Mixed-precision Supernet Training for Quantized Neural Architecture Search

Jing-Teng Hwang
Dept. of Electrical Engineering,
National Taiwan University

Tzi-Dar Chiueh
Graduate Institute of Electrical Engineering,
National Taiwan University

***Abstract*—Recently, demands for edge computing have led to the development of efficient and compact deep neural networks (DNNs). Neural architecture search (NAS) and model quantization are proposed under this background. Several works have tried to combine NAS and quantization, but the searched architectures usually require additional re-training and their hardware/software performances still have room for improvement. In this work, we propose a mixed-precision supernet with quantization bit width search space. To evenly train the sub-nets, we propose a shared step size for different kernel sizes. To boost the accuracy of the quantized sub-nets, we design a step size and offset re-tune process before evaluating the sampled sub-net. Preliminary results on Imagenet show the proposed supernet generates sub-nets that have superior performance compared with other SOTA methods. The best sub-net achieves 72.31% top-1 accuracy with roughly half of the bit operations compared to APQ [25]. Our codes are available at https://github.com/JeterHwang/ofa2.0**

***Index Terms*—NAS, One-shot, Quantization, QNAS**

1. Introduction

In the past decades, Deep Neural Networks (DNNs) have achieved significant success in many fields including computer vision, natural language processing, and machine translation. Recently, edge computing has imposed strict hardware constraints on the development of DNNs, leading to a surge in demands for compact and efficient models. To this end, NAS [3, 4, 7, 16, 26, 31] is proposed to search for the optimal architecture under certain hardware constraints. Further combined with model quantization, NAS algorithms extend their search spaces to quantization schemes, becoming so-called Quantized NAS (QNAS) [1, 11, 20, 23, 25, 27]. Although previous QNAS algorithms succeed in lowering the latency and the parameter size of DNNs, they still face the following issues:

1) The accuracy drop when quantized to low bit width (e.g. ≤ 4 bit) is so large that there is apparently room for improvement.
2) The searched model usually needs to be retrained to restore its accuracy, which can be time-consuming for different deployment scenarios.

Therefore, we propose a novel mixed-precision QNAS search space. The proposed supernet includes elastic block depth, elastic expansion ratio, elastic kernel size, and elastic quantization bit width. Furthermore, the sub-nets sampled from the trained supernet can be directly evaluated without re-training. The main contributions of this work are the followings:

1) We propose a novel elastic bit width architecture for the mixed-precision supernet, which stabilizes the training process and requires no sub-net re-training.
2) The best sub-net achieves 72.31% top-1 accuracy on Imagenet with merely 6.17G bit operations (bitOps), outperforming the existing QNAS algorithms.

The rest of this article is organized as follows. Related works are discussed in section 2. Proposed elastic quantization architecture, supernet architecture, quantization-aware training, and validation process are provided in section 3. The implementation details and the evaluation results are presented in section 4 and section 5, concluding the article.

2. Related Work

Hyper parameter Optimization (HPO). Before ML design automation comes into being, DNN models are designed by scientists by trial and error, which is super ineffective and inevitably produces sub-optimal results. Therefore, some early works focus on HPO, aiming to finetune the hyperparameters either in the training process or the model architecture. For instance, [10, 15, 18] succeeds in the search for training hyperparameters, such as batch size, learning rate, and the number of training epochs by Bayesian Optimization with Hyper Band (BOHB). Several ML frameworks [8, 13, 29] also support HPO, allowing users to tune their training hyperparameters.

Neural Architecture Search (NAS). As training HPO becomes possible, some may think: *Can model hyperparameter, or equivalently model architecture, be optimized using the same techniques?* However, model hyperparameters are much more than that of training. The operation type, the input/output dimension of each layer can have its own hyperparameters, not to mention the fact that the number of layers is also scalable. Therefore, new methods must be proposed to deal with the gigantic search space in NAS. Initially, scientists perform NAS with reinforcement learning (RL) based NAS algorithms [30, 31]. The algorithm first uses a controller to sample architectures in the search space and evaluates the sampled model after certain epochs of training. Afterward, the evaluation result is fed back to the controller to update its sampling policy.

Differentiable NAS (DNAS). Since training both the sub-nets and the controller is super time-consuming, some propose training a ”supernet”, which encompasses all possible model architectures, and searching for the optimal sub-net directly from the supernet. The most distinguished algorithm in this category is DNAS. Starting from DARTS [16], where the architecture parameters are co-optimized with model parameters by gradient descent, plenty of DNAS algorithms are proposed, such as FBnet [7, 26] and ProxylessNAS [4]. They consider each output feature map as a ”node” and connect the nodes with several paths, where each path represents an operation. During inference, each node is the weighted sum of all results from the path connected to it, and the weights, or so-called architectural parameters, are obtained by the Gumbel Softmax of a set of trainable parameters. During backpropagation, the model parameters are updated first, and then the architectural parameters. After training the supernet, the final architecture is decided by selecting the operation between two nodes with the maximal weight. In addition, FBnet and ProxylessNAS incorporate hardware-aware loss in the gradient, so the algorithms not only search for architectures that achieve the best accuracy but also search for architectures that have the lowest hardware cost.

One-shot NAS. Although DNAS algorithms are successful in image classification, they have shortcomings that make them superseded by other methods. First of all, including all operations in the supernet makes it too heavy to scale to large search spaces. Furthermore, selecting the final architecture based on the architectural parameters does not always generate the optimal network [24]. Last but not least, it is hard to incorporate hardware costs into the differentiable loss, and we cannot set a constraint on the final networks, leading to difficulty in deploying the model on a resource-constrained platform. To deal with the exploding supernet, [19] proposes the weight-sharing method, where all operation choices on an edge share the same model parameters, thus increasing the scalability. To avoid training the whole supernet, [11] proposes to train a single path from input to output in one iteration, letting the rest of the parameters untouched. To select the best sub-net meeting the hardware and software requirements, [17] proposes a multi-objective genetic algorithm (GA) to search for the best sub-nets in a sub-net pool. Combined, they evolve into a novel algorithm, one-shot NAS [3, 21, 22]. Once-for-All (OFA) network [3] is the paradigm of many one-shot methods. It first trains an elastic supernet and then evaluates some sub-nets to train an accuracy predictor. Finally, it performs GA to search for the optimal network under certain hardware constraints, where the sampled sub-nets are evaluated by the accuracy predictor and the hardware metric calculators. Most importantly, the final network doesn’t need retraining owing to the outstanding supernet training technique, progressive shrinking. Due to its prominent design, our NAS algorithm is mainly based on the OFA network.

Quantized NAS (QNAS). With the advent of edge computing, slimmable ML models become a hot topic. Model compression methods, such as channel pruning and quantization are proposed to reduce the parameter size and inference latency on the edge devices. Among the model compression schemes, channel pruning has been implemented in several NAS algorithms [3, 28], while quantization schemes are still left to be explored. In quantization, the key hyperparameter to be searched is the quantization bit width. The smaller the bit width, the more noise it will bring to the model output. Hence, QNAS algorithms search for the best neural architecture along with the optimal bit width for model weights and feature maps. OQA [20] proposes a fixed-precision QNAS algorithm, where the architecture is searched under a certain bit width. HAQ [23], DNAS [27], SPOS [11],and APQ [25] search for a mixed-precision network, where the bit width of each operation is searched independently. Among the mixed-precision QNAS algorithms, only SPOS trains a mixed-precision supernet using the PACT [5] quantization scheme, while others perform proxy-based methods to evaluate the performance of the mixed-precision network, which tends to miscalculate the actual performance of mixed-precision networks. Furthermore, recent NAS networks are usually searched on MobilenetV3 [12] search space since Mobilenets are already optimized under NAS and thus intrinsically more compact. Besides, LSQ [9] and LSQ+ [2] are proven to have better quantization effects than PACT. These factors motivate us to design a novel mixed-precision supernet and its training recipe for QNAS.

3. Methods

This section introduces the proposed algorithm. We will give a detailed description of the design of the elastic quantization scheme and supernet architecture, along with the training and validation process.

3.1. Proposed Elastic Quantization Bit Width

Inspired by OFA [3] network, whose supernet can have elastic channels, elastic kernel size, and elastic block depth, we design our supernet to have elastic quantization bit width. More specifically, we quantize the model weights using the LSQ quantization scheme for each output channel, which is provided in eq. (1),

$$\begin{aligned} \bar{x} &= \lfloor clip(\frac{x}{s_k}, n, p) \rceil \\ \hat{x} &= \bar{x} \times s_k \end{aligned} \tag{1}$$

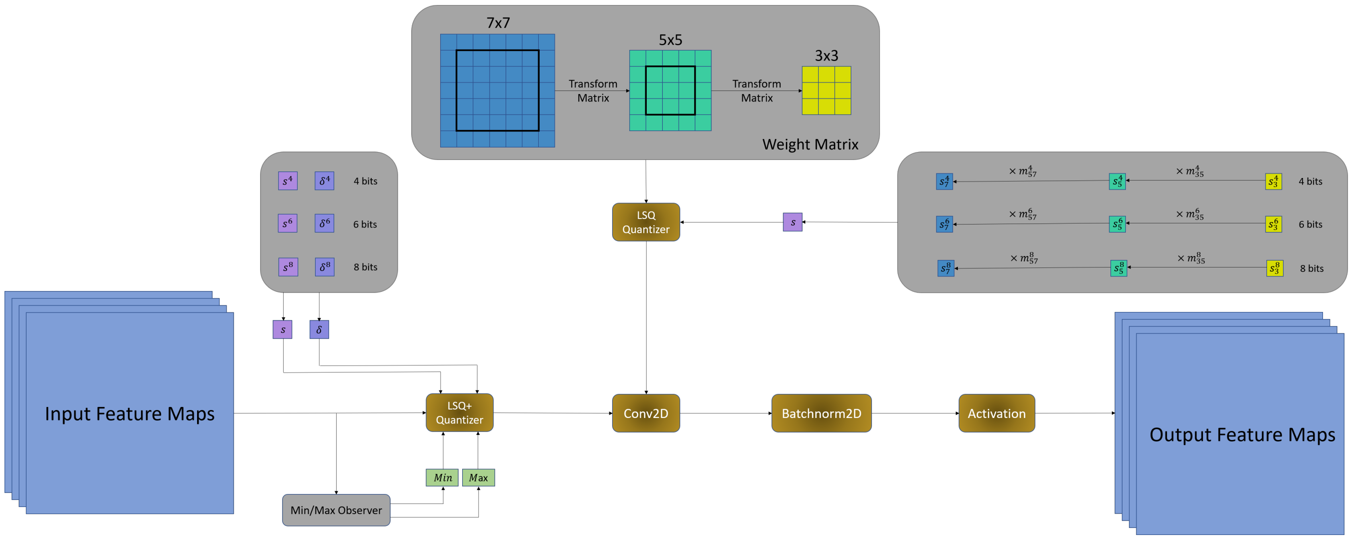

Figure 1: Overall architecture of the elastic quantization bit width. The input feature maps are quantized by LSQ+ quantizer, while the weight matrix is quantized by LSQ quantizer as stated in section 3.1. After being quantized, they are fed into 2D convolution, followed by 2D batch normalization and activation (e.g. relu, h_swish), and then output the result.

where x is the weight to be quantized, k is the convolution kernel size, $n = -2^{bit-1}$ is the minimum quantized value, $p = 2^{bit-1} - 1$ is the maximum quantized value, s_k is the step size for $k \times k$ convolution, and $\hat{x}$ is the quantized weight.

Inspired by the shared step size proposed in OQA [20], we design a relation between the step sizes for different kernel sizes. Suppose the elastic kernel sizes are chosen from $[3, 5, 7]$, the step sizes for each kernel size are given by eq. (2),

$$\begin{aligned} s_3 &= s \\ s_5 &= s_3 \times m_{35} = s \times m_{35} \\ s_7 &= s_5 \times m_{57} = s \times m_{35} \times m_{57} \end{aligned} \tag{2}$$

where s is the base step size, m_{ij} is the trainable coefficient between s_i and s_j. Need to notice that our method is different from OQA in that OQA uses fixed-precision quantization while ours uses mixed-precision quantization. Furthermore, OQA shares the step sizes in a layer, where the step sizes of the weight s_w and the step sizes of the activation s_a are the same, while our method only shares a base step size between different kernel sizes. When training the supernet, both s and m_{ij} are trained, but s will be trained more since whichever kernel size is sampled, the shared parameter s will always be updated. This is reasonable because, in our observation, small convolution kernels tend to require more iterations of fine-tuning to find the optimal step size.

On the other hand, we quantize the activation with the LSQ+ quantization scheme for each tensor, which is provided in eq. (3),

$$\begin{aligned} \bar{x} &= \lfloor clip(\frac{x - \beta}{\gamma}, n, p) \rceil \\ \hat{x} &= \bar{x} \times \gamma + \beta \end{aligned} \tag{3}$$

where $n, p, \bar{x}, \hat{x}$ have the same definition as LSQ. Here we introduce two new variables β and γ as the offset and the scale respectively. They are calculated by eq. (4).

$$\begin{aligned} \gamma &= X_{min} + \delta \\ \beta &= \frac{X_{max} - X_{min}}{n - p} \times s \end{aligned} \tag{4}$$

X_{max} and X_{min} are the maximal and minimal values in a tensor. During training, these two values are calculated for each batch of training data, while during validation, they are fixed to certain values (the calculation process is detailed in section 3.4). δ and s are trainable parameters that are learned to counteract the bias from different batches of data. The detailed architecture of elastic bit width is illustrated in fig. 1.

3.2. Supernet Architecture

We build our supernet following the OFAnet [3], featuring elastic expansion ratio (possible choices: [3,4,6]), elastic convolution kernel size (possible choices: [3,5,7]), and elastic block depth (possible choices: [2,3,4]) for each of the 5 building blocks. Plus the elastic bit width, which also has 3 possible choices, our supernet has roughly $((3\times3\times3)^2+(3\times3\times3)^3+(3\times3\times3)^4)^5 \approx 5\times10^{28}$ sub-nets. Some studies show that quantizing the head and the tail of the model will severely degrade overall accuracy, so we only quantize the 5 building blocks in the middle, leaving the first inverted residual block, the final 2 convolution layers, and 1 fully connected layer unquantized. For simplicity, we also leave the Squeeze-and-Excitation (SE) layer unquantized.

3.3. Quantization Aware Training (QAT)

Following the training scheme used in one-shot NAS [3, 11, 14, 22], we sample one sub-net (i.e. one single path from input to output) for one batch of training data. Different from OFAnet, adopting the progressive shrinking training technique, we open all possible choices for elastic properties throughout the whole training process, which can save a large amount of supernet training time.

3.4. Validation

After training the supernet, we sample some sun-nets and evaluate them on the validation data. We use batch statistics for the running mean and running variance of batch normalization as well as the maximal and minimal value for activation quantization during training, but these values are not available during inference. Therefore, we forward 2000 training data on the sampled sub-nets, averaging the four values over all calibration data, and fixing them throughout the inference process. Our experiments find that the Min/Max re-tune for quantized sub-nets is of critical importance for obtaining high validation accuracy.

Apart from the validation accuracy, we measure the hardware performance, such as the parameter size and the bitOps, of the sampled sub-nets. The number of bitOps in a layer is calculated by eq. (5),

$$BitOps = (\#ofmultiplication) \times bit_w \times bit_a \quad (5)$$

where bit_w and bit_a are the quantization bit width of the weight and the activation respectively. On the other hand, the parameter size is calculated by multiplying bit_w and the number of elements in the weight matrix.

4. Implementation Details

Our design is implemented with Pytorch, with some weights loaded from pre-trained OFAnet [3]. The model is trained and evaluated on Imagenet [6]. However, due to a lack of computing resources, the supernet is only trained on half of the training data to save time. What's worse, due to the same reason, the supernet is only trained for 12 epochs with batch size 48 on 2 Nvidia GTX 1080Ti GPUs, which is far less than other NAS algorithms. We use SGD optimizer with Nesterov momentum 0.9 and weight decay $3e^{-5}$. The initial learning rate is $5e^{-3}$, and we use a cosine learning rate scheduler for learning rate decay. We train our supernet for 24 hours and directly evaluate sampled sub-nets **without** re-training.

5. Result

Due to limited time, we randomly sample 3 sub-nets for validation. The configurations of these sub-nets are shown in table 1. Among the 3 sub-nets, the first one with W32A32 is the baseline without quantization, while the other two are quantized. One can see that the second sub-net has better post-QAT accuracy in that its activation is not quantized, while the third one has fewer bitOps for its activation is quantized to 8 bits.

TABLE 1: Specifications of sampled sub-nets

D[1]	E[1]	K[1]	Q[1,2]	Before[3]	After[3]	bitOps(G)
2	6	7	W32A32	74.12%	-	389
2	6	7	W4A32	68.62%	**73.09%**	24.68
2	6	7	W4A8	59.03%	72.31%	**6.17**

[1] D, E, K, Q represents block depth, expansion ratio, kernel size, and quantization bit width. **All search blocks apply to the settings.**
[2] WpAq means the weight and activation are quantized to p bits and q bits.
[3] The top-1 accuracy before/after training the supernet

Though we don't implement the evolutionary search algorithm to exhaustively find the optimal sub-nets as many one-shot NAS algorithms do, Preliminary result shows the randomly sampled sub-nets have comparable accuracy to many SOTA QNAS algorithms under the same hardware constraint. Fig. 2 compares the existing mixed-precision QNAS algorithms with our method. As one can see, the sub-net with 72.31% top-1 accuracy and 6.17 BitOps(G) outperforms other QNAS sub-nets by having fewer bitOps under the same top-1 accuracy. Compare to the SOTA QNAS algorithm, APQ, our sub-net achieves the same accuracy with roughly half of the BitOps.

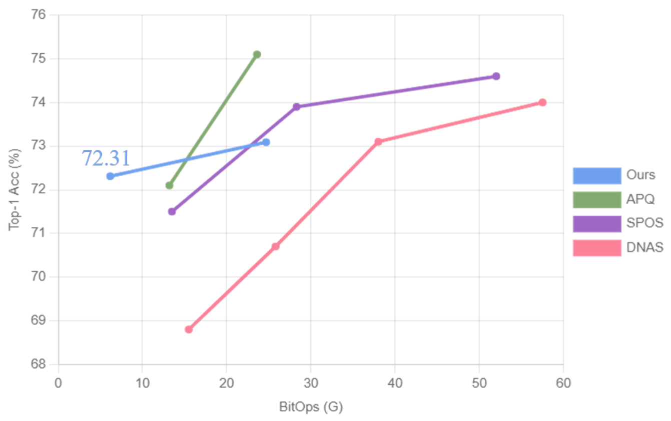

Figure 2: The comparison of our method with other SOTA QNAS algorithms. Note we just randomly sample 2 sub-nets instead of performing evolutionary search to search for optimal sub-nets.

6. Conclusion

In this paper, we present a novel quantized supernet architecture that searches for quantized sub-nets without additional re-training costs. In addition, a scale/offset re-tune scheme is proposed, which significantly boosts the accuracy of sampled sub-nets. Comparison with other SOTA mixed-precision QNAS algorithms reveals the superiority of our supernet architecture along with the training techniques.

7. Future Work

Due to limited time, we didn't implement the evolutionary search algorithm to exhaustively search for optimal sub-nets. With a proper search algorithm, we believe that the proposed supernet can generate sub-nets far exceed those presented in this work.

References

[1] Haoping Bai, Meng Cao, Ping Huang, and Jiulong Shan. Batchquant: Quantized-for-all architecture search with robust quantizer. *Advances in Neural Information Processing Systems*, 34:1074–1085, 2021.

[2] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020.

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

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[8] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[9] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[10] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR, 2018.

[11] 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 *European conference on computer vision*, pages 544–560. Springer, 2020.

[12] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[13] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019.

[14] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[15] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Joshua Marben, Philipp Müller, and Frank Hutter. Boah: A tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters. *arXiv preprint arXiv:1908.06756*, 2019.

[16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[17] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference*, pages 419–427, 2019.

[18] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *Workshop on automatic machine learning*, pages 58–65. PMLR, 2016.

[19] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[20] Mingzhu Shen, Feng Liang, Ruihao Gong, Yuhang Li, Chuming Li, Chen Lin, Fengwei Yu, Junjie Yan, and Wanli Ouyang. Once quantization-aware training: High performance extremely low-bit architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5340–5349, 2021.

[21] Dilin Wang, Chengyue Gong, Meng Li, Qiang Liu, and Vikas Chandra. Alphanet: improved training of supernets with alpha-divergence. In *International Conference on Machine Learning*, pages 10760–10771. PMLR, 2021.

[22] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6427, 2021.

[23] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

[24] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021.

[25] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2078–2087, 2020.

[26] 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*, pages 10734–10742, 2019.

[27] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

[28] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019.

[29] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

[30] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[31] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.