NLR): It is well-known that deeper networks usually have higher performance than shallow ones. To understand which reasons, support this phenomenon, Razvan Pascanu et al[10] develop a theory for ReLU network. They have shown that deeper networks divide the input into regions which are larger than shallow ones, which is termed as the expressivity of neural networks. Huan Xiong et al[11] further extend the result for convolution neural networks. Let $\mathcal{N}$ be a ReLU CNN and θ be the parameters. The regions $R(.)$ for P is defined as:

$$R(P;\theta):=\{x^0\in R^{C\times H\times W}: z(x^0;\theta)\ \bullet\ P(z)>0, \forall z\in \mathrm{N}\},$$

where $z(.)$ is the pre-activation of a neuron and p is an activation pattern such that $P(z) \in \{1, -1\}$. The NLR score can be calculated as:

$$S_{\mathrm{NLR}} = \#\{R(P;\theta) : R(P;\theta) \neq \phi \text{ for some P}\}.$$

The Condition Number of Neural Tangent Kernel (CNNTK): Arthur Jacot et al[12] proposed a new tool to analyze the behavior of NN during training called neural tangent kernel. They have proved that using NTK, we can obtain the evolution of linearized NN without performing gradient descent:

$$\mu_t(X_{\mathrm{train}}) = (Id - e^{-\eta \hat{\Theta}_{\mathrm{train,train}^t}})\, Y_{\mathrm{train}}$$

where $\hat{\Theta}_{\mathrm{train,train}}$ is finite width NTK, η is the learning rate, X_{train} and Y_{train} are the input and target from the training set. Lechao Xiao et al[13] study the trainability of NNs via NTK by analyzing the eigenvalues of $\hat{\Theta}_{\mathrm{train,train}}$. The above equation can be expressed in terms of spectrum of $\hat{\Theta}_{\mathrm{train,train}}$ as follows:

$$\tilde{\mu}_t(X_{\mathrm{train}})_i = (ID - e^{-\eta \lambda_i t})\, \tilde{Y}_{\mathrm{train}},$$

where λ_i is the eigenvalue of $\hat{\Theta}_{\mathrm{train,train}}$. Denoting λ_0 and λ_m as the maximum and minimum eigenvalue of $\hat{\Theta}_{\mathrm{train,train}}$. Jaehoon Lee et al[14] have shown that the maximum learning rate scales as $2/\lambda_0$. Thus, the smallest eigenvalue converges exponentially at a rate of λ_m/λ_0 and the trainability of a network is defined as $k = \lambda_m/\lambda_0$. If k diverges, the network is untrainable. We invert k as the score for trainability so that higher score leads to better trainability:

$$S_{CNNTK} = \frac{1}{k}.$$

Output Sensitivity (OS): Mahsa Forouzesh et al[15] studies the generalization of NNs using the sensitivity of output. Let x be the input of a neural network f_θ with parameters θ. The error corresponding to input x with random noise ε sampled from a normal distribution is defined as:

$$err_y = f_\theta(x + \varepsilon) - f_\theta(x).$$

The output sensitivity is measured by the variance of the err_y averaged over M samples and is expressed as:

$$S_{os} = Var(\overline{err_y}).$$

Fusion Indicator (FI): Tran et al[16] proposed a feature fusion indicator for training-free NAS which harmonizes CJ, NLR, CNNTK, and OS in a weighted sum manner. They have shown that the FI outperforms standalone indicator by a large margin on several image classification tasks (e.g., CIFAR-10, CIFAR-100, ImageNet-16-120) using NAS-Bench-201 search space. By considering multiple aspects of a network such as trainability, expressivity, output sensitivity…, FI can rank the network properly.

2. Hardware-aware neural architecture search

NAS methods can find the best network on computer vision tasks[1,2,3,4,5]. However, it is difficult to deploy these networks on devices which have limited resources such as mobiles or embedded boards (e.g., Raspberry Pi). Moreover, the latency when running these networks on hardware devices tends to be high. Thus, hardware-aware NAS methods have been introduced to tackle this problem[7,8]. By incorporating the latency loss into the total loss function, it is possible to search for high-performing architectures under latency constrained. However, these methods require training subnetworks or a supernet, which takes a huge amount of time (e.g., FBNet[7] takes 216 GPU hours). We suggest performing hardware-aware NAS in a training-free manner by utilizing some indicators together with

the network's latency.

3. Benchmarking NAS algorithms

NAS usually consists of a search space which has a lot of candidate networks and a search algorithm. Different NAS methods use different search spaces and hyper-parameters for training. Thus, it is difficult to compare NAS methods because executing an algorithm on a poor search space may degrade its performance. To fairly compare NAS, NAS-Bench-201[17] is introduced. NAS-Bench-201 consists of 15,626 architectures which are trained using the same hyper-parameters on CIFAR-10, CIFAR-100, and ImageNet-16-120. NAS-Bench-201 provides the train, validation, and test accuracies for these datasets.

To compare hardware-aware NAS methods, HW-NAS-Bench[18] is proposed. HW-NAS-Bench provides the latency and energy consumption on six hardware devices, namely Edge GPU, Raspi 4, Edge TPU, Pixel 3, ASIC-Eyeriss, and FPGA. The search space used in this dataset is NAS-Bench-201 and FBNet search space. Researchers now can focus only on designing the algorithms targeted for hardware devices without measuring the network's latency as it can be obtained from HW-NAS-Bench.

Ⅲ. Proposed Training-free Hardware-Aware Neural Architecture Search

1. Multiple objectives reward formulation

We use the Reinforcement Learning approach to search for the best network under hardware constrained. The objective of our problem is to maximize the predicted score while keeping the latency under an expected latency. We use a product of predict score and a custom latency reward to update the controller. Let m be a network sampled from a search space $\mathcal{A}$ and T is the expected latency. The reward for latency is defined as:

$$R_{LAT}(m) = \begin{cases} \frac{T}{LAT(m)}, & if LAT(m) > T \\ g(LAT(m)), & otherwise \end{cases}$$

where $LAT(m)$ is the latency of network m and $g(.)$ is a linear reward function which gives higher reward for networks that have the latency closes to the expected latency T and vice versa. Thus, $g(.)$ can be written as:

$$g(f) = \alpha + \frac{(1-a)*f}{T}$$

where α is the hyper-parameter which controls how much reward is given. The final reward $R(m)$ to update the controller is defined as:

$$R(m) = Score(m) * R_{LAT}(m)$$

where $Score(m)$ can be obtained from training-free indicators.

2. Search algorithm

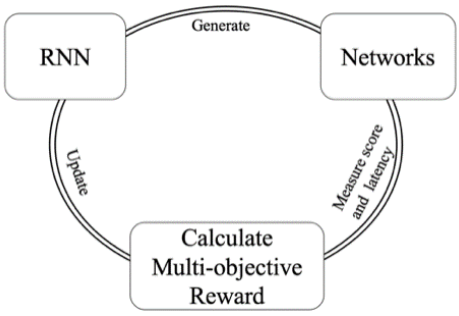

Fig. 1. Overview of the RL-based search algorithm. A controller RNN learns to generate network which has high score with latency below expected latency

We adopt RL for our training-free hardware-aware NAS as it is convenient and easy to implement. We train an RNN controller to generate network operations such as 3x3

convolution, 1x1 convolution, 3x3 average pooling, skip-connect, and zeroize operation. Our goal is to find the networks which have a high score and low latency. To achieve this, we aim to maximize the reward:

$$J = E_{p(a_{1:T};\theta)}[R(m)]$$

where m is the network sampled by the controller with action $a_{1:T}$, and R is the reward of network m.

Ⅳ. Experiments

We conduct the experiment on NAS-Bench-201 search space using 5 training-free indicators: CJ, NLR, CNNTK, OS, and FI under three hardware devices, namely Edge GPU, Raspi 4, and Pixel 3. The latency for these devices is obtained from HW-NAS-Bench. We set the expected latency for Edge GPU, Raspi 4, and Pixel 3 to 6ms, 40ms, and 20ms for CIFAR-10 and CIFAR-100 datasets, respectively. On ImageNet-16-120, the expected latency is 6ms, 9ms, and 6ms for these hardware devices. It is noted that the highest latency for Edge GPU, Raspi 4, and Pixel 4 is 11.46ms, 96.7ms, and 43.88ms on CIFAR-100 and 11.52ms, 17.93ms, and 12.62ms on ImageNet-16-120, respectively.

NAS-Bench-201: all networks in this dataset are trained with Nesterov momentum SGD optimizer for 200 epochs. The initial learning rate is 0.1, decaying with cosine annealing. The batch size is 256.

Training setup: The controller we used has one RNN layer with 100 hidden units. We train the controller with Adam optimizer for 500 epochs. The learning rate is set to 5e-4. The α is set to 0.9.

Architecture selection: once the training of the controller is finished, we first omit the architectures which have higher latency than the expected latency. Then, we select the candidate that has the highest predicted score as the final architecture for evaluation.

Result: We run the experiment 50 times with different random seeds and report the average test accuracy in Table 1. From Table 1, we can see that FI outperforms standalone indicator in terms of test accuracy on CIFAR-10 and CIFAR-100. This is natural since FI has a higher rank correlation than other indicators as demonstrated by Tran et al[15]. On CIFAR-10, the proposed method can find the network which has 93.76% test accuracy with a latency of 5.49 ms on Edge GPU using Fusion Indicator.

Table 1. Performance comparison between different training-free indicators under hardware constrained

Dataset	Indicator	Edge GPU		Raspi 4		Pixel 3	
		Test Acc (%)	Latency (ms)	Test Acc (%)	Latency (ms)	Test Acc (%)	Latency (ms)
CIFAR-10	CJ	93.56	4.83	93.07	28.95	93.67	15.74
	NLR	91.75	5.36	90.27	17.55	90.47	7.35
	CNNTK	93.47	4.67	93.41	28.92	93.49	14.34
	OS	93.32	5.44	92.65	34.00	93.35	18.00
	FI	93.76	5.49	93.42	28.06	93.8	16.20
CIFAR-100	CJ	70.65	4.64	70.27	30.00	71.07	15.89
	NLR	68.82	5.44	66.35	23.40	66.94	8.72
	CNNTK	70.70	4.56	70.31	32.69	70.53	16.32
	OS	69.32	5.57	68.63	36.37	69.38	19.19
	FI	72.00	5.67	70.91	29.06	72.11	17.73
ImageNet-16-120	CJ	39.77	4.36	40.49	5.34	40.57	3.16
	NLR	40.77	5.42	42.76	6.31	43.5	3.73
	CNNTK	42.99	4.90	42.54	5.81	42.10	3.96
	OS	41.26	5.69	42.83	8.64	44.71	5.32
	FI	44.49	5.37	44.56	7.77	44.57	4.44

Table 2. Performance comparison between different reward formulations of FI on NAS-Bench-201

Dataset	Edge GPU		Raspi 4		Pixel 3	
	Without R_{LAT}	With R_{LAT}	Without R_{LAT}	With R_{LAT}	Without R_{LAT}	With R_{LAT}
CIFAR-10	93.44	93.76	92.87	93.42	93.27	93.8
CIFAR-100	70.09	72.00	69.87	70.91	70.45	72.11
ImageNet-16-120	44.14	44.49	44.44	44.56	44.52	44.57

On Pixel 3, the best network has 93.8% test accuracy and 16.20 ms latency on average. On CIFAR-100, FI outperforms other indicators in terms of test accuracy for all hardware devices. Specifically, FI achieves 72.00%, 70.91%, and 72.11% test accuracy for Edge GPU, Raspi 4, and Pixel 3. The best standalone indicator obtains 70.7%, 70.32%, and 71.07 for these hardware devices, which is lower than FI. On ImagetNet-16-120, the best network on Edge GPU, Raspi 4 are obtained by FI while on Pixel 3, OS indicator shows the best performance.

To demonstrate the effectiveness of our proposed reward function, we compare the performance of the RL approach without using the reward for latency (i.e., only maximizing the score).

We run the experiment 50 times with FI as this indicator has a good rank correlation so that it can reflect the result properly. We summarize the result in Table 2.

As shown in Table 2, the average test accuracy is much lower when we only maximize the score for all datasets. This is because the controller does not explore the candidates that have a high score with low latency. By contrast, the proposed reward, which takes the product of score and latency's reward, allows the controller to learn to generate networks which have low latency but high score. The result from Table 2 confirms the effectiveness of the proposed method.

Ⅴ. Conclusion

In this paper, we propose a new method for hardware-aware NAS in a training-free manner. Previous works require a lot of GPU resources which is extremely expensive. To reduce the search cost, we suggest utilizing training-free indicators which can estimate the performance of networks before training for hardware-aware NAS. By incorporating the proposed reward formulation into Reinforcement Learning approach, we can find the best networks with low latency on various hardware devices. Furthermore, this work also enables a new research direction in the future which performs hardware-aware NAS without training.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385, 2015.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv: 1704.04861, 2017.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pp. 779-788, 2016.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "SSD: Single Shot MultiBox Detector," in European Conference on Computer Vision, pp 21-37, 2016.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, "Mask-RCNN," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2961-2969, 2017.

[6] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley, "Neural Architecture Search without Training," under review at https://openreview.net/forum?id=g4E6SAAvACo.

[7] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer, "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10734-10742, 2019.

[8] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le, "MnasNet: Platform-Aware Neural Architecture Search for Mobile," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2820-2828, 2019.

[9] Terrance DeVries, Graham W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout", arXiv: 1708.04552, 2017.

[10] Razvan Pascanu, Guido F. Montufar, and Yoshua Bengio, "On the number of inference regions of deep feed forward networks with piece-wise linear activations," CoRR, arXiv:1312.6098, 2014.

[11] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao, "On the Number of Linear Regions of Convolutional Neural Networks," in Proceedings of the 37th International Conference on Machine Learning, PMLR 119:10514-10523, 2020.

[12] Arthur Jacot, Franck Gabriel, and Clement Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," in Advances in Neural Information Processing Systems 31, 2018.

[13] Lechao Xiao, Jeffrey Pennington, and Samuel S. Schoenholz, "Disentangling Trainability and Generalization in Deep Neural Networks," in Proceedings of the 37th International Conference on Machine Learning, pp. 10462—10472, 2020.

[14] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington, "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent," in Advances in Neural Information Processing Systems 32, 2019.

[15] Mahsa Forouzesh, Farnood Salehi, and Patrick Thiran, "Generalization Comparison of Deep Neural Networks via Output Sensitivity," in International Conference on Pattern Recognition 25th, pp. 7411-7418, 2020.

[16] L.T. Tran, M. S. Ali and S. -H. Bae, "A Feature Fusion Based Indicator for Training-Free Neural Architecture Search," in IEEE Access, vol. 9, pp. 133914-133923, 2021.

[17] Xuanyi Dong and Yi Yang, "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search," in International Conference on Learning Representations, 2020.

[18] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin, "HW-NAS-Bench: Hardware-Aware Neural Architecture Search Benchmark," in International Conference on Learning Representations, 2021.

Introduction Authors

Linh Tam Tran

- Sep. 2020 ~ : He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, Yongin, South Korea.
- Mar. 2016 ~ Feb. 2018 : He received the the M.S. degree from Hongik University, Seoul, South Korea
- Sep. 2009 ~ Feb. 2014 : He received the bachelor's degree from the Department of Computer Science and Engineering, Ho Chi Minh City University of Technology, Vietnam
- ORCID : https://orcid.org/0000-0002-9699-1747
- Interests : Neural Architecture Search for practical applications

Sung-Ho Bae

- Sep. 2017 ~ : He has been an Assistant Professor with Department of Computer Science and Engineering, Kyung Hee University, Yongin, South Korea.
- Jul. 2016 ~ Aug. 2017 : He was a Postdoctoral Associate with Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), MA, USA.
- Feb. 2011 ~ Aug. 2016 : He received M.S. and Ph. D. degrees with Department of Electrical and Electronic Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea.
- May. 2004 ~ Feb. 2011 : He received the B.S. degree with he Department of Computer Science and Engineering/ Electronics from Kyung Hee University, South Korea.
- ORCID : https://orcid.org/0000-0002-3389-1159
- Interests : Adversarial Attack, Model Compression, Semi-supervised Learning and Data Augmentation, Explainable DNN, Neural Architecture Search for Hardware Implementation, International Standardization