DMS: DIFFERENTIABLE DIMENSION SEARCH FOR BINARY NEURAL NETWORKS

Yuhang Li[1], Ruihao Gong[1,2], Fengwei Yu[1], Xin Dong[3], Xianglong Liu[2]
[1]SenseTime Group Limited, [2]SKLSDE, Beihang University, [3]Harvard University
{liyuhang1, gongruihao, yufengwei}@sensetime.com

ABSTRACT

Binary Neural Network (BNN) has been widely explored in the field of the efficient deep learning area. However, since the inherent essence of BNN brings intensive oscillation in training and validation (Zhu et al., 2019), existing methods either seek new optimization methods or increase bit-width to bridge the gap to the full precision models. In this work, we focus on determining the dimension (i.e. number of weights or kernels) of BNN via Neural Architecture Search (NAS). Different from the conventional NAS method which rarely explores the channel numbers, we conjecture that the optimal dimension (channels) can be viewed as a continuous logistic random variable, then we sample the candidate architecture and learn the distribution of this variate. The search space is defined over a thousand nodes for a single conv-layer, but our method is efficient and able to reduce the search complexity from $\mathcal{O}(N)$ to $\mathcal{O}(1)$. Extensive experiments on CIFAR-10 and ImageNet dataset validate the effectiveness of the proposed algorithm.

1 INTRODUCTION

Quantized Neural Network (QNN) (Hubara et al., 2017) aims to lower the energy consumption, latency and memory cost when deep neural networks are deployed to resource-limited devices (e.g., mobile phones and embedded devices). Binary Neural Network (BNN), representing both weights and activation with binary values, is the most promising QNN approach for hardware performance, with 32× memory saving and about 58× computation acceleration on CPU (Rastegari et al., 2016). However, in practice finding a better local optimum of BNN over the binary (discrete) space becomes highly non-trivial (Zhu et al., 2019). For example, Kim et al. (2020) reach state-of-the-art BNN accuracy on ResNet-18 (He et al., 2016), but still suffer 10% top-1 accuracy drop compared to the full precision models. One compromising solution is to increase bit-width, where both uniform precision network (Zhou et al., 2016; Jung et al., 2019; Li et al., 2020) and mixed precision network (Wang et al., 2019) have been studied. Unfortunately, limited to the fixed network architecture and the increased bit-width, these methods inevitably face inferior speedup and memory saving.

Recently researchers have empirically or theoretically found that channels (number of kernels) greatly affect the performance of BNNs, which reveals the fact that there might be better structures that are more friendly to binarization. Wide Reduced Precision Network (Mishra et al., 2018) manually increases the width (channel number) and shows that performance can be improved along with the dimension in QNNs. Anderson & Berg (2018) show that in high dimension space the angle between a Gaussian vector and its binary vector can be small when dimension increases. Such theoretical results indicate that BNN can benefit from higher dimensions. Intuitively, each layer in CNN can have different optimal channels for a certain architecture. So it is natural to find the optimal channels by an automatic optimization rather than manually and roughly increase them.

Although NAS (Zoph & Le, 2016) can be used to obtain a state-of-the-art architecture, most NAS methods (Pham et al., 2018; Zoph et al., 2018; Real et al., 2019; Liu et al., 2018; Cai et al., 2018) determine the architecture with a deterministic channel number, rather than an adaptively optimized one. The reason is that it is difficult to optimize with such a huge search space (from 16 to 4k) for channels and meanwhile brings very little performance gains for the full-precision model. But as aforementioned, different from the full-precision model, finding the optimal dimension of BNN has greater potential in performance. Shen et al. (2019) try to use an evolutionary algorithm to search the channel numbers in BNN, however, they only search 6 expansion ratios for each layer, which causes suboptimal results. One way to solve it is to make the expansion ratio continuous but unfortunately, no existing NAS methods including DAS (Shin et al., 2018) can support this in an efficient way.

To resolve these problems, we proposed DMS (Differentiable diMension Search) which views the expansion ratio as a continuous logistic variate, then the continuous space is discretized to clustered search space, which is the key for decreasing the search complexity and makes the optimization possible. Finally, with the reduced continuous search space, we directly optimize the distribution as a whole in a differentiable way by the Gumbel-Softmax trick. In Sect. 2, we show the details of the DMS algorithm. In Sect. 3, extensive experiments on CIFAR-10 and ImageNet dataset prove the superiority of our method.

2 Differentiable Dimension Search

2.1 Preliminaries

Suppose weights in a convolutional layer are represented by a 4D tensor $\boldsymbol{W} \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k \times k}$, where c_{in} and c_{out} denote the input and the output channel of the layer, and k indicates the squared kernel size. Activations are denoted by tensor $\boldsymbol{X}$. In BNN, the weights and activations are binarized by:

$$\boldsymbol{w}^b = \text{sign}(\boldsymbol{w}) \times \frac{\|\boldsymbol{w}\|_1}{c_{\text{in}} \times k \times k}, \ \boldsymbol{X}^b = \alpha \times \text{round}(\text{clip}(\boldsymbol{X}/\alpha, 0, 1)), \tag{1}$$

where $\boldsymbol{w}$ is a convolutional kernel, α is the learnable quantization step size for activations. The sign function returns a vector of $\{-1, +1\}$ according to the sign of input vector and Eq. 1 rounds each element in $\boldsymbol{X}$ to $\{0, \alpha\}$. Our task is to find an optimal c_{out} for each layer.* Following Shen et al. (2019), we use channels times an expansion ratio r to denote the final channel number in the architecture. However, they only define 6 candidate value for r in search space, which is suboptimal.

We consider the search space $\mathcal{S}$ where $r \in \mathcal{S}$ and any $r \cdot c_{\text{out}}$ is a positive integer. In DARTS (Liu et al., 2018), the forward propagation of a node in the directed acyclic graph (DAG) is computed by:

$$\boldsymbol{O} = \sum_{r_i \in \mathcal{S}} \frac{\exp(\gamma_i)}{\sum_{r_j \in \mathcal{S}} \exp(\gamma_j)} (\boldsymbol{W}_i^b * \boldsymbol{X}^b), \tag{2}$$

where γ_i is the strength for the channel choice i and $\boldsymbol{W}_i \in \mathbb{R}^{r_i \cdot c_{\text{out}} \times c_{\text{in}} \times k \times k}$. Unfortunately, differentibale method require $\mathcal{O}(N)$ GPU memory to retain the graph nodes, and the search space in this task could be huge. For example, consider the original channel $c_{\text{out}} = 256$, the expansion ratio takes from 0.1 to 4, then, there could be near 1000 choices, which is not hardware-friendly.

2.2 Search Space

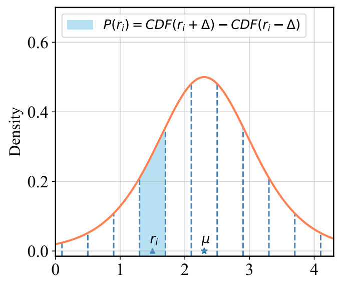
Our work is inspired by Louizos et al. (2019), where the weights are added with a noise and thus can be stochastically quantized. In Eq. 2, we notice that the channel numbers may be discrete, but all candidate channels are consecutive integers. This indicates that we do not need to use softmax to relax the discrete space. We conjecture that the optimal expansion ratio lies in a small range, (i.e., several consecutive candidates yield optimal results) whereas the other range in the search space is less favorable for the architecture. To model this probability among the search space, we first set the expansion ratio as a continuous random variable r and let it follow a Logistic distribution $L(\mu, \sigma)$. Given a certain distribution of the expansion ratio, we can calculate the probability of a candidate r_i by discretizing the range around this candidate and use the Cumulative Density Function (CDF) of the distribution.

$$p_i(\text{r} = r_i | \text{r} \sim L(\mu, \sigma)) = \text{CDF}(r_i + \Delta) - \text{CDF}(r_i - \Delta) = \frac{1}{1 + e^{((r_i + \Delta - \mu)/\sigma)}} - \frac{1}{1 + e^{((r_i - \Delta - \mu)/\sigma)}}, \tag{3}$$

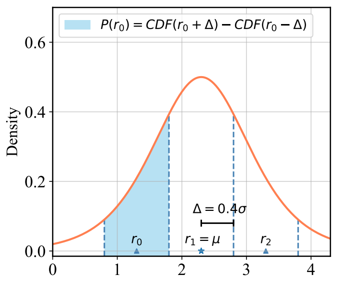
where $\Delta = 0.5/c_{\text{out}}$ is the half step size between two expansion ratio in the search space. In the search space, we want to measure the probability of being selected. First, let r_0 and r_m (where $m = |\mathcal{S}|$ and can be $+\infty$) be the least and largest expansion ratio in $\mathcal{S}$. Then we can compute the marginal probability by

$$\hat{p}_i(\text{r} = r_i | \text{r} \in (r_0 - \Delta, r_m + \Delta)) = \frac{\text{CDF}(r_i + \Delta) - \text{CDF}(r_i - \Delta)}{\text{CDF}(r_m + \Delta) - \text{CDF}(r_0 - \Delta)}, \tag{4}$$

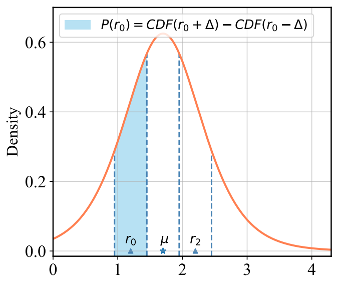
*Our method can also find the optimal kernel size and groups, we leave those in the future work.



(a) Logistic search space



(b) clustered search space



(c) clustered search space

Figure 1: **Left:** the categorical probability of r_i is the probability over a continous range. **Middle:** the search space has been clustered to only 3 representative expansion ratio, which reduces the search complexity. **Right:** after epochs of training, we reduce the variance of the distribution to narrow the range that contains optimal expansion ratio.

where $\sum_{r_i \in \mathcal{S}} \hat{p}_i = 1$. Fig. 1a illustrates the process that the continuous Logistic distribution becomes categorical distribution.

2.3 Candidate Clustering and Optimization

Unfortunately, even we can get the categorical probability of each candidate expansion ratio under the Logistic distribution. The search complexity is still $\mathcal{O}(N)$ because we have to compute each convolution in backward (Cai et al., 2018). Therefore, we cluster all candidates to only 3 representative candidates in the search space. The intuition is that since the channels are consecutive integers, it is unnecessary to strictly choose one candidate in forward. One candidate can share the same probability with its neighbors. In our experiments, we use 3 candidates in search space with $\mathcal{S}_c = \{\mu, \mu + 0.8\sigma, \mu - 0.8\sigma\}$, and we find they are sufficient enough to represent the search space. Fig. 1b shows the clustered candidate in the search space, where the search complexity is reduced to $\mathcal{O}(1)$. To calculate the probability of each candidate, we set $\Delta = 0.4 \cdot \sigma$ in Eq. 4.

In the clustered search space, the σ controls the variance of the distribution as well as the variance of the candidates. Therefore, we initialize σ to a high variance distribution so that the search space can cover a broad range in the search space. During training, we progressively decrease the value of σ, since we want to restraint the optimal channel range in a relatively small range as shown in Fig. 1c.

We use the binarized path for computing the feature maps like Cai et al. (2018), i.e., given the probability of each choice, we randomly sample one path in the forward pass. However, sampling from a categorical distribution is not differentiable. Here Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) is applied to make sure gradients can flow to the distribution:

$$\boldsymbol{O} = \sum_{r_i \in \mathcal{S}_c} \frac{\exp((\log \hat{p}_i + g_i)/\tau)}{\sum_{r_j \in \mathcal{S}_c} \exp((\log \hat{p}_j + g_j)/\tau)} (\boldsymbol{W}_i^b * \boldsymbol{X}^b), \text{ where } g_i \sim \textit{Gumbel}(0, 1). \tag{5}$$

τ is called temperature and controls the tightness of the softmax function. We use the same bilevel optimization problem (Liu et al., 2018) and use alternative method to optimize architecture parameters (expansion ratio) and weights. Since increasing channels for BNN will lead to greater latency and model size, we add hardware penalties in the optimization objective. Denote $\mathcal{L}_{\text{train}}$, $\mathcal{L}_{\text{valid}}$ as the training loss and the validation loss. We formulate the bilevel optimization problem as follows:

$$\min_{\mu} \; \mathcal{L}_{\text{valid}}(w^{b*}(\mu, \sigma), \mu) + \lambda \max(0, \text{Memory} - \text{Memory}_{target}), \tag{6}$$

$$\text{s.t. } \; w^{b*}(\mu, \sigma) = \arg\min_{w^b} \; \mathcal{L}_{\text{train}}(w^b, \mu), \tag{7}$$

where w^b indicates the binary weights in BNN, λ is the tradeoff parameter for hardware performances. Higher λ as well as lower Memory_{target} result in fewer parameter numbers but the accuracy may degrade and vice versa. In particular, we optimize the distribution ($L(\mu, \sigma)$) of expansion ratio. Unlike ProxylessNAS where only two candidates will be updated in the search space, all candidates in the search space can be updated after updating μ and adjusting σ while the search complexity can

Table 1: Accuracy and model size comparison between uniformly wide BNN and our DMS architecture on VGG-11 and ResNet-18. Acc.-1 and Acc.-5 denote Top-1 and Top-5 accuracy, respectively.

Models	**VGG-11 CIFAR-10**			**Res-18 CIFAR-10**			**Res-18 ImageNet**			
	Params	Saving	Acc.-1	Params	Saving	Acc.-1	Params	Saving	Acc.-1	Acc.-5
Full Prec.	2.20 MB	$1\times$	88.10	2.67 MB	$1\times$	92.75	44.5 MB	$1\times$	69.6	89.2
BNN $1\times$	0.08 MB	$27\times$	78.31	0.09 MB	$29\times$	85.33	3.29 MB	$13.5\times$	52.77	76.85
BNN $2\times$	0.29 MB	$7.6\times$	85.37	0.40 MB	$6.7\times$	90.25	9.24 MB	$4.8\times$	64.00	85.45
BNN $3\times$	0.65 MB	$3.4\times$	88.17	0.98 MB	$2.7\times$	92.25	17.8 MB	$2.5\times$	68.51	88.25
BNN $4\times$	1.14 MB	$1.9\times$	88.68	1.92 MB	$1.4\times$	93.01	29.1 MB	$1.5\times$	70.35	89.27
DMS-A	**0.08 MB**	$27\times$	**84.16**	**0.10 MB**	$27\times$	**89.32**	**2.29 MB**	$19.4\times$	**60.20**	**82.94**
DMS-B	**0.64 MB**	$2.6\times$	**89.10**	**0.84 MB**	$3.2\times$	**92.70**	**14.0 MB**	$3.18\times$	**67.93**	**87.84**

still be reduced to $\mathcal{O}(1)$. In architecture evaluation, we choose the mean of the distribution μ as the optimal expansion ratio for BNNs.

However, whenever μ is updated, the search space will change accordingly, which means the computation graph (including weights) we trained before is no longer retained and the network has to be trained from scratch. To extenuate such a problem, we use several methods: First, we do not alternatively optimize $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{valid}}$ for each batch iteration like DARTS but for several epochs. Second, we use cosine annealing learning rate (Loshchilov & Hutter, 2016) to accelerate the convergence of the weights when descending $\mathcal{L}_{\text{train}}$. Last but most important, we use warm restarts for w^b after updating μ, which initializes from the last trained model by intercepting the corresponding weights starting from the first channel. We call this as best effort initialization. Please refer to the detailed algorithm in Appendix. A.

3 Experiments and Conclusion

In this section, we test our DMS algorithm on two architecture channels for CIFAR-10 and ImageNet dataset. We report the model size and accuracy in Table 1. More experiments details and implementation can be found in Appendix. B.

CIFAR-10 We conduct experiments on VGG-11 (Simonyan & Zisserman, 2014) and ResNet-18 (He et al., 2016). Note that we keep other layer architecture settings (kernel size) the same except for the channel numbers. We compare the accuracy and the model size of the full precision model, BNN (with uniform expansion ratio) model. The results are shown in Table 1, from which we can see that directly binarize the full precision original model could result in a severely degraded BNN. BNN (without channel expansion) only achieves 78.31% on VGG-11 (85.33% on Res18) accuracy, which is 9.79% (resp. 7.42%) less than the full precision counterparts. When uniformly expand the channel to $2\times$ greater or even $4\times$ greater, the accuracy will approach to the full precision one. However, not all layers need expanding their channels. We first show DMS-A, where we set a relatively large penalty for model size, the model size is on par with the ordinary architecture, demonstrating our DMS algorithm can prune some redundant channels. DMS-A achieves almost the same accuracy of BNN $2\times$ with **3.6**$\times$ less model size in VGG-11. DMS-B has less penalty for model size, we show that DMS-B for ResNet-18 is on par with the full precision accuracy while still shares a $3.2\times$ compression ratio. The channel distribution for each layer is reported in Fig. 4.

ImageNet We highlight our search algorithm is efficient since the search space is clustered, therefore no proxy model is needed for large scale datasets like ImageNet. Due to the time limit, we directly use the r trained in CIFAR-10 for now. From Table 1 we notice that even if the DMS-A model is about 6% smaller than the vanilla binary counterpart (BNN $1\times$), it still achieves a significant accuracy improvement (7%) due to its binarization friendly characteristics. As for the DMS-B model, it enjoys more than a 20% reduction in model size with only 0.49% performance loss, which further shows the potential of our searched model.

Conclusion We have introduced the DMS algorithm to search the optimal channel numbers for BNN. DMS clusters the search space and learns the pre-defined logistic distribution of the expansion ratio. DMS is capable of searching the large-scale candidate (channel range) in a relatively short time. Various experiments validate the effectiveness and transferability of the proposed algorithm.

References

Alexander G. Anderson and Cory P. Berg. The high-dimensional geometry of binary neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1IDRdeCW.

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

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*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1x0lxrFPS.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkgXT24tDS.

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

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkxjYoCqKX.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. WRPN: Wide reduced-precision networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1ZvaaeAZ.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

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*, volume 33, pp. 4780–4789, 2019.

Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for accurate binary neural architectures. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Richard Shin, Charles Packer, and Dawn Song. Differentiable neural network architecture search, 2018. URL https://openreview.net/forum?id=BJ-MRKkwG.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4923–4932, 2019.

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

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*, pp. 8697–8710, 2018.

Algorithm 1: Search Algorithm for DMS.

```
Input: Initial distribution $\mu_0$, $\sigma_0$ for BNN; Total training epoch T ; training set and validation
       set.
for all $i = 1, 2, \ldots, T$-epoch do
    Cluster candidates and sample a specific architecture $w^b(\mu, \sigma)$;
    if Previous states exists then
        Best Effort Initialization($w^b(\mu, \sigma)$, Previous states);
    Get training set data;
    for all $j = 1, 2, \ldots, T'$-epoch do
        Descending $\mathcal{L}_{\text{train}}$ and update $w^b(\mu, \sigma)$ with training set;
        Cosine annealing learning rate $\eta$;
    Get validation set data;
    Descending $\mathcal{L}_{\text{valid}}$ and update $\mu$ with validation set;
    Previous states = $w^b(\mu, \sigma)$;
    Deflate $\sigma$ with a multiplier $\beta$.
return optimized $\mu$ ;
```

A Optimization Details

A.1 Algorithm

Alg. 1 illustrates the search algorithm of DMS. Our search algorithm is different from that of DARTS (Liu et al., 2018) as a result of the dynamic DAG. First, we update $w^b(\mu, \sigma)$ for T' epochs to get a closer approximation for w^{b*}. In each epoch, we use cosine annealing learning rate to accelerate the convergence. After updating μ and before we sample a new architecture from the distribution $L(\mu, \sigma)$, we first store the weights and then use the best effort initialization to warmly restart the weights in the sampled architecture. In particular, denote the w^{t-1} as the stored weights in the previous architecture and w^t weights in the new architecture. We initialize them as follows:

- if $\mathit{dim}(w^{t-1}) > \mathit{dim}(w^t)$, we intercept the first corresponding weights. i.e.,

$$w^t[\,:\,] = w^{t-1}[0 : \mathit{dim}(w^{t-1})] \tag{8}$$

- if $\mathit{dim}(w^{t-1}) \leq \mathit{dim}(w^t)$, we could only initialize the first corresponding dimension in w^t:

$$w^t[0 : \mathit{dim}(w^{t-1})] = w^{t-1}[\,:\,] \tag{9}$$

Though we could not restarts completely from the previous weights, the discrepancy will become smaller when the learning rate and variance of the distribution are downscaled. We show our best effort initialization is crucial for the convergence of the model in Sect. B.1.

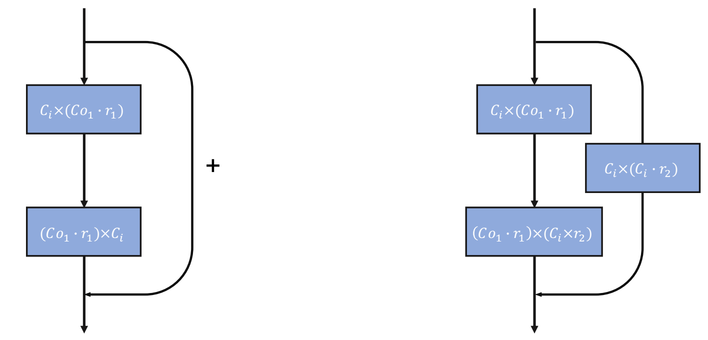
A.2 ResNet Architecture

To compare with full precision ResNets (He et al., 2016), and the uniformly wide BNN, we do not modify the architecture except for the channel numbers. Therefore, to keep residual connections intact, the input and output channel of a residual block should be kept the same. Shown in Fig. 2, we only set one expansion ratio for one residual block in an effort to keep the skip connections. For the blocks that have downsample layers, it is reasonable to allocate a learnable expansion ratio r_2 to the second convolutional layer.

B Experiments Details

B.1 Convergence

As aforementioned, DAG is dynamic during searching since either updating μ or adjusting σ leads to the change of search space. Therefore, we propose the best effort initialization (BEI) to warmly


$C_i \times (Co_1 \cdot r_1)$
+
$(Co_1 \cdot r_1) \times C_i$
$C_i \times (Co_1 \cdot r_1)$
$C_i \times (C_i \cdot r_2)$
$(Co_1 \cdot r_1) \times (C_i \times r_2)$

(a) ResBlk without downsample layer (b) ResBlk with downsample layer

Figure 2: ResNet architecture in DMS.

restart the sampled architecture. In Fig. 3, we compare the training error whether the BEI is applied or not. Though after the initialization the training error is slightly raised, it can reach the lowest in history after only a few epochs of training and thus can continue to be optimized. Without BEI, the final distribution of expansion ratio may not be suitable for BNNs since every time its input is not the optimal weight.

B.2 Implementation

We use PyTorch to implement our DMS algorithm. Our search code and evaluation code are released anonymously at https://github.com/yhhhli/DMS_for_BNN.

B.2.1 Network Search

For VGG-11 and ResNet-18, we do not modify any layer numbers and hyper-parameters (e.g., kernel size, padding, group) except for channel numbers. The original channels for CIFAR-10 are 1/4 of those in ImageNet, i.e., the channels increase from 16 to 128. We initialize the expansion ratio uniformly at 1.5 for DMS-A and 2.5 for DMS-B. σ is initialized uniformly to 1 and progressively decreased to 0.3 at the end of the search. This initialization ensures the expansion ratio has the same explore space at the beginning of the search. In our search space, we set the minimum expansion ratio to 0.3 and the minimum $\mu - \Delta$ is 0.25. We do not set the upper bound for expansion ratio, which means our search space is infinite. We also match the target memory to its corresponding 1x or 3x BNN. In particular, the target memory is set to 0.12, 1 for DMS-A and DMS-B. Half of the training data are held out as validation data, which is the same with DARTS. Total training epoch is set to 250, and we optimize μ for every 5 epochs training of weights. Batch size is set to 64 for both training and validation. SGD with momentum of 0.9 is adopted to optimize the weights, where the learning rate is set to 0.05 and annealed to 0.01 with a cosine schedule (Loshchilov & Hutter, 2016) in every 5 epochs. Adam Kingma & Ba (2014) optimizer is adopted when descending validation loss and the learning rate is set to 2×10^{-3} for VGG and 1×10^{-3} for ResNet. Weight decay is set to 10^{-4} and the tradeoff parameter λ is set to 0.01. We do not use Gumbel Straight Through when sampling the convolution layer with different dimensions. The temperature τ is set to 1 as a constant. The search algorithm only takes 2-4 hours on a single NVIDIA GTX 1080Ti GPU.

B.2.2 Network Evaluation

For CIFAR-10 experiments, we directly use the optimized μ for the expansion ratio when evaluating a model. VGG models are trained from scratch with SGD optimizer. Momentum is set to 0.9. We train the model for 300 epochs, the learning rate is initialized to 0.1 and decayed with a factor of 0.1 at epoch 140, 220 and 260. Weight decay is set to 10^{-4} for DMS-A and 2×10^{-4} for DMS-B. For ResNet-18, we first train the full precision model for 200 epochs and then we use the full precision model to initialize BNN. Other configurations are the same as VGG training.

For the ImageNet dataset, we directly apply the model searched on CIFAR-10 and train it for 100 epochs with the same hyper-parameter setting as the original ResNet-18. Then the binarized version

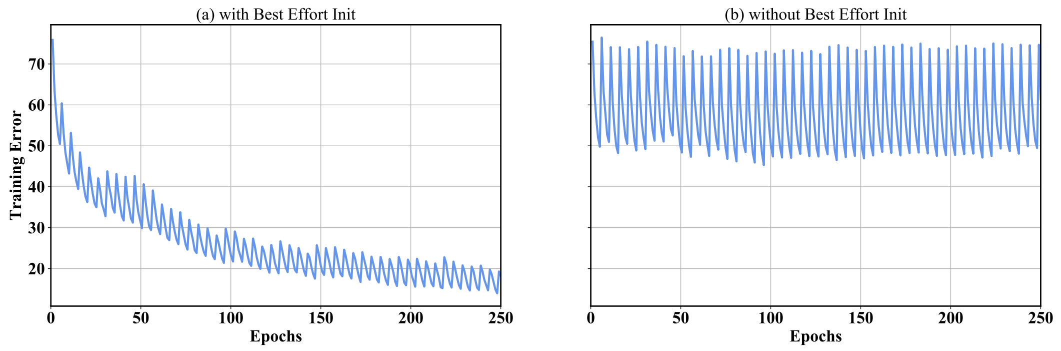


Figure 3: Training errors during searching. We show model without best effort initialization cannot converge.

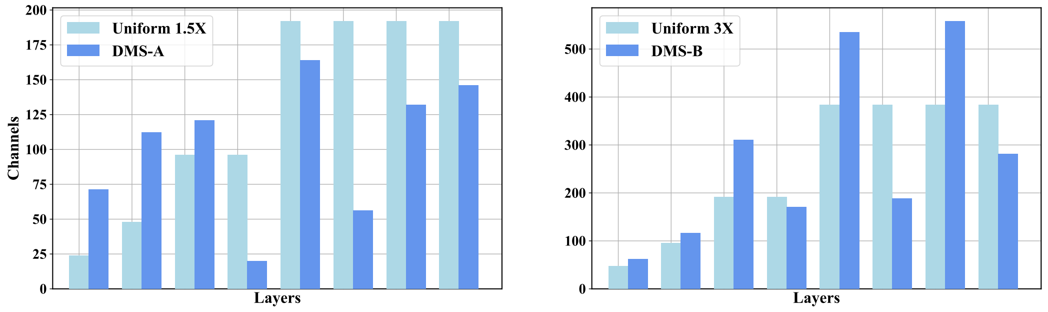
is trained with the initialization from this full-precision model. The learning rate is set to 4×10^{-4} and decays with a cosine learning rate scheduler.

B.2.3 BNN Settings

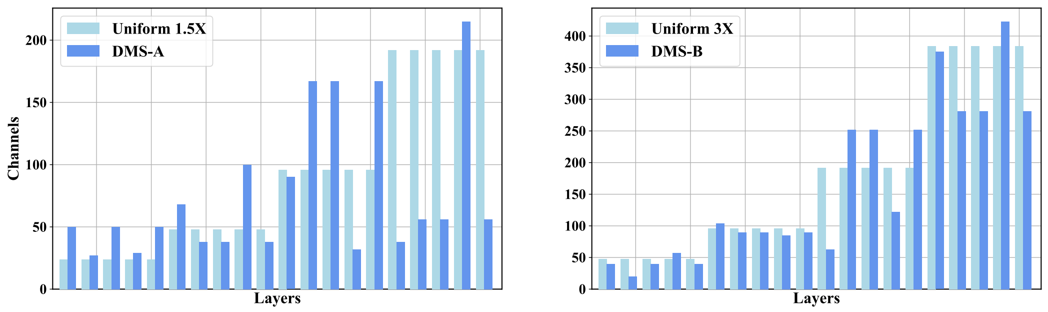
We use the binarization introduced in XNOR-Net (Rastegari et al., 2016) for weights. Before weight binarization, we apply Weight Standardization (Qiao et al., 2019) to normalize each filter to zero mean and unit variance. For activation binarization, the quantization step α is jointly optimized with network parameters. We do not quantize the first and the last layers like the prior works (Zhou et al., 2016).

B.3 Channel Distribution

We visualize the channel numbers of our DMS searched architecture on VGG-11 and ResNet-18 in Fig. 4. Conventional architecture design tends to progressively increase the channel numbers as the layers go deep. However, recent research (He et al., 2017) on structured pruning shows that channels are redundant in full precision neural networks. This redundancy may also exist when we want to widen the layers, typically in BNN. From Fig. 4, we can find that some layers only require small channels such as the fourth and the sixth layer in the DMS-A model of VGG-11, which means that they contain redundant information even when activations are binarized and can be extremely pruned. Comparing the left and right of Fig. 4a or Fig. 4b, we can find that the optimized channels differ slightly with respect to different model size penalty. And the results of VGG-11 and ResNet-18 also show different patterns. These two phenomena reveal that choosing the optimal architecture-specific and size-specific channels are crucial for pushing the accuracy of BNNs to the extreme.



(a) Channel distribution for VGG-11.



(b) Channel distribution for ResNet-18.

Figure 4: Comparison of channel numbers between wide BNN and our DMS architecture.