QuantNAS: Quantization-aware Neural Architecture Search For Efficient Deployment On Mobile Device

Tianxiao Gao, Li Guo, Shanwei Zhao, Peihan Xu, Yukun Yang, Xionghao Liu,
Shihao Wang, Shiai Zhu, Dajiang Zhou
Ant Group
{juanji.gtx, li.gl, shanwei.zsw, yukun.yyk, peihan.xph, xionghao.lxh}@antgroup.com
{shihao.wsh, shiai.zsa, dajiang.dzj}@antgroup.com

Abstract

Deep convolutional networks are increasingly applied in mobile AI scenarios. To achieve efficient deployment, researchers combine neural architecture search (NAS) and quantization to find the best quantized architecture. However, existing methods overlook the on-device implementation of quantization. The searching result is usually sub-optimal or has limited latency reduction. To this end, we propose QuantNAS, a novel quantization-aware NAS based on a two-stage one-shot method. Different from the previous method, our method considers the on-device implementation of the quantized network and searches for the architecture from a fully quantized supernet. During training, we propose a batch-statistics-based strategy to alleviate the non-convergence problem. Besides, a scale predictor is proposed and is jointly trained with the supernet. During search, the scale predictor can provide optimal scale for different subnets without retraining. At different latency levels on Kirin 9000 mobile CPU, the proposed method achieves 1.53%-1.68% Top-1 accuracy improvement on ImageNet 1K dataset and 1.7% mAP improvement.

1. Introduction

Convolutional neural networks (CNNs) are widely used in mobile AI scenarios, serving thousands of users through applications. However, mobile devices usually have limited resources. To achieve efficient deployment on mobile devices, CNNs need to be carefully designed and then quantized [3, 7, 13, 19, 21, 23, 24, 29–31]. Both processes rely on experts' experience and manual tuning, which need unacceptable development cost.

The combination of neural architecture search (NAS) [6, 9, 10, 32, 36, 37] and quantization is an efficient solution to reduce the cost. As shown in Figure 1, existing methods [1, 4, 8, 14, 35, 38–40] can be categorized as NAS-then-quantize and Quantization-aware NAS. **NAS-then-quantize** methods retrain the best searched floating-point subnets into the quantized networks. However, the accuracies of floating-point models and quantized models are not always directly proportional. This not only brings additional training cost, but also results in *sub-optimal* performance. In contrast, **Quantization-aware NAS** directly searches the best quantized subnet from a quantized supernet and can obtain the optimal performance.

However, existing methods do not deliver expected latency reduction on real devices. As can be seen in Figure 2, the searched subnet from existing methods is not *fully quantized* during on-device deployment. When training the quantized networks, FakeQuant operators are employed to simulate the quantization. As illustrated in Figure 2, existing quantization-aware NAS methods only quantize operators with large FLOPs, such as convolutional (Conv) layers, while ignoring other parts such as batch normalization (BN) layers and pooling layers. This results in a *non-fully quantized* model, leading to latency overhead in both floating-point calculations and data conversion between floating-point and fixed-point during practical deployment.

To this end, we propose QuantNAS, a quantization-aware neural architecture search framework for efficient deployment on mobile device. As shown in Figure 1, the proposed method trains a fully quantized weight-sharing supernet and searches the subnets for deployment directly from the supernet without retraining. Searching on a quantized supernet ensures that our method finds end-to-end optimal structures. Besides, a fully quantized network offers 50.3% latency reduction compared with a floating-point network, which is much higher than the non-fully quantized network.

However, achieving fully quantization in NAS is challenging. As shown in Figure 2, in addition to quantizing all operators, it is also necessary to consider the optimizations during deployment, such as Conv-BN fusion. For efficient deployment, Conv-BN fusion simplifies to a convolutional

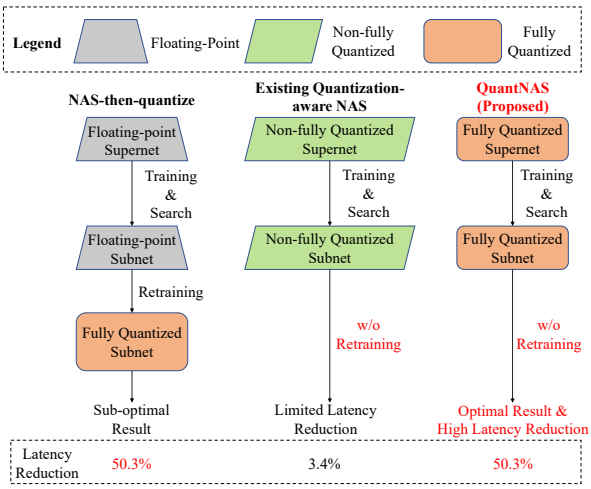

Figure 1. Overall framework of different combinations of NAS and quantization. The latency reduction is evaluate with Mobilenet-V2 [34] on Kirin 9000 mobile CPU.

layer by folding the original weights of Conv and the moving statistics of BN. Therefore, to simulate this behavior, we need to quantize the folded weights during supernet training. This will bring two challenges for quantization-aware NAS: 1) **Training a fully quantized supernet is difficult to converge**: The moving statistics of BN directly affect the value of the folded weights. For different subnets, the moving statistics vary greatly. It is impossible to update moving statistics separately for billions of subnets. If we only update one moving statistic of the supernet across all subnets, it will mislead the direction of supernet optimization. 2) **A shared scale is not optimal during search**: Scale value that maps the floating-point number into a fixed-point number is significant to the accuracy of a quantized network. Previous methods [1, 14, 35, 38, 39] usually learn a shared scale for all subnets during supernet training. However, during search phase of NAS, BN layers of sampled subnets need to be calibrated [6]. After calibration, the folded weights are changed correspondingly, and the shared scale is obviously not suitable. Therefore, searching a quantized subnet based on a mismatched scale will lead to non-optimal results.

In order to address the first challenge, we propose a batch-statistics-based strategy for stabilizing supernet training. Experimentally, we find that the moving statistics of the subnets are more similar to the batch statistics of the supernet than the moving statistics of the supernet. To make the fully quantized supernet convergent, the folded weights are computed with the batch statistics of the supernet instead of the moving statistics of the supernet during supernet training. In addition, to solve the non-optimal scale problem brought by learning a shared quantization scale, we propose a scale predictor for each quantized Conv-BN layer.

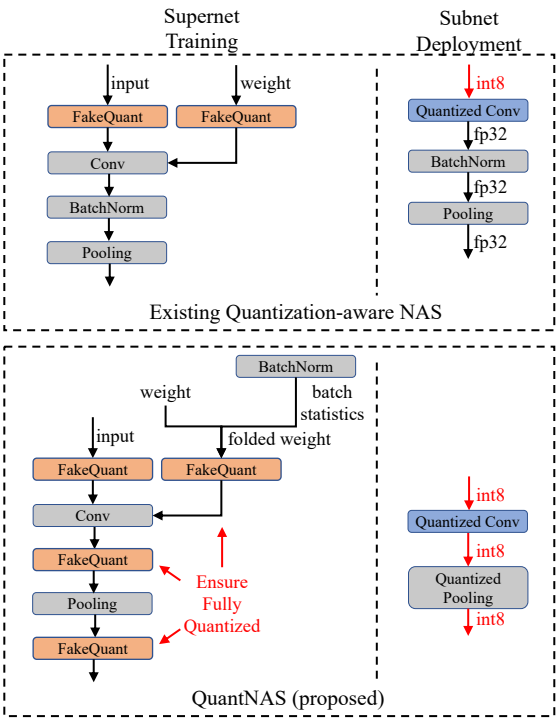

Figure 2. Comparison of training and inference between the previous works and out method.

The scale predictor is jointly trained with the supernet to obtain optimal scale for different subnets. During search, the scale of each subnet is calibrated adaptively with the scale predictor based on the calibrated data. Benefiting from the scale predictor, the accuracy of the subnets sampled from the supernet can be improved without retraining. With the aforementioned optimizations, QuantNAS achieves better performance compared with the existing quantization-aware NAS methods.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to consider on-device implementation in quantization-aware NAS. By ensuring the fully quantized supernet and subnets, the proposed QuantNAS achieves better latency-accuracy trade-off.
- We propose a batch-statistics-based training strategy and a scale predictor to achieve both stable training and optimal searching results.
- We conduct plentiful experiments to prove the effectiveness of our method. On Kirin 9000 mobile CPU, the quantized networks searched from our method achieve 1.53%-1.68% Top-1 accuracy improvement on ImageNet 1K dataset and 1.7% mAP improvement on COCO dataset, compared with the state-of-the-art methods.

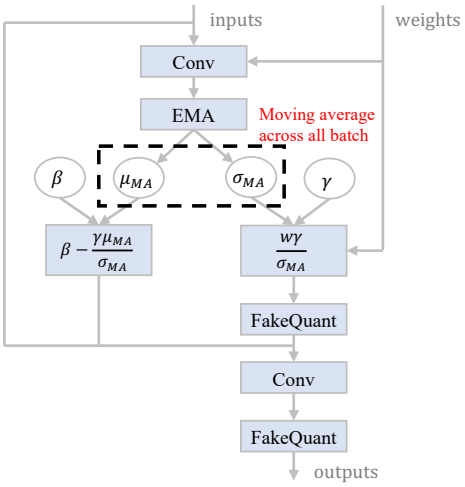

Figure 3. Quantization of Conv-BN layer in stand-alone networks [19]. Stand-alone strategy uses moving average statistics to compute the folded weights.

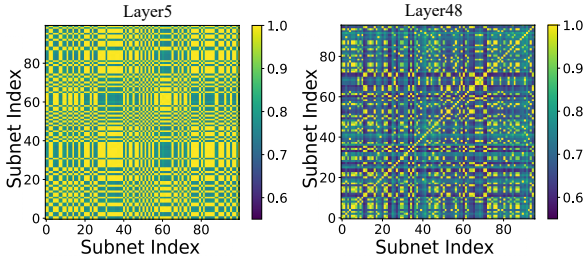

Figure 4. Cos similarity of moving std from different subnets. Moving statistics of different subnets are not similar, training the supernet with moving statistics across all subnets will lead to non-convergence.

2. Background and related work

2.1. Network quantization

Network quantization [3, 7, 13, 19, 21, 23, 24, 29–31] is widely used to accelerate deep neural networks. For a floating-point activation or weight r, its corresponding fixed-point number q under k bits can be formulated as:

$$q = Q(r, s) = \lfloor clip(\frac{r}{s}, -2^{k-1}, 2^{k-1} - 1) \rceil, \quad (1)$$

where $\lfloor * \rceil$ denotes the rounding-to-nearest function and s is the scale value that maps the floating-point number into a fixed-point number.

The scale value plays an important role in improving the accuracy of quantized model. Early works utilized exponential moving average of data to update the scale for activation quantization and use a pre-computed fixed scale for weight quantization. Esser *et al.* [13] proposed LSQ to improve the accuracy by learning the scale value during training. LSQ is commonly used in the subsequent quantization methods and quantization-aware NAS methods.

For a fully quantized model, the parameters and statistics of BN layer are folded into the weights and bias of Conv layer for efficiency. As an operator to accelerate the convergence of neural network training, BN layer [18] is widely used in numerous networks [15, 17, 34]. The calculation of BN layer with parameters γ and β can be formulated as:

$$y = \gamma\frac{(x - \mu)}{\sigma} + \beta, \quad (2)$$

where μ and σ are the mean and standard deviation (which we call it *std* in the following) of the input activations. Thus, the folded weights w_{fold} and bias b_{fold} can be formulated as:

$$w_{fold} = \frac{w\gamma}{\sigma_{MA}}, b_{fold} = -\frac{\gamma\mu_{MA}}{\sigma_{MA}} + \beta, \quad (3)$$

where σ_{MA} and μ_{MA} are the moving average of std and mean. Hence, the actual weights that need to be quantized are the folded weights.

To accurately simulate quantization effects, the fusion of Conv-BN layer is necessary to be simulated during training. A standard strategy is given in [19] and is followed by the subsequent works. As shown in Figure 3, fake quantization operator is used to train the quantized model. Fake quantization transfers the fixed-point number q into float-point number $\hat{r}$ by multiplying the scale s and performs backpropagation through straight through estimator [2]. The simulation of Conv-BN fusion is realized by quantizing the folded weight that combines the BN parameters. The quantized Conv-BN layer is reduced to a simple Conv layer by folding the moving statistics into the weights. The fake quantization operator directly quantizes the folded weights to simulate the inference behavior. Moving statistics of BN are updated with the output feature across all batch data.

2.2. NAS and Existing quantization-aware NAS

Neural Architecture Search (NAS) demonstrates effectiveness on automatically designing optimal network structures. Early works on NAS utilized reinforcement learning over a discrete set of candidate architectures [27, 33], so there are computationally demanding. DARTS [28] relaxed the search space to be continuous to obtain the best model by gradient descent. However, there is additional development cost because the searched architecture from DARTS need to be retrained. Recent studies [6, 9, 10, 32, 36, 37] on NAS trained a weight-sharing supernet in one-shot to save the training cost and decoupled NAS in two stages of training and searching.

Benefiting from the improvement of NAS, many works focus on combining quantization with NAS to automatically find the optimal quantized architecture. BATs [4] searched an optimal cell structure and stacked it to construct quantized models. APQ [39] trained a predictor to estimate

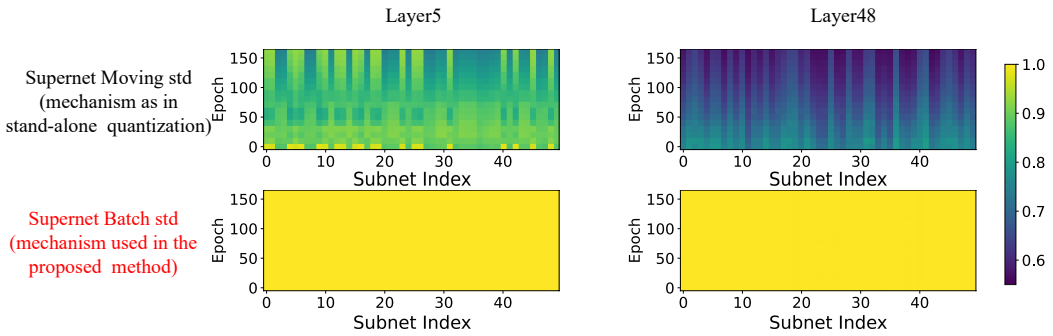

Figure 5. Cos similarity of moving std in subnets and moving std/batch std in supernet. Batch std is more similar to subnets' moving std.

quantized accuracy and used a look-up table to compute the latency of the models during search. The above methods cannot directly sample a quantized model, so the obtained architecture needs to be retrained to recover accuracy. To reduce the development cost, recent works tried to train a weight-sharing quantized supernet. OQAT [35] trained the supernet with shared scale and improved performance of low bits models by bit inheritance strategy. Bai *et al.* [[1]] proposed a quantizer for activation quantization to alleviate the unstable training problem for mix-precision supernet. However, the searched models from these methods are not fully quantized, which results in limited latency reduction.

3. Challenging in quantization-aware NAS

The training strategy as mentioned in Sec. 2.1 has proved to be effective in the stand-alone quantized model, but it is not suitable for a weight-sharing supernet. During each iteration of supernet training, a subnet is sampled from the supernet to perform forward and backward propagation for updating the shared weights. However, the subnets of the supernet are diverse in input resolution, channel numbers, depth, and expand ratios, so the moving statistics for each subnets are commonly different. Figure 4 depicts the cos similarity of moving std from 100 random sampled subnets. It can be clearly seen that the moving std is various across different subnets. As the depth increases, the similarity decreases further. The difference between subnets' moving std will bring two challenges to the quantization-aware NAS.

3.1. Non-convergence during training

It is unable to stash different moving statistics for different subnets during supernet training since the search space of the supernet contains numerous subnets. Thus, the moving statistics of the shared BN layer in the supernet are updated across different subnets during training. Figure 5 shows the cos similarity between the moving std of the supernet and the corresponding moving std of 50 randomly selected subnets at different epochs during training. The chosen BN layer is the same as Figure 4. As can be seen, there is a large difference between the moving std in the supernet and the moving std in the subnets. The difference is intensify as the training progresses. This difference makes the weights folded with the supernet's moving std different from the actual folded weight in each subnet. So this training strategy will mislead the direction of supernet optimization, result in non-convergence of the supernet.

3.2. Non-optimal searching results

As mentioned in Sec. 2.1, LSQ [13] demonstrates superiority in different works. Previous quantization-aware NAS methods commonly employed LSQ in supernet by learning a shared scale for each layer. However, the shared scale value learned by the LSQ quantizer during supernet training is not optimal for each subnet. Due to the difference in moving statistics between supernet and subnets, BN calibration is required to recover the performance as mentioned in [6]. According to Eq. 3, the folded weights of each subnet are different after BN calibration. Therefore, the shared scale is not the best for each subnets. Experimentally, the difference in the optimal scale can be up to 3 times, which brings almost 2-bit precision loss. Detailed experiment results can be found in supplementary material. Therefore, searching a quantized subnet based on a mismatched scale will lead to non-optimal results.

4. Framework

In this section, we present QuantNAS, a quantization-aware NAS framework for efficient deployment on mobile device. As shown in Figure 6, to tackle the problems of non-convergence during training and non-optimal searching results, the proposed framework jointly trains a scale predictor with the supernet under a batch-statistics-based training strategy. The batch-statistics-based training strategy and the mechanism of the scale predictor will be described in detail in Sec. 4.1 and 4.2. During search, the scale of each subnet is calibrated with the scale predictor after BN calibration

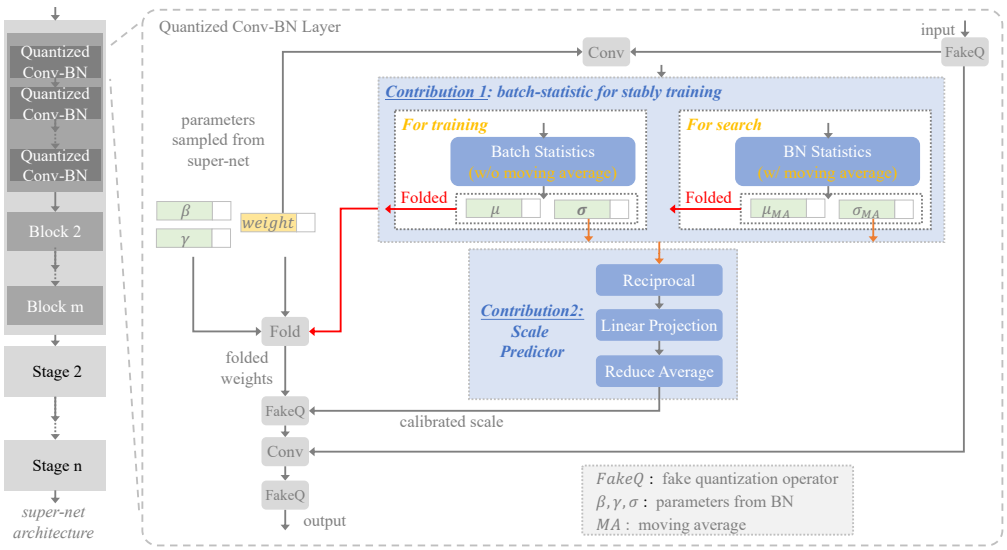

Figure 6. Framework of the proposed QuantNAS method. We train and search on a fully quantized supernet. We propose a batch-statistics-based strategy for stably training. A scale predictor is jointly trained with the supernet and is used to calibrate the scale during search.

to improve the performance. In this manner, the proposed framework is capable to find the optimal quantized architectures which can be directly deployed under different resource constraints without retraining.

4.1. Stable training with batch statistics

To tackle the problem of non-convergence during supernet training, we investigate the similarity between the batch std of the supernet and the corresponding moving std of the subnets. To be specific, in each iteration, batch std is computed with the input activations of the shared BN layer, and moving std is calibrated with a set of data on the sampled subnet. In Figure 5, we randomly select 50 iterations during supernet training and depicts the similarity between batch std of supernet and the moving std of the sampled subnets in these iterations. It can be seen that the moving std of different subnets is much more similar to the batch std of supernet than the moving std of the supernet. The similarity of batch std is over 0.999 on each epoch. Since the folded weights of each subnets is computed with the moving std of the subnets, using batch std of the supernet during training can improve the accuracy of each subnet.

Based on this investigation, we propose a stable batch-statistics-based training strategy for quantized supernet. As shown in Figure 6, at each iteration in training, the weights of the shared Conv layer and the parameters of the shared BN layer are sampled based on the activated subnet. Instead of folding the moving std of the supernet, we fold the std of the current batch into the sampled weights. The fake quantization takes the folded weights as input and the shared Conv-BN layer is reduced to a simple Conv layer to simulate the inference behavior.

4.2. Adaptive scale with a robust predictor

We propose a scale predictor to tackle the non-optimal searching results problem brought by BN calibration of subnets. The proposed scale predictor adaptively calibrates the optimal scale value for different subnets after BN calibration. To describe the cogitation in designing scale predictor, we analyze the process of Conv-BN fusion. For the Conv-Bn layer with input activation x, the output activation y *before fusion* can be formulated as:

$$y = \gamma * \frac{\hat{w}\hat{x} - \mu}{\sigma} + \beta = \frac{\gamma\hat{w}}{\sigma}\hat{x} - \frac{\gamma\mu}{\sigma} + \beta = \frac{\gamma\hat{w}}{\sigma}\hat{x} + b_{fold}, \quad (4)$$

where $\hat{w}$, $\hat{x}$ are the quantized weight and input of the Conv layer. During deployment, the folded bias $\hat{b}_{fold}$ in Eq. 4 is commonly quantized into 16 or 32 bits and has limited influence on accuracy. Therefore, we focus on the part of $\gamma\hat{w}\hat{x}/\sigma$. For a well-trained quantized model, the expansion form of Eq. 4 with scale s is:

$$y = \frac{s\gamma}{\sigma}\lfloor\frac{w}{s}\rceil\hat{x} \quad (5)$$

Here, we omit the folded bias and the clipping value for simplifying notification.

Now we consider the situation of quantizing the *fused* Conv-BN layer. The folded weights which should be quantized is $w\gamma/\sigma$. Supposing the channel number of the BN layer is m, which means $\sigma = [\sigma_1, \sigma_2, ..., \sigma_m]$ and $\gamma = [\gamma_1, \gamma_2, ..., \gamma_m]$. If each element of σ is σ_0 and each element of γ is γ_0, then the optimal scale s_{fold} for the folded weights

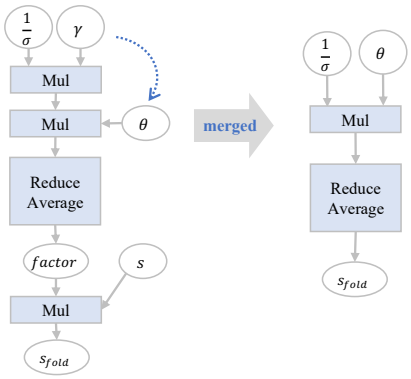

Figure 7. Illustration of the proposed scale predictor. The scale is predict with std σ and a trainable parameter θ.

can be simply reached by multiplying s with a transformation factor γ_0/σ_0. As formulated in Eq. 6, with optimal scale s_{fold}, the output activation of Conv-BN layer y is the same as the one in Eq. 4.

$$y' = s_{fold}\lfloor\frac{\gamma w}{\sigma s_{fold}}\rceil\hat{x} = \frac{\gamma s}{\sigma}\lfloor\frac{\gamma w}{\sigma}\frac{\sigma}{\gamma s}\rceil\hat{x} = \frac{s\gamma}{\sigma}\lfloor\frac{w}{s}\rceil\hat{x} \quad (6)$$

Unfortunately, the value of each element in σ or γ is typically different in the network. Hence, the role of the scale predictor is to extract a transformation factor from the vector $[\gamma_1/\sigma_1, \gamma_2/\sigma_2, ..., \gamma_m/\sigma_m]$ to minimize the loss of the quantized network.

As shown in Figure 6, scale predictor employs a trainable parameter θ to map the input vector $[\gamma_1/\sigma_1, \gamma_2/\sigma_2, ..., \gamma_m/\sigma_m]$ into output coefficients $f = [f_1, f_2, ..., f_m]$. The optimal scale for folded weights is obtained by multiplying the scale s and the reduce average of the coefficients f. For a well-trained quantized network fixed to be deployed, s and γ are fixed values. So we convert the scale predictor by merging γ and s into the parameter θ. The converted scale predictor can directly map the std statistics into the optimal scale. In this manner, the proposed scale predictor can learn a more flexible transformation. The formulation of the scale predictor is:

$$s_{pred} = \sum_{i}^{m} \frac{\theta_i}{\sigma_i}. \quad (7)$$

The trainable parameter θ is shared across different subnets, which is consistent with the weight-sharing scheme of the supernet.

The mechanism of scale predictor can perfectly cooperate with weight-sharing supernet. According to Eq. 6 and 7, scale predictor has generalization when the value of each σ_i is linearly enlarged. As described in Sec. 4.1, the batch std of supernet is consistent with the moving std of subnets after normalization, since they have a high cosine similarity.

Methods		Kendall Tau (τ)		
FQ	SP	Cifar10	Cifar100	ImageNet sub-100
×	×	0.78	0.69	0.72
✓	×	0.75	0.73	0.69
✓	✓	**0.91**	**0.84**	**0.87**

Table 1. Comparison of model ranking with different methods. 'FQ' represents fully-quantized, 'SP' represents scale predictor.

Method	GPU hours
APQ	2400
OQAT	2400
BatchQuant	1805
Ours	**1728**

Table 2. Training cost comparison with the state-of-the-art quantization-aware NAS.

To this end, taking batch std as input during training will lead to stable convergence of scale predictor. Moreover, the calibrated scale is the mean value of the output vector. Thus the scale is not corresponding to the number of channels that are different in each subnets.

For the initialization of the scale predictor, we compute an init scale value by minimizing the mean square error of the fixed-point and floating-point fold weights. The parameter θ_i is initialized with the following formula:

$$\theta_i = s_{init}\frac{\sigma_i}{\gamma_i}, \quad (8)$$

where σ_i is computed with few calibration data and γ_i is reloaded from the floating-point supernet. This initialization manner ensures that each element of the output vector f has the same contribution to the calibrated scale at the beginning of the training process.

5. Experiment

5.1. Experiment settings

Search space: The same with [1] and [35], we build our search space based on the Mobilenet-V3 [16] structure. Our search space contains multiple stages, each stage stacks different numbers of inverted blocks which have alternative input resolution, number of channels, kernel size, and expand ratios. We remove the Squeeze-Extraction (SE) block and replace the Hard-swish activation function with quantization-friendly ReLU function in our search space.

Dataset: We demonstrate the effectiveness of the proposed method on image classification and object detection tasks. We choose ImageNet 1K [12] dataset for classification and COCO [25] dataset for object detection. For the

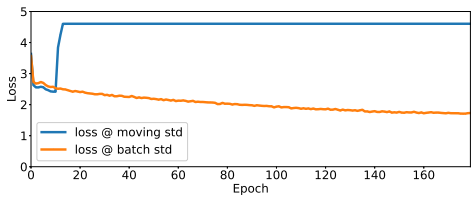

Figure 8. Comparison of training loss between folding moving std and batch std. The proposed batch-statistics-based strategy can lead to convergence.

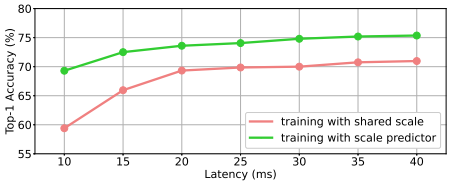

Figure 9. Pareto frontier of training with shared scale and the proposed scale predictor. The performance of training with scale predictor is superior.

ablation study in Sec. 5.2 and 5.3, we perform plentiful experiments on ImageNet sub-100, Cifar10 and Cifar100 [22] datasets. As in [32], the ImageNet sub-100 dataset is sampled from ImageNet 1K containing 100 categories, each category includes 250 training images and 50 validation images.

Quantization policy: During our experiment, we use a uniform symmetric quantization policy with a per-tensor scale. All the layers are quantized into 8-bit, since most hardware only supports 8-bit integer arithmetic. Previous works typically perform low-bit (*e.g.* 2, 3 or 4-bit) quantization, which this not only fails to reduce the on-device latency but also negatively impacts accuracy.

Training setting: The weight-sharing supernet is trained on 8 A100 GPUs for 180 epochs and the mini-batch size is 128 on each GPU. We use the SGD optimizer with the learning rate starting from 0.04. The learning rate is updated with the cosine decay. The momentum is set to 0.9 with 10^{-5} weight decay. The sandwich rule [41] is employed during training for fast convergence.

Search: During search, we utilize the evolutionary search strategy [6]. The evolutionary search is performed with 512 initial populations for 20 generations. For each generation, the mutate prob and the crossover prob are set to 0.2 and 0.25 respectively. Instead of using BitOps or the latency computed from a look-up table as a proxy as previous methods [1, 35, 39] did, we directly search the networks with the on-device latency. The latency is evaluated on Kirin 9000 CPU mobile device with tensorflow lite [11].

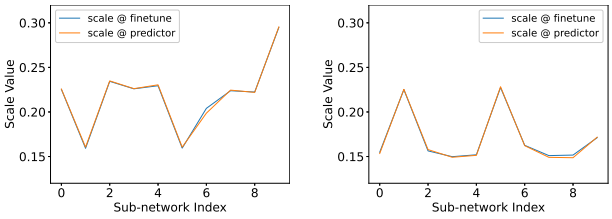

Figure 10. Comparison of the scale value from scale predictor and fintuned models.The calibrated scale is almost the same with ground-truth.

5.2. Effectiveness of batch-statistics-based training strategy

We compare the loss of training supernet with the moving statistics and the proposed batch statistics. As depicted in Figure 8, the loss trained with the moving statistics diverges after a few epochs, since the moving statistics in the supernet are not similar to the moving statistics in subnets. In contrast, the proposed strategy benefits from the high similarity between batch statistics in supernet and moving statistics in subnets, results in good convergence to supernet training.

5.3. Effectiveness of scale predictor

5.3.1 Model ranking

Model ranking refers to the ranking correlation between the architecture accuracy predicted from the NAS method and the ground truth accuracy by training from scratch. It is an important indicator to evaluate the performance of the NAS method as described in many articles [6, 9, 10, 32, 36, 37]. To demonstrate the effectiveness of the proposed scale predictor, we adopt Kendall coeffcient [20] to compare the ranking on 20 randomly sampled architectures. As shown in Table1, benefiting from the scale predictor, the proposed methods achieve the best model ranking, which is 11%∼18% higher than training with a shared scale on different datasets. This proves that the proposed method can find a more optimal structure by taking advantage of the scale predictor. Another finding is that the non-fully quantized method achieves lower ranking, which indicates that the existing quantization-aware NAS will result in suboptimal performance.

5.3.2 Pareto frontier

In figure 9, we verify the Pareto frontier of training the supernet with scale predictor and the one training with shared scale on ImageNet sub-100 dataset. More experiment results on Cifar10 and Cifar100 can be found in supplementary material. It can be seen that the structures on Pareto frontier of training with scale predictor have better performance than that of training with the shared scale. Under

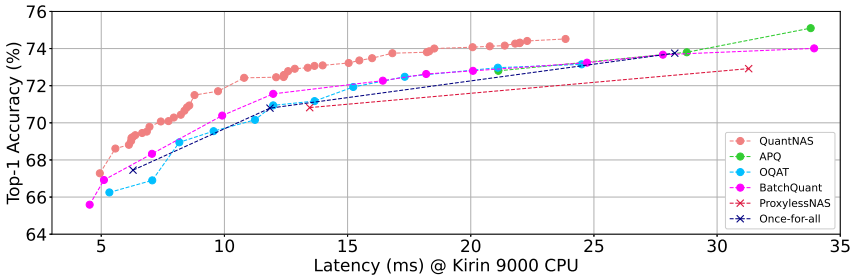

Figure 11. Comparison of the Pareto frontier on ImageNet-1K dataset. '•' represents quantization-aware NAS methods, '×' represents NAS-then-quantize methods.

the 40 ms latency target, the accuracy of the best architecture searched from the proposed method is 75.37%, which is 4.4% higher than that of the shared scale. This result shows the scale predictor can predict an optimal scale value for each subnet, resulting in better searching performance.

5.3.3 Robustness of scale predictor

To prove the robustness of the scale predictor, we randomly selected 10 subnets for only finetuning scale values. After performing BN calibration, we fix the weights of Conv and the BN statistics of each subnet, only updating the scale value with the LSQ quantizer. In Figure 10, a comparison of the finetuned scale and the calibrated scale in the 5th layer and 48th layer is depicted. As can be seen, the calibrated scale and the fine-tuned scale are nearly the same. In the randomly sampled subnets, the average relative error across all Conv-BN layers is 6.97%, which brings only 0.097 bits precision loss. This result proves that the scale calibrated by the proposed scale predictor is optimal and can recover the performance of subnets.

5.4. Comparison with SOTA methods

To demonstrate the superiority of the proposed method, we search for the quantized model under 25 ms latency target and compare the results with the state-of-the-art methods. On ImageNet-1K dataset, we compare our method with the NAS-then-quantized [5, 6] and existing quantization-aware NAS methods [1, 35, 39]. All the results are from our re-implementations, which keep the same quantization policy as our method. As shown in Figure 11, our method demonstrates the best searching performance. Existing quantization-aware NAS methods suffer from not being fully quantized, resulting in a worse latency-accuracy trade-off on real devices. Under 5ms, 15ms and 25ms latency constraints, the best models from our method achieve 67.28%, 73.10% and 74.51% accuracy respectively, which is 1.69%, 1.53% and 1.68% higher than the SOTA results. The design costs comparison with the quantization-aware NAS methods is shown in Table 2, our method can achieve better performance with lower cost.

	mAP	$mAP^{0.5}$	$mAP^{0.75}$	latency (ms)
ResNet-18	0.247	0.411	0.254	26.05
BatchQuant	0.255	0.425	0.263	24.72
Ours	**0.264**	**0.434**	**0.273**	**23.85**

Table 3. Performance on object detection task.

To verify the generalizability of our method, we evaluated the performance on object detection task with COCO dataset. The backbone of detection network is pretrained from ImageNet and BatchQuant [1] achieves the best result among the existing methods on ImageNet-1K. Therefore, we choose ResNet-18 [15] and the best models under 25ms latency from BatchQuant and our method to replace the backbone of RetinaNet [26]. All the networks are trained for 20 epochs. As shown in Table 3, benefiting from the fully quantization of the network, the mean average precision of our method is 0.9% higher than BatchQuant and 1.7% higher than ResNet-18. This experimental result proves that our method can achieve better performance on different tasks.

6. Conclusion

In this paper, we propose QuantNAS, a novel quantization-aware neural architecture search framework for efficient deployment on mobile device. The proposed framework can find the optimal quantized model under different resource constraints. By guaranteeing a fully quantized supernet, the sampled subnets can be directly deployed on different devices without retraining. To tackle the problems brought by Conv-BN fusion, we proposed a batch-statistics-based training strategy for stabilizing supernet training. A scale predictor is proposed to calibrate the scale of each subnet. With plentiful experiments, we prove the effectiveness of the proposed batch-statistics-based training strategy and the scale predictor. Our framework can improve the performance of the quantized model on different tasks compared with other methods.

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. 1, 2, 4, 6, 7, 8

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[3] 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. 1, 3

[4] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. Bats: Binary architecture search. In *European Conference on Computer Vision*, pages 309–325. Springer, 2020. 1, 3

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 8

[6] 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. 1, 2, 3, 4, 7, 8

[7] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020. 1, 3

[8] Hongjiang Chen, Yang Wang, Leibo Liu, Shaojun Wei, and Shouyi Yin. Hqnas: Auto cnn deployment framework for joint quantization and architecture search. *arXiv preprint arXiv:2210.08485*, 2022. 1

[9] Xiangxiang Chu, Bo Zhang, Qingyuan Li, Ruijun Xu, and Xudong Li. Scarlet-nas: bridging the gap between stability and scalability in weight-sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 317–325, 2021. 1, 3, 7

[10] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12239–12248, 2021. 1, 3, 7

[11] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems*, 3:800–811, 2021. 7

[12] 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. 6

[13] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1, 3, 4

[14] 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. 1, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 8

[16] 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. 6

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 1, 3

[20] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 7

[21] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1, 3

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[23] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019. 1, 3

[24] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 1, 3

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 6

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[27] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017. 3

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 3

[29] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 1, 3

[30] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[31] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. *arXiv preprint arXiv:2203.11086*, 2022. 1, 3

[32] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *Advances in Neural Information Processing Systems*, 33:17955–17964, 2020. 1, 3, 7

[33] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, pages 4780–4789, 2019. 3

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2, 3

[35] 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. 1, 2, 4, 6, 7, 8

[36] Xiu Su, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. K-shot nas: Learnable weight-sharing for nas with k-shot supernets. In *International Conference on Machine Learning*, pages 9880–9890. PMLR, 2021. 1, 3, 7

[37] 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. 1, 3, 7

[38] 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. 1, 2

[39] 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. 2, 3, 7, 8

[40] 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. 1

[41] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020. 7