# CP-NAS: Child-Parent Neural Architecture Search for 1-bit CNNs

**Li'an Zhuo**[1] , **Baochang Zhang**[1*] , **Hanlin Chen**[1] , **Linlin Yang**[2] , **Chen Chen**[3] , **Yanjun Zhu**[4] and **David Doermann**[4]
[1]School of Automation Science and Electrical Engineering, Beihang University
[2]University of Bonn
[3]University of North Carolina at Charlotte
[4]University at Buffalo
{lianzhuo, bczhang, hlchen}@buaa.edu.cn

## Abstract

Neural architecture search (NAS) proves to be among the best approaches for many tasks by generating an application-adaptive neural architecture, which is still challenged by high computational cost and memory consumption. At the same time, 1-bit convolutional neural networks (CNNs) with binarized weights and activations show their potential for resource-limited embedded devices. One natural approach is to use 1-bit CNNs to reduce the computation and memory cost of NAS by taking advantage of the strengths of each in a unified framework. To this end, a Child-Parent (CP) model is introduced to a differentiable NAS to search the binarized architecture (Child) under the supervision of a full-precision model (Parent). In the search stage, the Child-Parent model uses an indicator generated by the child and parent model accuracy to evaluate the performance and abandon operations with less potential. In the training stage, a kernel-level CP loss is introduced to optimize the binarized network. Extensive experiments demonstrate that the proposed CP-NAS achieves a comparable accuracy with traditional NAS on both the CIFAR and ImageNet databases. It achieves the accuracy of $95.27\%$ on CIFAR-10, $64.3\%$ on ImageNet with binarized weights and activations, and a $30\%$ faster search than prior arts.

## 1 Introduction

Neural architecture search (NAS) has attracted a great deal of attention with a remarkable performance in many computer vision tasks. The goal is to design network architectures automatically to replace conventional hand-crafted counterparts, but at the expense of huge search space and high computational cost. To achieve efficient NAS, one line of existing NAS approaches focus on improving their search efficiency to explore the large search spaces, reducing the search time from thousands of GPU days [Zoph *et al.*, 2018; Zoph and Le, 2016] to few GPU days [Cai *et al.*, 2018a; Liu *et al.*, 2018b; Xu *et al.*, 2019; Chen *et al.*, 2019b]. These approaches were also developed into a more elegant framework named one-shot architecture search. Another line of NAS aims to search a more efficient network. Proxyless-NAS [Cai *et al.*, 2018c] introduces latency loss to search architectures on the target task instead of adopting the conventional proxy-based framework. EfficientNet [Tan and Le, 2019] introduces a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient to obtain efficient networks. Binarized neural architecture search (BNAS) [Chen *et al.*, 2019a] searches binarized networks with a significant memory saving, which provides a more promising way to efficiently find network architectures. However, BNAS only focuses on the kernel binarization, while the extremely compressed 1-bit CNNs with binarized weights and activations have not been well explored in the field of NAS.

Comparatively speaking, 1-bit CNNs based on hand-craftd architectures have been extensively researched. Filters binarization has been used in conventional CNNs to compress deep models [Rastegari *et al.*, 2016; Courbariaux *et al.*, 2016; Courbariaux *et al.*, 2015; Juefei-Xu *et al.*, 2017], showing up to $58\times$ speedup and $32\times$ memory saving, which is widely considered as one of the most efficient ways to perform computing on embedded devices with low computational cost. In [Juefei-Xu *et al.*, 2017], the XNOR network is presented where both the weights and inputs attached to the convolution are approximated with binarized values. This results in an efficient implementation of convolutional operations by reconstructing the unbinarized filters with a single scaling factor. In [Gu *et al.*, 2019], a projection convolutional neural network (PCNN) is proposed to implement binarized neural networks (BNNs) based on a simple back propagation algorithm. [Zhao *et al.*, 2019] proposes Bayesian optimized 1-bit CNNs, taking advantage of Bayesian learning to significantly improve the performance of extreme 1-bit CNNs. Binarized models show the advantages on computational cost reduction and memory saving, however, they suffer from poor performance in practical applications. There still remains a gap between 1-bit weights/activations and full-precision counterparts, which motivates us to explore the potential relationship between 1-bit and full-precision models to evalutate the performance of binarized networks based on NAS.

In this paper, we introduce a Child-Parent model to effi-

*Baochang Zhang is the corresponding author.

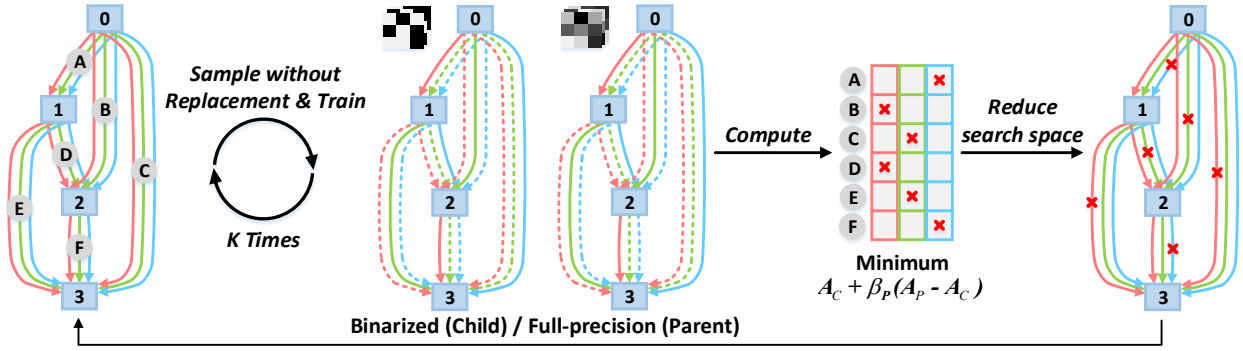



Figure 1: The main framework of the proposed Child-Parent search strategy. In a loop, we first sample the operation without replacement for each edge from the search space, and then train the Child model and Parent model generated by the same architecture simultaneously. Second, we use the Eqs. 1 and 2 to compute the evaluation indicator calculated by the accuracy of both models on the validation dataset. Until all the operations are selected, we remove the operation in each edge with the worst performance.

ciently search a binarized network architecture in a unified framework. The search strategy for Child-Parent model consists of three steps shown in Fig. 1. First, we sample the operations without replacement and construct two classes of sub-networks that share the same architecture, i.e., binarized networks (Child) and full-precision networks (Parent). Second, we train both sub-networks and obtain the performance indicator of the corresponding operations by calculating the child network accuracy and the accuracy loss between child and parent networks. It is observed that the worse operations in the early stage usually have the worse performance at the end. Based on this observation, we then remove the operation with the worst performance according to the performance indicator. This precoess is repeated until there is only one operation left in each edge. For binarized optimization of Child-Parent model, we reformulate the traditional binarization loss as a kernel-level Child-Parent loss.The main contributions of our paper include:

- A Child-Parent model is introduced to guide the binarized architecture search and to optimize BNNs in a unified framework.
- An indicator is proposed to evaluate the operation performance based on Child-Parent model. The search space is greatly reduced through this search strategy for Child-Parent model, which improves the search efficiency significantly.
- Extensive experiments demonstrate the superiority of the proposed algorithm over other light models on the CIFAR-10 and ImageNet datasets.

## 2 Child-Parent NAS

In this section, we first describe the proposed CP-NAS, our Child-Parent model for NAS. Then, the search space and strategy for CP-NAS is introduced to effectively find an powerful binarized architecture. Finally, a kernel-level CP loss is proposed for binarized optimization. The framework of CP-NAS is shown in Fig. 1, and details are provided below.

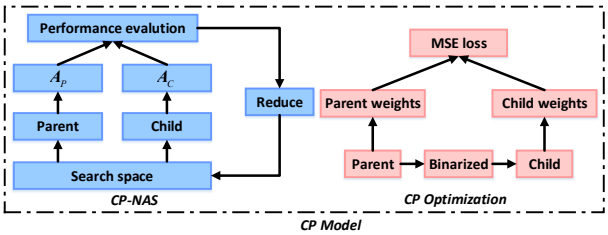



Figure 2: The main framework of Child-Parent model. The Child-Parent model focuses on both the binarized architecture search (left) and binarized optimization (right).

### 2.1 Child-Parent Model for Network Binarization

Network binarization, which calculates neural networks with 1-bit weights and activations to fit the full-precision network, can significantly compress the deep convolutional nerual networks (CNNs). Prior work [Zhao *et al.*, 2019] usually investigates the binarization problem by exploring the full-precision model to guide the optimization of binarized models. Based on the investigation, we reformulate NAS-based network binarization as a Child-Parent model as shown in Fig 2. The binarized model and the full-precision counterpart are the child and parent models respectively.

Conventional NAS is inefficient due to the complicated reward computation in network training where the evaluation of a structure is usually done after the network training converges. There are also some methods to perform the evaluation of a cell during the training of the network. [Zheng *et al.*, 2019] points out that the best choice in early stages is not necessarily the final optimal one, however, the worst operation in the early stages usually has a bad performance at the end. And this phenomenon will become more and more significant as the training goes. Based on this observation, we propose a simple yet effective operation removing process, which is the key task of the proposed CP model.

Intuitively, the difference between the children and parents ability, and how much children can independently handle their problems, are two main aspects that should be considered to define a reasonable performance evaluation measure. Our Child-Parent model introduces a similar perfor-

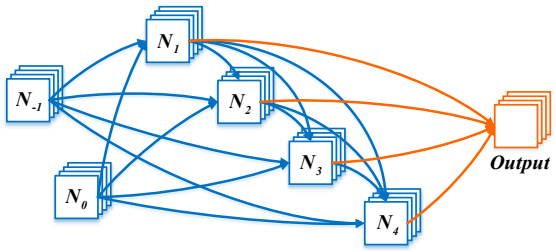



Figure 3: The cell architecture for CP-NAS. One cell includes 2 input nodes, 4 intermediate nodes and 14 edges (blue).

mance indicator to improve the search efficiency. The performance indicator includes two parts, the performance loss between the binarized network (Child) and the full-precision network (Parent), and the performance of the binarized network (Child). We can thuse define it for each operation of the sampled network as

$$z_{k,t}^{(i,j)} = A_{C,t} + \beta_P(A_{P,t} - A_{C,t}) \tag{1}$$

where $A_{P,t}$ and $A_{C,t}$ represent the network performance calculated by the accuracy of the full-precision model (Parent) and the binarized model (Child) on the validation dataset, and $\beta_P$ is the hyper-parameter to control the performance loss. $i$,$j$ represent the index of the node to generate edge $(i, j)$ shown in Fig. 3, $k$ is the operation index of corresponding edge, and $t$ represents the $t$th sampling process. Note that we use the performance of the sampled network to evaluate the performance of the corresponding selected operations.

CP-NAS not only uses the accuracy on the validation dataset to guide the search process directly, but also takes the information of full-precision model into consideration to better investigate the full potential that the binarized model can ultimately reach. Additional details are provided in the following section.

As shown in Fig. 2, unlike the traditional teacher-student model [Hinton *et al.*, 2015] which transfers the generalization ability of the first model to a smaller model by using the class probabilities as "soft targets", the Child-Parent model focuses on the performance measure particularly suitable for NAS-based network binarization. Furthermore, the loss function for the teacher-student model is constrained on the feature map or the output, while ours focuses on the kernel weights to minimize the variations between two networks.

## 2.2 Search Space

We search for computation cells as the building blocks of the final architecture. As in [Zoph and Le, 2016; Zoph *et al.*, 2018; Liu *et al.*, 2018b; Real *et al.*, 2019], we construct the network with a pre-defined number of cells and each cell is a fully-connected directed acyclic graph (DAG) $\mathcal{G}$ with $M$ nodes, $\{N_1, N_2, ..., N_M\}$. For simplicity, we assume that each cell only takes the outputs of the two previous cells as input and each input node has pre-defined convolutional operations for preprocessing. Each node $N_j$ is obtained by $N_j = \sum_{i<j} o^{(i,j)}(N_i)$. $N_i$ is the dependent node of $N_j$ with the constraint $i < j$ to avoid cycles in a cell. We also define nodes $N_{-1}$ and $N_0$ without inputs as the first two nodes of

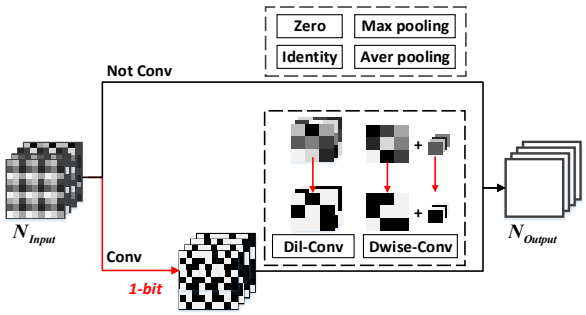



Figure 4: The operations of each edge. Each edge has 4 convolutional operations, including 2 types of binarized convolutions with $3 * 3$ or $5 * 5$ receptive fields, and 4 non-convolutional operations.

a cell. Each node is a specific tensor like a feature map, and each directed edge $(i, j)$ denotes an operation $o^{(i,j)}(.)$ shown in Fig. 4, which is sampled from following $K = 8$ operations:

- no connection (zero)
- skip connection (identity)
- $3 \times 3$ dilated convolution with rate 2
- $5 \times 5$ dilated convolution with rate 2
- $3 \times 3$ max pooling
- $3 \times 3$ average pooling
- $3 \times 3$ depth-wise separable convolution
- $5 \times 5$ depth-wise separable convolution

We replace the convolution with a binarized form. We also remove the ReLU operation to avoid the vanishing of the negative in the 1-bit convolution. The optimization of BNNs is more challenging than that of the conventional CNNs [Rastegari *et al.*, 2016], [Gu *et al.*, 2019], since binarization brings additional computation burdens to NAS.

## 2.3 Search Strategy for CP-NAS

As shown in Fig. 1, we randomly sample one operation from the $K$ operations in $\mathcal{O}^{(i,j)}$ for every edge and then obtain the performance based on Eq. 1 by training the sampled parent and child networks for one epoch. Finally, we assign this performance to all the sampled operations. These steps are performed $K$ times by sampling without replacement, leading to each operation having exactly one accuracy for every edge for fairness.

We repeat the complete sampling process $T$ times. Thus each operation for every edge has $T$ performance measures $\{z_{k,1}^{(i,j)}, z_{k,2}^{(i,j)}, ..., z_{k,T}^{(i,j)}\}$ calculated by Eq. 1. Furthermore, to reduce the undesired fluctuation in the performance evaluation, we normalize the performance of $K$ operations for each edge to obtain the final evaluation indicator as

$$e(o_k^{(i,j)}) = \frac{exp\{\bar{z}_k^{(i,j)}\}}{\sum_{k'} exp\{\bar{z}_{k'}^{(i,j)}\}}, \tag{2}$$

where $\bar{z}_k^{(i,j)} = \frac{1}{T}\sum_t z_{k,t}^{(i,j)}$. Along with the increasing epochs, following [Zheng *et al.*, 2019] and [Chen *et al.*, 2019a], we progressively abandon the worst evaluation operation from each edge until there is only one operation for each edge. The complete algorithm is shown in Alg. 1.

**Algorithm 1** Child-Parent NAS

**Input**: Training data, Validation data

**Parameter**: Searching hyper-graph: $\mathcal{G}$, $K = 8$, $e(o_k^{(i,j)}) = 0$ for all edges

**Output**: Optimal structure $\alpha$

**while** $(K > 1)$ **do**
  **for** $t = 1, ..., T$ epoch **do**
    **for** $e = 1, ..., K$ epoch **do**
      Select an architecture by sampling (without replacement) one operation from $\mathcal{O}^{(i,j)}$ for every edge;
      Construct the Child model and Parent model with the same selected architecture, and then train both models to get the accuracy on the validation data; Use Eq.1 to compute the performance and assign that to all the sampled operations;
    **end for**
  **end for**
  Update $e(o_k^{(i,j)})$ using Eq. 2;
  Reduce the search space $\{\mathcal{O}^{(i,j)}\}$ with the worst performance evaluation by $e(o_k^{(i,j)})$ ;
  $K = K - 1$;
**end while**
**return** solution

### 2.4 Optimization for 1-bit CNNs

Inspired by XNOR and PCNN, we reformulate the binarized optimization as Child-Parent optimization in our unified framework.

To binarize the weights and activations of CNNs, we introduce the kernel-level Child-Parent loss for binarized optimization from two respects. First, we minimize the distributions between the full-precision filters and their corresponding binarized filters. Second, we minimize the intra-class compactness based on the output features. We then have a loss function as

$$\mathcal{L}_{\hat{H}} = \sum_{c,l} \text{MSE}(H_c^l, \hat{H}_c^l) + \frac{\lambda}{2} \sum_{s} \|f_{C,s}(\hat{H}) - \overline{f}_{C,s}(H)\|^2, \tag{3}$$

where $\lambda$ is a hyperparameter to balance the two terms. $H_c^l$ is the $c$th full-precision filter of the $l$th convolutional layer and $\hat{H}_c^l$ denotes its corresponding reconstructed filter; MSE$(\cdot)$ represents the mean square error (MSE) loss. The second term is used to minimize the intra-class compactness, since the binarization process causes feature variations. $f_{C,s}(\hat{H})$ denotes the feature map of the last convolutional layer for the $s$th sample, and $\overline{f}_{C,s}(\hat{H})$ denotes the class-specific mean feature map for corresponding samples. Combining $\mathcal{L}_{\hat{H}}$ with the conventional loss $\mathcal{L}_{CE}$, we obtain the final loss as

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{\hat{H}}. \tag{4}$$

$\mathcal{L}$ and its derivatives are easily calculated by the Pytorch package directly.

## 3 Experiments

In this section, we compare our CP-NAS with the state-of-the-art NAS methods and 1-bit CNNs methods on two publicly available datasets: CIFAR-10 [Krizhevsky *et al.*, 2014] and ILSVRC12 ImageNet [Russakovsky *et al.*, 2015].

### 3.1 Training and Search Details

In our experiments, we first search binarized neural architectures on an over-parameterized network on CIFAR-10, and then evaluate the best architecture with a stacked deeper network on the same dataset. We perform experiments to search binarized architectures directly on ImageNet. We run the experiment multiple times and find that the resulting architectures show only a slight variation in performance, which demonstrates the stability of our method.

We use the same datasets and evaluation metrics as previous NAS works [Liu *et al.*, 2018b; Cai *et al.*, 2018b; Zoph *et al.*, 2018; Liu *et al.*, 2018a]. The color intensities of all images are normalized to $[-1, +1]$. During the architecture search, the training set of the dataset is divided into two subsets, one for training the network weights and the other for perfomrance evaluation as a validation set.

In the search process, we consider a total of 6 cells with the initial 16 channels in the network, where the reduction cell are inserted in the second and the fourth layers, and the others are normal cells. There are $M = 4$ intermediate nodes in each cell. We set $T = 3$ and the initial number of operations $K$ is set to 8, so the final number of search epochs is $(8 + 7 + 6 + 5 + 4 + 3 + 2) * 3 = 105$. $\beta_P$ is set as 2, and the batch size is set to 512. We use SGD with momentum to optimize the network weights, with an initial learning rate of $0.025$ (annealed down to zero following a cosine schedule), a momentum of 0.9, and a weight decay of $5 \times 10^{-4}$. When we search for the architecture directly on ImageNet, we use the same parameters for searching with CIFAR-10 except that the initial learning rate is set to $0.05$ and $\beta_P$ is set to $0.33$. Due to the efficient guidance of CP model, we only use $50\%$ of the training set with CIFAR-10 and ImageNet for architecture search and $5\%$ of the training set for evaluation, leading to a faster search.

After search, in the architecture evaluation step, our experimental settings are similar to [Liu *et al.*, 2018b; Zoph *et al.*, 2018; Pham *et al.*, 2018]. A larger network of 10 cells (8 normal cells and 2 reduction cells) is trained on CIFAR-10 for 600 epochs with a batch size of 96 and an additional regularization cutout [DeVries and Taylor, 2017]. The initial number of channels is set as 56, 72, 112 for different model sizes. We use the SGD optimizer with an initial learning rate of $0.025$ (annealed down to zero following a cosine schedule without restart), a momentum of $0.9$, a weight decay of $3 \times 10^{-4}$ and a gradient clipping at $5$. When stacking the cells to evaluate on ImageNet, the evaluation stage follows that of DARTS [Liu *et al.*, 2018b], which starts with three convolutional layers with a stride of 2 to reduce the input image resolution from $224 \times 224$ to $28 \times 28$. 10 cells (8 normal cells and 2 reduction cells) are stacked after these three layers, with the initial channel number being 102. The network is trained from scratch for 250 epochs using a batch size of 256. We use the SGD optimizer with a momentum of $0.9$, an

| Architecture | Test Error (%) | # Params (M) | W | A | Search Cost (GPU days) | Search Method |
|---|---|---|---|---|---|---|
| WRN-22 [Zagoruyko and Komodakis, 2016] | 5.04 | 4.33 | 32 | 32 | - | Manual |
| DARTS [Liu *et al.*, 2018b] | 2.83 | 3.4 | 32 | 32 | 4 | Gradient-based |
| PC-DARTS [Xu *et al.*, 2019] | 2.78 | 3.5 | 32 | 32 | 0.15 | Gradient-based |
| WRN-22 (PCNN) [Gu *et al.*, 2019] | 5.69 | 4.33 | 1 | 32 | - | Manual |
| BNAS (PCNN) [Chen *et al.*, 2019a] | 3.94 | 2.6 | 1 | 32 | 0.09 | Performance-based |
| BNAS (PCNN, larger) [Chen *et al.*, 2019a] | 3.47 | 4.6 | 1 | 32 | 0.09 | Performance-based |
| WRN-22 (BONN) [Zhao *et al.*, 2019] | 8.07 | 4.33 | 1 | 1 | - | Manual |
| BNAS$^\dagger$ | 8.29 | 4.5 | 1 | 1 | 0.09 | Performance-based |
| **CP-NAS** (Small) | **6.5** | 2.9 | 1 | 1 | 0.1 | Child-Parent model |
| **CP-NAS** (Medium) | **5.72** | 4.4 | 1 | 1 | 0.1 | Child-Parent model |
| **CP-NAS** (large) | **4.73** | 10.6 | 1 | 1 | 0.1 | Child-Parent model |

Table 1: Test error on CIFAR-10. 'W' and 'A' refer to the weight and activation bitwidth respectively. '**M**' means million ($10^6$). BNAS$^\dagger$ is approximately implemented by us by setting $\beta_P = 0$ in CP-NAS, which means that we only use the performance measure for the operation selection.

initial learning rate of $0.05$ (decayed down to zero following a cosine schedule), and a weight decay of $3 \times 10^{-5}$. Additional enhancements are adopted including label smoothing and an auxiliary loss tower during training. All the experiments and models are implemented in PyTorch [Paszke *et al.*, 2017].

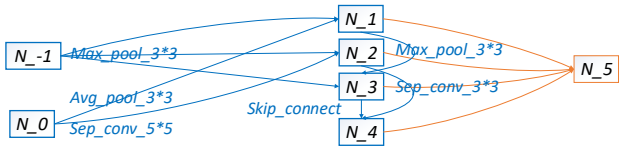
### 3.2 Results on CIFAR-10

We first evaluate our CP-NAS on CIFAR-10 and compare results with both manually designed networks [Zagoruyko and Komodakis, 2016] and networks searched by NAS [Liu *et al.*, 2018b; Xu *et al.*, 2019] at different levels of binarization.

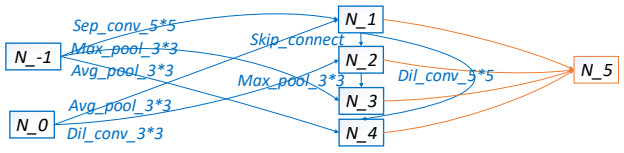
The results for different architectures on CIFAR-10 are summarized in Tab. 1. We search for three binarized networks with different model sizes which binarize both weight and activation. Note that for the model size, in addition to the number of parameters, we should also consider the number of bits of each parameter. The binarized networks only need 1 bit to save and compute the weight parameter or the activation parameter, while the full-precision networks need 32. More details about the efficiency are discussed in section 3.4.

Compared with manually designed networks, e.g., WRN-22(BONN) [Zhao *et al.*, 2019], our CP-NAS achieved comparable or smaller test errors ($6.5\%$ vs. $8.07\%$) and more compressed models ($2.9$M vs. $4.33$M). Compared with full-precision networks obtained by other NAS methods, our CP-NAS achieved comparable test errors and significantly more compressed models, with similar or less search time. Compared with BNAS, our CP-NAS binarized the activation parameters and achieved comparable test errors with only slightly longer search time. We further implement BNAS$^\dagger$ for 1-bit CNNs by setting $\beta_P = 0$ in CP-NAS, which means that we only use the performance measure for the operation selection. The result shows that CP-NAS achieve better performance than BNAS$^\dagger$ with lower test error ($5.72\%$ vs. $8.29\%$) and similar model size. CP-NAS outperforms BNAS in both network efficiency and 1-bit CNNs performance. More detailed comparison with BNAS is presented in section 3.4.

In terms of search efficiency, compared with the previous work PC-DARTS [Xu *et al.*, 2019], our CP-NAS is $30\%$ faster (tested on our platform - 6 NVIDIA TITAN V GPUs). We attribute our superior results to the proposed scheme of search space reduction. As shown in Fig. 5, the architectures of CP-NAS prefer smaller receptive fields. Our CP-NAS also results in more pooling operations, which can increase the nonlinear representation ability of BNNs.




(a) The normal cell




(b) The reduction cell

Figure 5: The normal cell (a) and the reduction cell (b) searched for CIFAR-10.

### 3.3 Results on ImageNet

To further evaluate the performance of our CP-NAS, we compare our method with the state-of-the-art image classification methods on the ImageNet. All the searched networks are obtained directly by CP-NAS on ImageNet by stacking the cells. Due to the first convolutional layer in a depth-wise separable convolution with fewer parameters, we do not binarize the activations of the first layer for ImageNet. Tab. 2 shows the test accuracy on ImageNet. We observe that CP-NAS outperforms manually designed binarized networks ($64.3\%$ vs. $59.5\%$) with a similar number of parameters ($12.5$M vs. $11.17$M). Note that compared to the human-designed full-precision networks, our CP-NAS achieved comparable performance but with higher compression. Furthermore, to obtain a better performance, we do not binarize the activations of the preprocessing operations for the two input nodes, and

| Architecture | Accuracy (%) | | Params | W | A | Search Cost | Search |
|---|---|---|---|---|---|---|---|
| | Top1 | Top5 | (M) | | | (GPU days) | Method |
| ResNet-18 [Gu *et al.*, 2019] | 69.3 | 89.2 | 11.17 | 32 | 32 | - | Manual |
| PNAS [Liu *et al.*, 2018a] | 74.2 | 91.9 | 5.1 | 32 | 32 | 225 | SMBO |
| DARTS [Liu *et al.*, 2018b] | 73.1 | 91.0 | 4.9 | 32 | 32 | 4 | Gradient-based |
| PC-DARTS [Xu *et al.*, 2019] | 75.8 | 92.7 | 5.3 | 32 | 32 | 3.8 | Gradient-based |
| ResNet-18 (PCNN) [Gu *et al.*, 2019] | 63.5 | 85.1 | 11.17 | 1 | 32 | - | Manual |
| BNAS (PCNN) [Chen *et al.*, 2019a] | 71.3 | 90.3 | 6.2 | 1 | 32 | 2.6 | Performance-based |
| ResNet-18 (Bi-real Net) [Liu *et al.*, 2018c] | 56.4 | 79.3 | 11.17 | 1 | 1 | - | Manual |
| ResNet-18 (PCNN) [Gu *et al.*, 2019] | 57.5 | 80.0 | 11.17 | 1 | 1 | - | Manual |
| ResNet-18 (BONN) [Zhao *et al.*, 2019] | 59.3 | 81.5 | 11.17 | 1 | 1 | - | Manual |
| **CP-NAS*** | **64.3** | 85.6 | 12.5 | 1 | 1 | 2.8 | Child-Parent model |
| **CP-NAS**** | **66.5** | 86.8 | 12.5 | 1 | 1 | 2.8 | Child-Parent model |

Table 2: Test accuracy on ImageNet. * represents that we do not binarize the activations of the first convolutional layer in depth-wise separable convolution. ** represents that we do not binarize the activations of preprocessing operations for 2 input nodes either.

achieve an accuracy of 66.5%, which is much closer to the full-precision hand-crafted model, e.g., 69.3% for ResNet-18.

### 3.4 Ablation Study

We test different $\beta_P$ for our method on the CIFAR-10 dataset, as shown in Fig. 6. We can see that when $\beta_P$ increases, the accuracy increases at the beginning, but decreases when $\beta_P \geq 2$. It validates that the performance loss between the Child and Parent models is a significant measure for 1-bit CNNs search. When $\beta_P$ becomes larger, CP-NAS tends to select the architecture with fewer convolutional operations, but a large imbalance between two elements in our CP model will cause a performance drop.

We also compare the architectures obtained by CP-NAS, Random (Random selection), PC (PC-DARTs) and BNAS$^\dagger$ as shown in Fig. 6. Unlike the case of the full-precision model, Random and PC-DARTs lack the necessary guidance, which have a poor performance for binarized architecture search. Both BNAS$^\dagger$ and CP-NAS have the evaluation indicator for the operation selection. Differently, our CP-NAS also considers an additional performance loss, which can outperform the other three strategies.

**Efficiency.** The 1-bit CNNs are extremely efficient for resource-limited devices, showing up to 58× speedup and 32× memory saving than the full-precision models [Rastegari *et al.*, 2016]. As shown in Tab. 3, our CP-NASs (Small, Medium, Large) for CIFAR-10 achieve comparable performance as the full-precision hand-crafted WRN-22 model,

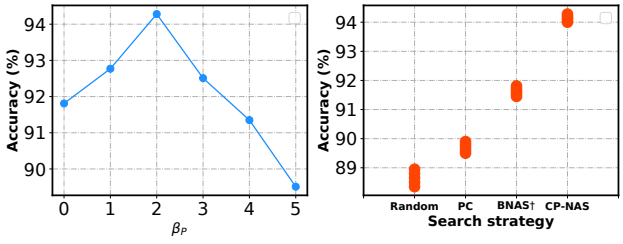



Figure 6: The result (left) for different $\beta_P$ on CIFAR-10. The 1-bit CNNs result (right) for different search strategys on CIFAR-10, including Random (Random selection), PC (PC-DARTs), BNAS$^\dagger$, CP-NAS.

| Architecture | Memory usage (Mbits)↓ | Memory saving↑ | FLOPs (M)↓ | Speedup↑ |
|---|---|---|---|---|
| WRN-22 | 138.27 | 1× | 647.70 | 1× |
| BNAS (PCNN, larger) | ≈5.12 | ≈27× | ≥300 | ≤2.16× |
| WRN-22 (BONN) | 5.71 | 24.19× | 17.03 | 28.03× |
| **CP-NAS** (Small) | 3.32 | 41.56× | 12.89 | 50.24× |
| **CP-NAS** (Medium) | 4.85 | 27.93× | 18.30 | 35.39× |
| **CP-NAS** (large) | 11.51 | 12.01× | 38.67 | 16.75× |

Table 3: Comparison of memory saving and speedup of BNAS (PCNN, larger), WRN-22 (BONN) and CP-NASs (Small, Medium, Large) on CIFAR-10 with respect to WRN-22. ↑ represents that the larger is better, vice versa for ↓.

with 41.56×, 27.93×, 12.01× memory saving and 50.24×, 35.39×, 16.75× speedup in terms of FLOPs. As a result, our CP-NAS models bring significant benefits for resource-contrained edge computing appliations.

## 4 Conclusion

In this paper, we calculate 1-bit CNNs based on the proposed Child-Parent model under the full-precision network supervision. We build a bridge between 1-bit CNNs and NAS using our proposed CP model, leading to the CP-NAS method. With our proposed CP-NAS, we are able to solve the neural architecture search and the binarized optimization in the same framework. Experiments on CIFAR-10 and ImageNet datasets demonstrate that our method achieves better performance than other state-of-the-art methods with a more compressed model and less search time. The future work will focus on more applications, such as object detection and tracking.

## Acknowledgements

Baochang Zhang is also with Shenzhen Academy of Aerospace Technology, Shenzhen, China, and he is the corresponding author. He is in part Supported by National Natural Science Foundation of China under Grant 61672079, Shenzhen Science and Technology Program (No.KQTD20-16112515134654).

## References


[Cai *et al.*, 2018a] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proc. of AAAI*, pages 2787–2794, 2018.

[Cai *et al.*, 2018b] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. *arXiv*, 2018.

[Cai *et al.*, 2018c] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv*, 2018.

[Chen *et al.*, 2019a] Hanlin Chen, Li'an Zhuo, Baochang Zhang, Xiawu Zheng, Jianzhuang Liu, David Doermann, and Rongrong Ji. Binarized neural architecture search. *arXiv*, 2019.

[Chen *et al.*, 2019b] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proc. of ICCV*, October 2019.

[Courbariaux *et al.*, 2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proc. of NIPS*, pages 3123–3131, 2015.

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv*, 2016.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv*, 2017.

[Gu *et al.*, 2019] Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proc. of AAAI*, pages 8344–8351, 2019.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Computer Science*, 14(7):38–39, 2015.

[Juefei-Xu *et al.*, 2017] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proc. of CVPR*, pages 19–28, 2017.

[Krizhevsky *et al.*, 2014] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[Liu *et al.*, 2018a] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proc. of ECCV*, pages 19–34, 2018.

[Liu *et al.*, 2018b] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, 2018.

[Liu *et al.*, 2018c] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proc. of ECCV*, pages 747–763. Springer, 2018.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Proc. of NIPS*, 2017.

[Pham *et al.*, 2018] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv*, 2018.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. of ECCV*, pages 525–542, 2016.

[Real *et al.*, 2019] 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, pages 4780–4789, 2019.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Tan and Le, 2019] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[Xu *et al.*, 2019] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Partial channel connections for memory-efficient differentiable architecture search. *arXiv*, 2019.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proc. of BMVC*, 2016.

[Zhao *et al.*, 2019] Junhe Zhao, Jiaxin Gu, Xiaolong Jiang, Baochang Zhang, Liu Jianzhuang, Guodong Guo, and Rongrong Ji. Bayesian optimized 1-bit cnns. In *Proc. of ICCV*, 2019.

[Zheng *et al.*, 2019] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *Proc. of ICCV*, October 2019.

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

[Zoph *et al.*, 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proc. of CVPR*, pages 8697–8710, 2018.