Learning Architectures for Binary Networks

https://github.com/gistvision/bnas

Dahyun Kim*[1][0000−0002−0820−4214], Kunal Pratap Singh*[2][0000−0003−3113−950X], and Jonghyun Choi[1][0000−0002−7934−8434]

[1] GIST (Gwangju Institute of Science and Technology), South Korea
[2] Indian Institute of Technology (IIT) Roorkee
killawhale@gm.gist.ac.kr ksingh@ee.iitr.ac.in jhc@gist.ac.kr

Abstract. Backbone architectures of most binary networks are well-known floating point (FP) architectures such as the ResNet family. Questioning that the architectures designed for FP networks might not be the best for binary networks, we propose to search architectures for binary networks (BNAS) by defining a new search space for binary architectures and a novel search objective. Specifically, based on the cell based search method, we define the new search space of binary layer types, design a new cell template, and rediscover the utility of and propose to use the *Zeroise* layer instead of using it as a placeholder. The novel search objective *diversifies early search* to learn better performing binary architectures. We show that our method searches architectures with stable training curves despite the quantization error inherent in binary networks. Quantitative analyses demonstrate that our searched architectures outperform the architectures used in state-of-the-art binary networks and outperform or perform *on par* with state-of-the-art binary networks that employ various techniques other than architectural changes.

Keywords: Binary networks · Backbone architecture · Architecture search

1 Introduction

Increasing demand for deploying high performance visual recognition systems encourages research on efficient neural networks. Approaches include pruning [12], efficient architecture design [14, 15, 42], low-rank decomposition [16], network quantization [6, 20, 32] and knowledge distillation [13, 37]. Particularly, network quantization, especially binary or 1-bit CNNs, are known to provide extreme computational and memory savings. The computationally expensive floating point convolutions are replaced with computationally efficient XNOR and bit-count operations, which significantly speeds up inference [32]. Hence, binary networks are incomparable with efficient floating point networks due to the extreme computational and memory savings.

Current binary networks, however, use architectures designed for floating point weights and activations [25, 28, 32, 34]. We hypothesize that the backbone

* indicates equal contribution. This work is done while KPS is at GIST for internship.

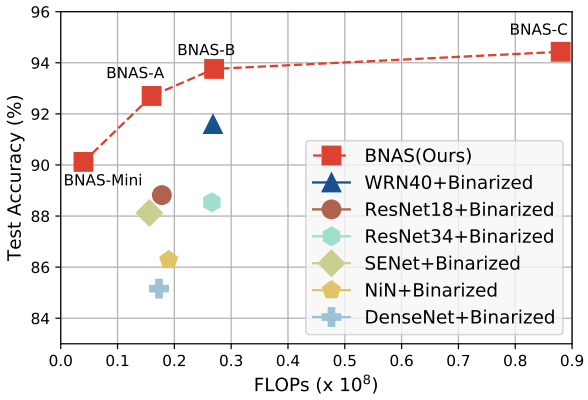


Fig. 1. Test accuracy (%) *vs*. FLOPs on CIFAR10 using the XNOR-Net binarization scheme [32]. Our searched architectures outperform the binarized floating point architectures. Note that our BNAS-Mini, which has much less FLOPs, outperforms all other binary networks except the one based on WideResNet40 (WRN40)

architectures used in current binary networks may not be optimal for binary parameters as they were designed for floating point ones. Instead, we may learn better binary network architectures by exploring the space of binary networks.

To discover better performing binary networks, we first apply one of the widely used binarization schemes [32] to the searched architectures from floating point NAS which use cell based search and gradient based search algorithms [9,27,40]. We then train the resulting binary networks on CIFAR10. Disappointingly, the binarized searched architectures do not perform well (Sec. 3). We hypothesize two reasons for the failure of binarized searched floating point architectures. First, the search space used in the floating point NAS is not necessarily the best one for binary networks. For example, separable convolutions will have large quantization error when binarized, since nested convolutions increase quantization error (Sec. 4.1). Additionally, we discover that the *Zeroise* layer, which was only used as a placeholder in floating point NAS, improves the accuracy of binary networks when kept in the final architecture (Sec. 4.1). Second, the cell template used for floating point cell based NAS methods is not well suited for the binary domain because of unstable gradients due to quantization error (Sec. 4.2).

Based on the above hypotheses and empirical observations, we formulate a cell based search space explicitly defined for binary networks and further propose a novel search objective with the diversity regularizer. The proposed regularizer encourages exploration of diverse layer types in the early stages of search, which is particularly useful for discovering better binary architectures. We call this method as Binary Network Architecture Search or *BNAS*. We show that the new search space and the diversity regularizer in BNAS helps in searching better performing binary architectures (Sec. 5).

Given the same binarization scheme, we compare our searched architectures to several handcrafted architectures including the ones shown in the Fig. 1. Our searched architectures clearly outperforms the architectures used in the state-of-the-art binary networks, indicating the prowess of our search method in discovering better architectures for binary networks.
We summarize our contributions as follows:

- We propose the first architecture search method for binary networks. The searched architectures are adjustable to various computational budgets (in FLOPs) and outperform backbone architectures used in state-of-the-art binary networks on both CIFAR10 and ImageNet dataset.
- We define a new search space for binary networks that is more robust to quantization error; a new cell template and a new set of layers.
- We propose a new search objective aimed to diversify early stages of search and demonstrate its contribution in discovering better performing binary networks.

2 Related Work

2.1 Binary Neural Networks

There have been numerous proposals to improve the accuracy of binary (1-bit) precision CNNs whose weights and activations are all binary valued. We categorize them into binarization schemes, architectural modifications and training methods.

Binarization Schemes. As a pioneering work, [6] proposed to use the sign function to binarize the weights and achieved compelling accuracy on CIFAR10. [7] binarized the weights and the activations by the sign function and use the straight through estimator (STE) to estimate the gradient. [32] proposed XNOR-Net which uses the sign function with a scaling factor to binarize the weights and the activations. They showed impressive performance on a large scale dataset (ImageNet ILSVRC 2012) and that the computationally expensive floating point convolution operations can be replaced by highly efficient XNOR and bit counting operations. Many following works including recent ones [25, 28] use the binarization scheme of XNOR-Net as do we. [22] approximated both weights and activations as a weighted sum of multiple binary filters to improve performance. Very recently, new binarization schemes have been proposed [3,10]. [10] uses projection convolutional layers while [3] improves upon the analytically calculated scaling factor in XNOR-Net.

These different binarization schemes do not modify the backbone architecture while we focus on finding better backbone architectures given a binarization scheme. A newer binarization scheme can be incorporated into our search framework but that was not the focus of this work.

Architectural Advances. It has been shown that appropriate modifications to the backbone architecture can result in great improvements in accuracy [25, 28, 32]. [32] proposed XNOR-Net which shows that changing the order of

batch normalization (BN) and the sign function is crucial for the performance of binary networks. [28] connected the input floating point activations of consecutive blocks through identity connections before the sign function. They aimed to improve the representational capacity for binary networks by adding the floating point activation of the current block to the consequent block. They also introduced a better approximation of the gradient of the sign function for back-propagation. [25] used circulant binary convolutions to enhance the representational capabilities of binary networks. [31] proposed a modified version of separable convolutions to binarize the MobileNetV1 architecture. However, we observe that the modified separable convolution modules do not generalize to architectures other than MobileNet. These methods do not alter the connectivity or the topology of the network while we search for entirely new network architectures.

Training Methods. There have been a number of methods proposed for training binary networks. [44] showed that quantized networks, when trained progressively from higher to lower bit-width, do not get trapped in a local minimum. [11] proposed a training method for binary networks using two new losses; Bayesian kernel loss and Bayesian feature loss. Recently, [18] proposed to pretrain the network with ternary activation which are later decoupled to binary activations for fine-tuning. The training methods can be used in our searched networks as well, but we focus on the architectural advances.

Recent Works on Binary Architecture Search. Recently, [34] performed a hyper-parameter search (*e.g.*, number of channels) using an evolutionary algorithm to efficiently increase the FLOPs of a binarized ResNet backbone. However, they trade more computation cost for better performance, reducing their inference speed up ($\sim 2.7\times$) to be far smaller than other binary networks ($\sim 10\times$). Concurrently in this conference, [2] propose to search for binary networks but using a different search space and search strategy than ours. However, the reported accuracy was difficult to reproduce with the given configuration details. We expect further progress in this field with reproducible public codes.

2.2 Efficient Neural Architecture Search

We search architectures for binary networks by adopting ideas from neural architecture search (NAS) methods for floating point networks [27, 30, 40, 45, 46]. To reduce the severe computation cost of NAS methods, there are numerous proposals focused on accelerating the NAS algorithms [1, 4, 5, 8, 9, 21, 23, 24, 26, 27, 29, 30, 38–41, 43]. We categorize these attempts into cell based search and gradient based search algorithms.

Cell Based Search. Pioneered by [46], many NAS methods [1, 5, 8, 9, 21, 23, 24, 26, 27, 29, 38–41, 43] have used the cell based search, where the objective of the NAS algorithm is to search for a cell, which will then be stacked to form the final network. The cell based search reduces the search space drastically from the entire network to a cell, significantly reducing the computational cost. Additionally, the searched cell can be stacked any number of times given the computational budget. Although the scalability of the searched cells to higher

computational cost is a non-trivial problem [36], it is not crucial to our work because binary networks focus more on smaller computational budgets.

Gradient Based Search Algorithms. In order to accelerate the search, methods including [5, 9, 27, 38, 40] relax the discrete sampling of child architectures to be differentiable so that the gradient descent algorithm can be used. The relaxation involves taking a weighted sum of several layer types during the search to approximate a single layer type in the final architecture. [27] uses softmax of learnable parameters as the weights, while other methods [9, 38, 40] use the Gumbel-softmax [17] instead, both of which allow seamless back-propagation by gradient descent. Coupled with the use of the cell based search, certain work has been able to drastically reduce the search complexity [9].

We make use of both the cell based search and gradient based search algorithms but propose a novel search space along with a modified cell template and a new regularized search objective to search binary networks.

3 Binarizing Searched Architectures by NAS

It is well known that architecture search results in better performing architecture than the hand-crafted ones. To obtain better binary networks, we first binarize the searched architectures by cell based gradient search methods. Specifically, we apply the binarization scheme of XNOR-Net along with their architectural modifications [32] to architectures searched by DARTS [27], SNAS [40] and GDAS [9]. We show the learning curves of the binarized searched floating point architectures on CIFAR10 dataset in Fig. 2.

Disappointingly, GDAS and SNAS reach around 40% test accuracy and quickly plummet while DARTS did not train at all. This implies that floating point NAS methods are not trivially extended to search binary networks. We investigate the failure modes in training and find two issues; 1) the search space is not well suited for binary networks, *e.g.*, using separable convolutions accumulates the quantization error repetitively and 2) the cell template does not propagate the gradients properly, due to quantization error. To search binary networks, the search space and the cell template should be redesigned to be robust to quantization error.

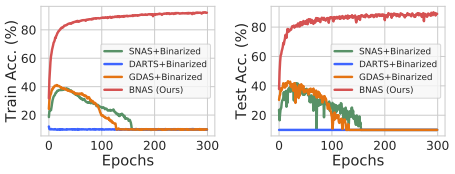


Fig. 2. Train (left) and test (right) accuracy of binarized searched architectures on CIFAR10. The XNOR-Net's binarization scheme and architectural modifications are applied in all cases. Contrasting to our BNAS, the binarized searched architectures fail to train well

4 Approach

To search binary networks, we first write the problem of cell-based architecture search in general as:

$$\alpha^* = \underset{\alpha \in A(\mathcal{S},T)}{\operatorname{argmin}} \mathcal{L}_S(D;\theta_\alpha), \tag{1}$$

where A is the feasible set of final architectures, $\mathcal{S}$ is the search space (a set of layer types to be searched), T is the cell template which is used to create valid networks from the chosen layer types, L_S is the search objective, D is the dataset, θ_α is the parameters of the searched architecture α which contain both architecture parameters (used in the continuous relaxation [27], Eq. 6) and the network weights (the learnable parameters of the layer types, Eq. 6), and α^* is the searched final architecture. Following [27], we solve the minimization problem using SGD.

Based on the observation in Sec. 3, we propose a new search space ($\mathcal{S}_B$), cell template (T_B) and a new search objective $\widetilde{\mathcal{L}}_S$ for binary networks which have binary weights and activations. The new search space and the cell template are more robust to quantization error and the new search objective $\widetilde{\mathcal{L}}_S$ promotes diverse search which is important when searching binary networks (Sec. 4.3). The problem of architecture search for binary network α_B^* can be rewritten as:

$$\alpha_B^* = \underset{\alpha_B \in A_B(\mathcal{S}_B,T_B)}{\operatorname{argmin}} \widetilde{\mathcal{L}}_S(D;\theta_{\alpha_B}), \tag{2}$$

where A_B is the feasible set of binary network architectures and θ_{α_B} is parameters of the binary networks. We detail each proposal in the following subsections.

4.1 Search Space for Binary Networks ($\mathcal{S}_B$)

Unlike the search space used in floating point NAS, the search space used for binary networks should be robust to quantization error. Starting from the search space popularly used in floating point NAS [9, 27, 40, 46], we investigate the robustness of various convolutional layers to quantization error and selectively define the space for the binary networks. Note that the quantization error depends on the binarization scheme and we use the scheme proposed in [32].

Convolutions and Dilated Convolutions. To investigate the convolutional layers' resilience to quantization error, we review the binarization scheme we use [32]. Let $\mathbf{W}$ be the weights of a floating point convolution layer with dimension $c \cdot w \cdot h$ (number of channels, width and height of an input) and $\mathbf{A}$ be an input activation. The floating point convolution can be approximated by binary parameters, $\mathbf{B}$, and the binary input activation, $\mathbf{I}$ as:

$$\mathbf{W} * \mathbf{A} \approx \beta \mathbf{K} \odot (\mathbf{B} * \mathbf{I}), \tag{3}$$

where $*$ denotes the convolution operation, $\odot$ is the Hadamard product (element wise multiplication), $\mathbf{B} = sign(\mathbf{W})$, $\mathbf{I} = sign(\mathbf{A})$, $\beta = \frac{1}{n}\|\mathbf{W}\|_1$ with $n = c \cdot w \cdot h$,

Table 1. Test Accuracy (%) of a small CNN composed of each layer type only, in floating point (FP Acc.) and in binary domain (Bin. Acc) on CIFAR10. *Conv*, *Dil. Conv* and *Sep. Conv* refer to the convolutions, dilated convolutions and separable convolutions, respectively. Separable convolutions show a drastically low performance on the binary domain

Layer Type	Conv		Dil. Conv		Sep. Conv	
Kernel Size	3×3	5×5	3×3	5×5	3×3	5×5
FP Acc. (%)	61.78	60.14	56.97	55.17	56.38	57.00
Bin. Acc. (%)	46.15	42.53	41.02	37.68	10.00	10.00

$\mathbf{K} = \mathbf{D} * \mathbf{k}$, $\mathbf{D} = \frac{\sum |A_{i,:,:}|}{c}$ and $\mathbf{k}_{ij} = \frac{1}{w \cdot h}$ $\forall ij$. Dilated convolutions are identical to convolutions in terms of quantization error.

Since both convolutions and dilated convolutions show tolerable quantization error in binary networks [32] (and our empirical study in Table 1), we include the standard convolutions and dilated convolutions in our search space.

Separable Convolutions. Separable convolutions [35] have been widely used to construct efficient network architectures for floating point networks [14] in both hand-crafted and NAS methods. Unlike floating point networks, we argue that the separable convolution is not suitable for binary networks due to large quantization error. It uses nested convolutions to approximate a single convolution for computational efficiency. The nested convolution are approximated to binary convolutions as:

$$Sep(\mathbf{W} * \mathbf{A}) \approx \beta_2(\mathbf{B}_2 * \mathbf{A}_2) \approx \beta_1\beta_2(\mathbf{B}_2 * (\mathbf{K}_1 \odot (\mathbf{B}_1 * \mathbf{I}_1))), \tag{4}$$

where $Sep(\mathbf{W} * \mathbf{A})$ denotes the separable convolution, $\mathbf{B}_1$ and $\mathbf{B}_2$ are the binary weights for the first and second convolution operation in the separable convolution layer, $\mathbf{I}_1 = sign(\mathbf{A})$, $\mathbf{A}_2 = \beta_1\mathbf{K}_1 \odot (\mathbf{B}_1 * \mathbf{I}_1)$ and $\beta_1, \beta_2, \mathbf{K}_1$ are the scaling factors for their respective binary weights and activations. Since every scaling factor induces quantization error, the nested convolutions in separable convolutions will result in more quantization error.

To empirically investigate how the quantization error affects training for different convolutional layer types, we construct small networks formed by repeating each kind of convolutional layers three times, followed by three fully connected layers. We train these networks on CIFAR10 in floating point and binary domain and summarize the results in Table 1.

When binarized, both convolution and dilated convolution layers show only a reasonable drop in accuracy, while the separable convolution layers show performance equivalent to random guessing (10% for CIFAR10). The observations in Table 1 imply that the accumulated quantization error by the nested convolutions fails binary networks in training. This also partly explains why the binarized architecture searched by DARTS in Fig. 2 does not train as it selects a large number of separable convolutions.

Table 2. DARTS and BNAS w/ and w/o the *Zeroise* layers in the final architecture on CIFAR10. *Zeroise Layer* indicates whether the *Zeroise* layers were kept (✓) or not (✗). The test accuracy of DARTS drops by 3.02% when you include the *Zeroise* layers and the train accuracy drops by 63.54% and the training stagnates. In constrast, the *Zeroise* layers improves BNAS in both train and test accuracy

Precision	Floating Point (DARTS)			Binary (BNAS)		
Zeroise Layer	✗	✓	Gain	✗	✓	Gain
Train Acc. (%)	99.18	35.64	-63.54%	93.41	97.46	+4.05%
Test Acc. (%)	97.45	94.43	-3.02%	89.47	92.70	+3.23%

Zeroise. The *Zeroise* layer outputs all zeros irrespective of the input [27]. It was originally proposed to model the lack of connections. Further, in the authors' implementation of [27][3], the final architecture excludes the *Zeroise* layers and replaces it with the second best layer type, even if the search picks the *Zeroise* layers. Thus, the *Zeroise* layers are not being used as they were originally proposed but simply used as a placeholder for a different and sub-optimal layer type. Such replacement of layer types effectively removes all architectures that have *Zeroise* layers from the feasible set of final architectures.

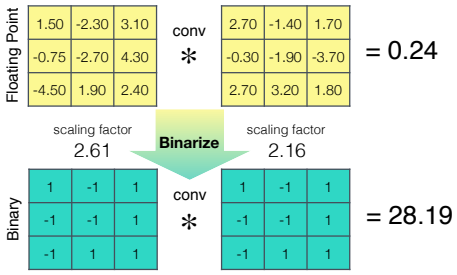


Fig. 3. An example when the *Zeroise* layer is beneficial for binary networks. Since the floating point convolution is close to zero but the binarized convolution is far greater than 0, if the search selects the *Zeroise* layer instead of the convolution layer, the quantization error reduces significantly

In contrast, we use the *Zeroise* layer for *reducing the quantization error* and are the first to keep it in the *final* architectures instead of using it as a placeholder for other layer types. As a result, our feasible set is different from that of [27] not only in terms of precision (binary), but also in terms of the network topology it contains.

As the exclusion of the *Zeroise* layers is not discussed in [27], we compare the accuracy with and without the *Zeroise* layer for DARTS in the DARTS column of Table 2 and empirically verify that the *Zeroise* layer is not particularly useful for floating point networks. However, we observe that the *Zeroise* layer improve the accuracy by a meaningful margin in binary networks as shown in the table. We argue that the *Zeroise* layer can reduce quantization error in binary networks as an example in Fig.3. Including the *Zeroise* layer in the final architecture is particularly beneficial when the situation similar to Fig. 3 happens frequently as the quantization error reduction is significant. But the degree of benefit may

[3] https://github.com/quark0/darts

Table 3. Proposed search space for BNAS. *Bin Conv*, *Bin Dil. Conv*, *MaxPool* and *AvgPool* refer to the binary convolution, binary dilated convolution, max pooling and average pooling layers, respectively

Layer Type	Bin Conv.		Bin Dil. Conv.		MaxPool	AvgPool	Zeroise
Kernel Size	3×3	5×5	3×3	5×5	3×3	3×3	N/A

differ from dataset to dataset. As the dataset used for search may differ from the dataset used to train and evaluate the searched architecture, we propose to tune the probability of including the *Zeroise* layer. Specifically, we propose a generalized layer selection criterion to adjust the probability of including the *Zeroise* layer by a transferability hyper-parameter γ as:

$$p^* = \max\left[\frac{p_z}{\gamma}, p_{op_1}, ..., p_{op_n}\right], \tag{5}$$

where p_z is the architecture parameter corresponding to the *Zeroise* layer and p_{op_i} are the architecture parameters corresponding to the i^{th} layer other than *Zeroise*. Larger γ encourages to pick the *Zeroise* layer only if it is substantially better than the other layers.

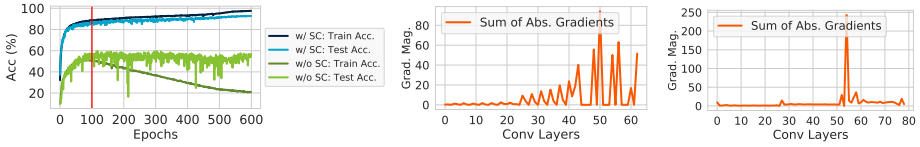
With the separable convolutions and the *Zeroise* layer type considered, we summarize the defined search space for BNAS ($\mathcal{S}_B$) in Table 3.

4.2 Cell Template for Binary Networks (T_B)

With the defined search space, we now learn a network architecture with the convolutional cell template proposed in [46]. However, the learned architecture still suffers from unstable gradients in the binary domain as shown in Fig. 4-(a) and (b). Investigating the reasons for the unstable gradients, we observe that the skip-connections in the cell template proposed in [46] are confined to be inside a single convolutional cell, *i.e.*, intra-cell skip-connections. The intra-cell skip-connections do not propagate the gradients outside the cell, forcing the cell to aggregate outputs that always have quantization error created inside the cell. To help convey information without the cumulative quantization error through multiple cells, we propose to add skip-connections between multiple cells as illustrated in Fig.5.

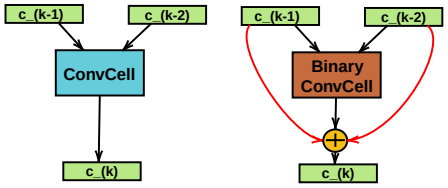
The proposed cell template with inter-cell skip-connections help propagate gradients with less quantization error throughout the network, stabilizing the training curve. We empirically validate the usefulness of the inter-cell skip connections in Sec. 5.4.

4.3 Search Objective with Diversity Regularizer ($\widetilde{\mathcal{L}}_S$)



(a) Learning curve (b) Gradients w/o SC (c) Gradients w/ SC

Fig. 4. Unstable gradients in the binary domain. 'w/o SC' indicates the cell template of [46] (Fig. 5-(a)). 'w/ SC' indicates the proposed cell template (Fig. 5-(b)). The gradient magnitudes are taken at epoch 100. With the proposed cell template ('w/ SC'), the searched network trains well (a). The proposed cell template shows far less spiky gradients along with generally larger gradient magnitudes ((b) vs. (c)), indicating that our template helps to propagate gradients more effectively in the binary domain



(a) CT in DARTS (b) CT in BNAS

Fig. 5. Cell templates (CT) of (a) DARTS and (b) BNAS. Red lines in BNAS indicate inter-cell skip connections. ConvCell indicates the convolutional cell. *c_(k)* indicates the output of the k$^{\text{th}}$ cell

With the feasible set of binary architectures (A_B) defined by S_B and T_B, we solve the optimization problem similar to [27]. However, the layers with learnable parameters (*e.g.*, convolutional layers) are not selected as often early on as the layers requiring no learning, because the parameter-free layers are more favorable than the under-trained layers. The problem is more prominent in the binary domain because binary layers train slower than the floating point counterparts [7]. To alleviate this, we propose to use an exponentially annealed entropy based regularizer in the search objective to promote selecting diverse layers and call it the *diversity regularizer*. Specifically, we subtract the entropy of the architecture parameter distribution from the search objective as:

$$\widetilde{\mathcal{L}}_S(D;\theta_{\alpha_B}) = \mathcal{L}_S(D;\theta,p) - \lambda H(p) e^{(-t/\tau)}, \tag{6}$$

where $\mathcal{L}_S(\cdot)$ is the search objective of [27], which is a cross-entropy, θ_{α_B} is the parameters of the sampled binary architecture, which is split into the architecture parameters p and the network weights θ, $H(\cdot)$ is the entropy, λ is a balancing hyper-parameter, t is the epoch, and τ is an annealing hyper-parameter. This will encourage the architecture parameter distribution to be closer to uniform in the early stages, allowing the search to explore diverse layer types.

Using the proposed diversity regularizer, we observed a 16% relative increase in the average number of learnable layer types selected in the first 20 epochs of the search. More importantly, we empirically validate the benefit of the diversity regularizer with the test accuracy on the CIFAR10 dataset in Table 4 and in Sec. 5.4. While the accuracy improvement from the diversity regularizer in

Table 4. Effect of searching diversity on CIFAR10. *Diversity* refers to whether diversity regularization was applied (✓) or not (✗) during the search. DARTS only gains 0.20% test accuracy while BNAS gains 1.75% test accuracy

Precision	Floating Point (DARTS)			Binary (BNAS)		
Diversity	✗	✓	Gain	✗	✓	Gain
Test Acc. (%)	96.53	96.73	+0.20	90.95	92.70	+1.75

the floating point NAS methods such as DARTS [27] is marginal (+0.2%), the improvement in our binary network is more meaningful (+1.75%).

5 Experiments

5.1 Experimental Setup

Datasets. We use CIFAR10 [19] and ImageNet (ILSVRC 2012) [33] datasets to evaluate the image classification accuracy. We follow [27] in splitting the datasets for search and training. Please refer to the supplementary material for additional details regarding the search and final evaluation settings.

Details on Comparison with Other Binary Networks. For XNOR-Net with different backbone architectures, we use the floating point architectures from `torchvision` or a public source[4] and apply the binarization scheme of XNOR-Net. Following previous work [10, 28] on comparing ABC-Net with a single base [22], we compare PCNN with a single projection kernel for both CIFAR10 and ImageNet. Please refer to the supplementary material for a qualitative comparison between our searched cells and other non-searched networks. The code and learned models will be available in our repository.

5.2 Comparisons on Backbone Architectures for Binary Networks

We quantitatively compare our searched architectures to various backbone architectures that have been used in the state-of-the-art binary networks with the binarization scheme of XNOR-Net [32] in Table 5. The comparisons differ only in the backbone architecture, allowing us to isolate the effect of our searched architectures on the final accuracy, *i.e.*, the comparison with XNOR-Net with different backbone architectures for various FLOPs and newer binary networks with the architectural contributions only. To single out the architectural contributions of Bi-Real Net, we used Table 1 in [28] to excerpt the ImageNet classification accuracy with using only the Bi-Real Net architecture. Note that CBCN is based on the Bi-Real Net architecture with the convolutions being changed to circulant convolutions[5]. Additionally, as mentioned in Sec. 2, we do

[4] https://github.com/kuangliu/pytorch-cifar

[5] They mention that center loss and gaussian gradient update is also used but they are not elaborated and not the main focus of CBCN's method.

Table 5. Comparison of different backbone architectures for binary networks with XNOR-Net binarization scheme [32] in various FLOPs budgets. Bi-Real* indicates Bi-Real Net's method with only the architectural modifications. We refer to [18] for the FLOPs of CBCN. CBCN* indicates the highest accuracy for CBCN with the ResNet18 backbone as [25] report multiple different accuracy for the same network configuration. Additionally, [25] does not report the exact FLOPs of their model, hence we categorized them conservatively into the '~ 0.27' bracket

Dataset	FLOPs ($\times 10^8$)	Model (Backbone Arch.)	Top-1 Acc. (%)	Top-5 Acc. (%)
CIFAR10	~ 0.16	XNOR-Net (ResNet18)	88.82	-
		XNOR-Net (DenseNet)	85.16	-
		XNOR-Net (NiN)	86.28	-
		XNOR-Net (SENet)	88.12	-
		BNAS-A	**92.70**	-
	~ 0.27	XNOR-Net (ResNet34)	88.54	-
		XNOR-Net (WRN40)	91.58	-
		CBCN* (ResNet18) [25]	91.91	-
		BNAS-B	**93.76**	-
	~ 0.90	XNOR-Net (ResNext29-64)	84.27	-
		BNAS-C	**94.43**	-
ImageNet	~ 1.48	XNOR-Net (ResNet18)	51.20	73.20
		BNAS-D	**57.69**	**79.89**
	~ 1.63	Bi-Real* (Bi-Real Net18) [28]	32.90	56.70
		BNAS-E	**58.76**	**80.61**
	~ 1.78	XNOR-Net (ResNet34)	56.49	79.13
		BNAS-F	**58.99**	**80.85**
	~ 1.93	Bi-Real* (Bi-Real Net34) [28]	53.10	76.90
		BNAS-G	**59.81**	**81.61**
	~ 6.56	CBCN (Bi-Real Net18) [25]	61.40	82.80
		BNAS-H	**63.51**	**83.91**

not compare with [34] as the inference speed-up is significantly worse than other binary networks ($\sim 2.7\times$ compared to $\sim 10\times$), which makes the comparison less meaningful.

As shown in Table 5, our searched architectures outperform other architectures used in binary networks in all FLOPs brackets and on both CIFAR10 and ImageNet. Notably, comparing XNOR-Net with the ResNet18 and ResNet34 backbone to BNAS-D and BNAS-F, we gain +6.49% or +2.50% top-1 accuracy and +6.69% or +1.72% top-5 accuracy on ImageNet.

Furthermore, BNAS retains the accuracy much better at lower FLOPs, showing that our searched architectures are better suited for efficient binary networks. Additionally, comparing CBCN to BNAS-H, we gain +2.11% top-1 accuracy and +1.11% top-5 accuracy, showing that our architecture can scale to higher FLOPs budgets better than CBCN. In sum, replacing the architectures used in current binary networks to our searched architectures can greatly improve the performance of binary networks.

Table 6. Comparison of other binary networks in various FLOPs budgets. The binarization schemes are: '*Sign + Scale*': using fixed scaling factor and the sign function [32], '*Sign*': using the sign function [7], '*Clip + Scale*': using clip function with shift parameter [22], , '*Sign + Scale**': using learned scaling factor and the sign function [3], '*Projection*': using projection convolutions [10], '*Bayesian*': using a learned scaling factor from the Bayesian losses [11] and the sign function, and '*Decoupled*': decoupling ternary activations to binary activations [18]

Dataset	FLOPs ($\times 10^8$)	Method (Backbone Arch.)	Binarization Scheme	Pretraining	Top-1 Acc. (%)	Top-5 Acc. (%)
CIFAR10	~ 0.04	PCNN($i=16$) (ResNet18) [10]	Projection	✗	89.16	-
		BNAS-Mini	Sign + Scale	✗	**90.12**	-
	~ 0.16	BinaryNet (ResNet18) [7]	Sign	✗	89.95	-
		BNAS-A	Sign + Scale	✗	**92.70**	-
	~ 0.27	PCNN($i=64$) (ResNet18) [10]	Projection	✓	**94.31**	-
		BNAS-B	Sign + Scale	✗	93.76	-
ImageNet	~ 1.48	BinaryNet (ResNet18) [7]	Sign	✗	42.20	67.10
		ABC-Net (ResNet18) [22]	Clip + Sign	✗	42.70	67.60
		BNAS-D	Sign + Scale	✗	**57.69**	**79.89**
	~ 1.63	Bi-Real (Bi-Real Net18) [28]	Sign + Scale	✓	56.40	79.50
		XNOR-Net++ (ResNet18) [3]	Sign + Scale*	✗	57.10	79.90
		PCNN (ResNet18) [10]	Projection	✓	57.30	80.00
		BONN (Bi-Real Net18) [11]	Bayesian	✗	59.30	81.60
		BinaryDuo (ResNet18) [18]	Decoupled	✓	**60.40**	**82.30**
		BNAS-E	Sign + Scale	✗	58.76	80.61
	~ 1.78	ABC-Net (ResNet34) [22]	Clip + Scale	✗	52.40	76.50
		BNAS-F	Sign+Scale	✗	**58.99**	**80.85**
	~ 1.93	Bi-Real (Bi-Real Net34) [28]	Sign + Scale	✓	**62.20**	**83.90**
		BNAS-G	Sign + Scale	✗	59.81	81.61

5.3 Comparison with Other Binary Networks

As we focus on improving binary networks by architectural benefits only, comparison to other binary network methods is not of our interest. However, it is still intriguing to compare gains from a pure architectural upgrade to gains from new binarization schemes or new training methods. As shown in Table 6, our searched architectures outperform other methods in more than half the FLOPs brackets spread across CIFAR10 and ImageNet. Moreover, the state-of-the-art methods that focus on discovering better training schemes are complementary to our searched architectures, as these training methods were not designed exclusively for a fixed network topology.

Note that, with the same backbone of ResNet18 or ResNet34, Bi-Real, PCNN, XNOR-Net++ and BONN have higher FLOPs than ABC-Net, XNOR-Net and BinaryNet. The higher FLOPs are from unbinarizing the downsampling convolutions in the ResNet architecture.[6]

5.4 Ablation Studies

We perform ablation studies on the proposed components of our method. We use the CIFAR10 dataset for the experiments with various FLOPs budgets and summarize the results in Table 7.

[6] We have confirmed with the authors of [32] that their results were reported without unbinarizing the downsampling convolutions.

Table 7. Classification acc. (%) of ablated models on CIFAR10. *Full* refers to the proposed method with all components. *No Skip*, *No Zeroise*, and *No Div* refers to our method without the inter-cell skip connections, with explicitly discarding the *Zeroise* layers, or without the diversity regularizer respectively.

Model	Full	No Skip	No Zeroise	No Div
BNAS-A	**92.70**	61.23	89.47	90.95
BNAS-B	**93.76**	67.15	91.69	91.55
BNAS-C	**94.43**	70.58	88.74	92.66

All components have decent contributions to the accuracy, with the inter-cell skip connection in the new cell template contributing the most; without it, the models eventually collapsed to very low training and test accuracy and exhibited unstable gradient issues as discussed in Sec. 4.2. Comparing *No Div* with *Full*, the searched cell with the diversity regularizer has a clear gain over the searched cell without it in all the model variants. Interestingly, the largest model (BNAS-C) without *Zeroise* layers performs worse than BNAS-A and BNAS-B, due to excess complexity. Please refer to the supplementary material for more discussion.

6 Conclusion

To design better performing binary network architectures, we propose a method to search the space of binary networks, called BNAS. BNAS searches for a cell that can be stacked to generate networks for various computational budgets. To configure the feasible set of binary architectures, we define a new search space of binary layer types and a new cell template. Specifically, we propose to exclude separable convolution layer and include *Zeroise* layer type in the search space for less quantization error. Further, we propose a new search objective with the diversity regularizer and show that it helps in obtaining better binary architectures. The learned architectures outperform the architectures used in the state-of-the-art binary networks on both CIFAR-10 and ImageNet.

Acknowledgement. This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2019R1 C1C1009283), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01842, Artificial Intelligence Graduate School Program (GIST) and No.2019-0-01351, Development of Ultra Low-Power Mobile Deep Learning Semiconductor With Compression/Decompression of Activation/Kernel Data), "GIST Research Institute(GRI) GIST-CNUH research Collaboration" grant funded by the GIST in 2020, and a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT' and NIPA.

The authors would like to thank Dr. Mohammad Rastegari for valuable comments and training details of XNOR-Net and Dr. Chunlei Liu and other authors of [25] for sharing their code.

References

1. Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V., Le, Q.: Understanding and simplifying one-shot architecture search. In: ICML (2018) 4
2. Bulat, A., Martínez, B., Tzimiropoulos, G.: Bats: Binary architecture search. ArXiv preprint arXiv:2003.01711 **abs/2003.01711** (2020) 4
3. Bulat, A., Tzimiropoulos, G.: Xnor-net++: Improved binary neural networks. In: BMVC (2019) 3, 13
4. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: ICLR (2019), https://openreview.net/forum?id=HylVB3AqYm 4
5. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In: ICCV. pp. 1294–1303 (2019) 4, 5
6. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: NIPS (2015) 1, 3
7. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830 (2016) 3, 10, 13
8. Dong, J.D., Cheng, A.C., Juan, D.C., Wei, W., Sun, M.: Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In: ECCV (2018) 4
9. Dong, X., Yang, Y.: Searching for a robust neural architecture in four gpu hours. In: CVPR (2019) 2, 4, 5, 6
10. Gu, J., Li, C., Zhang, B., Han, J., Cao, X., Liu, J., Doermann, D.: Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In: AAAI (2019) 3, 11, 13
11. Gu, J., Zhao, J., Jiang, X., Zhang, B., Liu, J., Guo, G., Ji, R.: Bayesian optimized 1-bit cnns. In: CVPR (2019) 4, 13
12. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: NIPS (2015) 1
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015) 1
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 1, 7
15. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arXiv:1602.07360 (2016) 1
16. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014) 1
17. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: ICLR (2017), https://arxiv.org/abs/1611.01144 5
18. Kim, H., Kim, K., Kim, J., Kim, J.J.: Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. In: ICLR (2020), https://openreview.net/forum?id=r1x0lxrFPS 4, 12, 13
19. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. Tech. rep. (2009) 11
20. Li, F., Zhang, B., Liu, B.: Ternary weight networks. arXiv preprint arXiv:1605.04711 (2016) 1

21. Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. arXiv preprint arXiv:1902.07638 (2019) 4
22. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. In: NIPS (2017) 3, 11, 13
23. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: CVPR (2019) 4
24. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: ECCV (2018) 4
25. Liu, C., Qi, Y., Xia, X., Zhang, B., Gu, J., Liu, J., Ji, R., Doermann, D.S.: Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In: CVPR (2019) 2, 3, 4, 12, 14
26. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. In: ICLR (2018), https://openreview.net/forum?id=BJQRKzbA- 4
27. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. In: ICLR (2019), https://openreview.net/forum?id=S1eYHoC5FX 2, 4, 5, 6, 8, 10, 11
28. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.T.: Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In: ECCV (2018) 2, 3, 4, 11, 12, 13
29. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. In: NIPS (2018) 4
30. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: ICML (2018) 4
31. Phan, H., Huynh, D., He, Y., Savvides, M., Shen, Z.: Mobinet: A mobile binary network for image classification. arXiv preprint arXiv:1907.12629 (2019) 4
32. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: ECCV (2016) 1, 2, 3, 4, 5, 6, 7, 11, 12, 13
33. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015) 11
34. Shen, M., Han, K., Xu, C., Wang, Y.: Searching for accurate binary neural architectures. In: ICCV Workshop (2019) 2, 4, 12
35. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification 7
36. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019) 5
37. Tan, S., Caruana, R., Hooker, G., Koch, P., Gordo, A.: Learning global additive explanations for neural nets using model distillation. arXiv preprint arXiv:1801.08640 (2018) 1
38. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: CVPR (2019) 4, 5
39. Xie, S., Kirillov, A., Girshick, R., He, K.: Exploring randomly wired neural networks for image recognition. arXiv preprint arXiv:1904.01569 (2019) 4
40. Xie, S., Zheng, H., Liu, C., Lin, L.: SNAS: stochastic neural architecture search. In: ICLR (2019), https://openreview.net/forum?id=rylqooRqK7 2, 4, 5, 6
41. Zhang, C., Ren, M., Urtasun, R.: Graph hypernetworks for neural architecture search. In: ICLR (2019), https://openreview.net/forum?id=rkgW0oA9FX 4
42. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: CVPR (2018) 1

43. Zhou, Y., Ebrahimi, S., Arık, S.Ö., Yu, H., Liu, H., Diamos, G.: Resource-efficient neural architect. arXiv preprint arXiv:1806.07912 (2018) 4
44. Zhuang, B., Shen, C., Tan, M., Liu, L., Reid, I.: Towards effective low-bitwidth convolutional neural networks. In: CVPR (2018) 4
45. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2017), https://openreview.net/forum?id=r1Ue8Hcxg 4
46. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR (2018) 4, 6, 9, 10