TOWARDS AUTOMATIC GENERATION OF ADVANCED SHIFT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multiplication-free neural networks significantly reduce the time and energy cost on the hardware platform, as the compute-intensive multiplications are replaced with lightweight bit-shift operations. However, existing shift networks are all directly transferred from state-of-the-art convolutional neural networks (CNNs), which lead to non-negligible accuracy drop or even failure of model convergence. To combat this, we propose `AutoShiftNet`, the first framework tailoring Neural Architecture Search (NAS) to substantially reduce the accuracy gap between bit-shift neural networks and their real-valued counterparts. Specifically, we pioneer dragging NAS into a *shift-oriented search space* and endow it with the robust *topology-related search strategy* and custom *regularization and stabilization*. As a result, our `AutoShiftNet` breaks through the incompatibility of traditional NAS methods for bit-shift neural networks and achieves more desirable performance in terms of accuracy and convergence. Extensive experiments demonstrate that `AutoShiftNet` generates more advanced model architectures for shift networks, where the accuracy increases by (1.69$\sim$8.07)% on CIFAR10, (5.71$\sim$18.09)% on CIFAR100 and $\geq$ 4.36% on ImageNet, especially when many conventional CNNs fail to converge on ImageNet with bit-shift weights.

1 INTRODUCTION

In recent years, large-scale commercial applications based on convolutional neural networks (CNNs) have prompted researchers to design more efficient networks, which can be deployed on platforms with limited resource budgets, such as mobile or IoT devices. Early works utilized network quantization (Cheng et al., 2017) to achieve this goal, by replacing high-precision model parameters with smaller bit-width representations. It can reduce the computational cost of model execution, but also suffer from a non-negligible performance degradation, especially on complex datasets (e.g., ImageNet). To address this issue, recent works (Zhou et al., 2017; Elhoushi et al., 2021) turned to using binary bit shifts rather than simple quantized bits to replace floating-point model parameters.

The key insight of these solutions is that multiplying an element by a power of 2 is mathematically equivalent to a bit-shift operation on it, which is computationally much cheaper and hardware-friendly. Based on this, researchers designed different types of *bit-shift techniques* (Zhou et al., 2017; Elhoushi et al., 2021; Li et al., 2021; 2022), which show promising overhead reduction in model execution. However, all these solutions only focus on designing advanced weight quantization algorithms to reduce the accuracy gap between shift networks and their real-valued counterparts, where the backbone models are all directly transferred from conventional CNNs, e.g., ResNets (He et al., 2016) and VGG (Simonyan & Zisserman, 2014). Given these CNN models are all designed for the continuous real-valued domain, such direct conversion would restrict the potential of bit-shift techniques, causing less optimal network architecture with a non-trivial accuracy drop.

To overcome this limitation, we aim to design advanced shift networks from another perspective, i.e., searching for network architectures that are more compatible with the bit-shift quantization. This is inspired by the Neural Architecture Search (NAS) technique, which can automatically identify the satisfactory network architecture for a given task. The searched models have shown better performance than carefully hand-crafted models (Liu et al., 2018b; Chen et al., 2019). One straightforward way is to directly transfer NAS models searched from real-valued domains to bit-shift

networks. However, similar to the manually-crafted networks, such strategy also leads to sub-optimal results due to the semantic gap between real and bit-shift domains (Sections 3 and 5.4).

For the first time, we present `AutoShiftNet`, a novel methodology to automatically search for the optimal bit-shift network architectures directly, aiming to reduce the accuracy drop from the state-of-the-art real-valued models. Moreover, the introduction of bit-shift operations can significantly reduce the searching, training and inference cost, which can facilitate the deployment of large models on dedicated hardware. Specifically, `AutoShiftNet` contains three components: (1) *Shift-oriented search space*. While existing NAS techniques mainly focus on the real-valued domain, we are the first to construct a new search space composed of bit-shift operations and design the corresponding forward and backward pass. (2) *Topology-related search strategy*. Since shift networks tend to have faster gradient descent or even vanishing gradient (Elhoushi et al., 2021), they are more vulnerable to the conventional gradient-based NAS techniques, i.e., searched networks can be dominated by skip connections (Liu et al., 2018a). Therefore, we decouple the search of model operations and topology, which can efficiently mitigate this issue (Gu et al., 2021). (3) *Search regularization and stabilization*. Given the weight sign freezing effect (Li et al., 2021) and unstable training process, we adopt multiple approaches to regularize and stabilize the search procedure, including shift-adaptive L2 regularization, learning rate reset scheme and shift weight re-parameterization.

We clarify that our work is orthogonal to and different from ShiftAddNAS (You et al., 2022), which aims to search for more accurate models from a hybrid search space with four operations (Attention, Convolution, Shift and Add). Although ShiftAddNAS also considers bit-shift operations, it actually still focuses on multiplication operations as they can provide much higher prediction accuracy. The model searched by ShiftAddNAS is still dominated by multiplications while the shift operations only take a very small part (ShiftAddNAS-T1↑ contains 7.1G multiplications and 8.5G additions, but only 1.4G shifts). Such model cannot be regarded as an actual shiftnet, and is difficult to be deployed on resource-constrained mobile devices, as the number of multiply-add operations is normally restricted below 600M for an ImageNet-mobile setting (Dong & Yang, 2019). In contrast, `AutoShiftNet` totally removes multiplications and only considers efficient bit-shifts and additions. The searched model only contains about 300M additions, so that it is more compatible for the bit-shift domain and also more practical for real-world applications on resource-restricted edge devices.

The networks searched by `AutoShiftNet` show much better performance than conventional CNNs in the bit-shift domain, especially when many CNNs fail to converge on large datasets (e.g., ImageNet) with bit-shift weights. `AutoShiftNet` achieves an accuracy improvement of (1.69∼8.07)% on CIFAR10, (5.71∼18.09)% on CIFAR100 and $\geq$ 4.36% on ImageNet, with more compact parameter sizes and smaller numbers of operation computations. Compared with previous NAS methods, networks from `AutoShiftNet` are more compatible with the bit-shift domain, which lead to a smaller accuracy drop from the complex real-valued models. More importantly, `AutoShiftNet` consumes less computing resources and time as it directly searches with the bit-shift weights.

2 PRELIMINARIES

2.1 BIT-SHIFT NETWORK QUANTIZATION

Bit-shift quantization techniques (Zhou et al., 2017; Elhoushi et al., 2021) round the float-point model weights to the powers of 2, so that the intensive multiplications on weights can be achieved with cheaper binary bit shifts. Formally, given a number x and a rounded model weight 2^p, their multiplication is mathematically equivalent to shifting p bits of x. Since model weights can be either positive or negative for input feature extraction, while 2^p is always positive, a *sign flip* function $flip(w, s)$ is thus introduced to represent the signs of weight values. This operation is achieved with a ternary sign operator $s \in \{-1, 0, +1\}$. Finally, we can replace the weight matrix W in the model as: $W = flip(2^P, S)$, where P is the shift matrix and S is the sign matrix. Both bit shift and sign flip are computationally cheap, as the former is the fundamental operation in modern processors and the latter just computes 2's complement of a number. Therefore, such weight replacement can efficiently reduce the computation cost of CNN model execution.

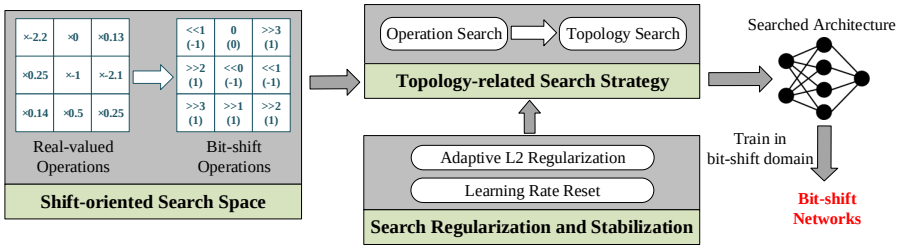


Figure 1: The overview of AutoShiftNet

2.2 Neural Architecture Search

NAS has gained great popularity in recent years, due to its capability of building machine learning pipelines with high efficiency and automation. Early methods used reinforcement learning (Zoph & Le, 2016) and evolutionary algorithms (Real et al., 2019) to search for optimal network architectures for a given task, which normally takes thousands of GPU hours. Recent works tended to use a gradient-based strategy (Liu et al., 2018b) that can reduce the search cost to a few hours. Such methods usually aim at searching for optimal cell structures, since stacking cells as a model is more efficient than searching the whole network architecture. Formally, a cell is represented as a directed cyclic graph (i.e., *supernet*) with N nodes $\{x_i\}_{i=1}^{N}$, including two inputs and one output, and several intermediate nodes. The j-th intermediate node x_j connects to all previous nodes x_i through the edge (i, j). The operation choice over the edge (i, j) can be relaxed as $\overline{o}^{(i,j)}(x) = \sum \alpha_o^{(i,j)} o^{(i,j)}(x_i)$, where $o \in \mathcal{O}$, and $\mathcal{O}$ denotes the search space of candidate operations. $\alpha_o^{(i,j)}$ is the trainable weight for each operation on the edge (i, j), which is normalized with the softmax function. Therefore, the feature map of node x_j can be computed by adding all results from its predecessors x_i: $x_j = \sum \overline{o}^{(i,j)}(x_i)$. Let $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the model loss on the training and validation sets. A bi-level optimization is applied to the operation weight α and network weight w as:

$$\min_{\alpha} \ \mathcal{L}_{val}(w^*(\alpha), \alpha), s.t. \ w^*(\alpha) = arg\min_{w}(\mathcal{L}_{train}(w, \alpha)) \tag{1}$$

The final model architecture can be derived from the trained operation weight α by retaining operations with the largest weight and pruning edges with the smaller weight.

3 Overview of AutoShiftNet

The main idea is to automatically generate well-performed bit-shift networks with high efficiency. Challenges arise when we apply exiting NAS techniques for searching bit-shift networks:

Design of shift-oriented search space. Given that existing NAS methods mainly focus on the real-valued models, their search spaces are also designed for real domain, which is not applicable to bit-shift models. Specifically, a conventional NAS search space normally consists of multiple manually defined operations, such as *dilated convolutions* and *separable convolutions*. To build the shift-oriented search space, we need to transfer these operations from the real domain into the bit-shift domain, in which the forward pass and backward pass need to be carefully designed.

Dominance of skip connections. While most of recent NAS methods adopt the gradient-based search strategy (i.e., DARTS (Liu et al., 2018b)), it has a big drawback: the searched networks are easy to be dominated by skip connections (Chen et al., 2019), as the strategy prefers the fastest way of gradient descent. Unfortunately, searching in the bit-shift domain inherits and amplifies this drawback, which would lead to the "cell collapsing" of searched architectures. Hence, a new search strategy considering both the model operations and topology should be adopted.

Less robust search procedure. Replacing floating-point weights with bit shifts brings fast computations, but also results in the accuracy drop and difficulty of model training. Specifically, the introduced shift parameters and sign flips should be well regularized to avoid errors in the gradient descent. Besides, since bit-shift operations are extremely sensitive to a large learning rate, the selection and scheduling of the learning rate should also be carefully crafted.

We design a novel NAS technique `AutoShiftNet` to address the above challenges. Figure 1 shows the overview of our methodology, which consists of three key components:

- *Shift-oriented search space.* This new search space consists of 8 operations, which are converted from the real domain to bit-shift domain.
- *Topology-related search strategy.* This new strategy considers the optimal combination of model operations and topology, which can efficiently mitigate the dominance of skip connections.
- *Search regularization and stabilisation.* Three approaches are proposed to regularize and stabilize the search procedure: applying a shift-adaptive L2 regularization for shift parameters and resetting the learning rate during search.

4 METHODOLOGY

4.1 SHIFT-ORIENTED SEARCH SPACE

Following previous NAS works (e.g., DARTS (Liu et al., 2018b)), we adopt 8 operations as our operation search space $\mathcal{O}$: 3×3 and 5×5 dilated convolutions, 3×3 and 5×5 separable convolutions, 3×3 max pooling, 3×3 average pooling, identity (skip) and the zero[1]. To construct a shift-oriented search space, we group and transfer these operations into the bit-shift domain and study the corresponding forward and backward pass computations.

Grouping candidate operations. Since not every candidate operation needs to be transferred into the bit-shift version, e.g., the identity and pooling, we first divide 8 candidate operations (excluding *zero*) into two groups. The first group $\mathcal{O}_c$ contains four *convolution* operations, which involve dense multiplications. The second group $\mathcal{O}_t$ contains the remaining operations, which mainly focus on the model topology, such as *skip* and *pooling*. The entire search space is denoted as $\mathcal{O}=\{\mathcal{O}_c,\mathcal{O}_t\}$. To construct the shift-oriented search space, we just need to transfer operations in $\mathcal{O}_c$ into the bit-shift domain, and keep operations in $\mathcal{O}_t$ unchanged. Note that this operation group scheme will also be adopted in the topology-related search strategy (Section 4.2).

Replacement of operation weights. As introduced in Section 2.1, quantization of bit-shift networks can be implemented by replacing the floating-point model weights with two parameters: bit shift P and sign flip S. Hence, the weights w of operations in $\mathcal{O}_c$ need to be replaced with the trainable parameters (P,S), which is formulated as below:

$$\overline{P}=round(P),\quad \overline{S}=sign(round(S)),\quad w=flip(2^{\overline{P}},\overline{S}) \tag{2}$$

where $\overline{P}$ is the rounded shift matrix and $\overline{S}$ is the rounded sign matrix. Note that the function *sign* generates a ternary value, and can be represented as:

$$sign(s)=\begin{cases}-1 & if\ \ s\le-0.5\\ 0 & if\ \ -0.5<s<0.5\\ +1 & if\ \ s\ge0.5\end{cases} \tag{3}$$

Designing forward and backward pass. Different from some previous works (Zhou et al., 2017) which just rounded the trained models into the bit-shift domain, our goal is to directly search and train the model in the shift domain. So we need to design and implement the forward and backward pass of shift operations. With the transferred weights $w=flip(2^{\overline{P}},\overline{S})$, the forward pass for convolutions in $\mathcal{O}_c$ can be represented as: $Y=w*X=flip(2^{\overline{P}},\overline{S})*X+b$, where (X,Y) denote the operation input and output, and b denotes the bias. The gradients of the backward pass can be formulated as:

$$\frac{\partial\mathcal{L}}{\partial X}=\frac{\partial\mathcal{L}}{\partial Y}\frac{\partial Y}{\partial X}=\frac{\partial\mathcal{L}}{\partial Y}w^T,\ \ \frac{\partial\mathcal{L}}{\partial P}=\frac{\partial\mathcal{L}}{\partial Y}\frac{\partial Y}{\partial w}\frac{\partial w}{\partial\overline{P}}\frac{\partial\overline{P}}{\partial P},\ \ \frac{\partial\mathcal{L}}{\partial S}=\frac{\partial\mathcal{L}}{\partial Y}\frac{\partial Y}{\partial w}\frac{\partial w}{\partial\overline{S}}\frac{\partial\overline{S}}{\partial S},\ \ \frac{\partial\mathcal{L}}{\partial b}=\frac{\partial\mathcal{L}}{\partial Y} \tag{4}$$

where $\mathcal{L}$ denotes the model loss.

We use the straight through estimators (Yin et al., 2019) to compute the derivatives of the round and sign functions as: $\frac{\partial round(x)}{\partial x}\approx1$ and $\frac{\partial sign(x)}{\partial x}\approx1$. For the sign flip function, we have: $\frac{\partial flip(x,s)}{\partial x}\approx flip(x,s)$ and $\frac{\partial flip(x,s)}{\partial s}\approx1$. With these estimations, we can set $\frac{\partial\overline{P}}{\partial P}\approx1$ and $\frac{\partial\overline{S}}{\partial S}\approx1$,

[1]Zero means no connection between two nodes.

and then obtain the following expressions:

$$\begin{aligned}\frac{\partial w}{\partial \overline{S}} &= \frac{\partial flip(2^{\overline{P}}, \overline{S})}{\partial \overline{S}} \approx 1 \\ \frac{\partial w}{\partial \overline{P}} &= \frac{\partial flip(2^{\overline{P}}, \overline{S})}{\partial \overline{P}} = \frac{\partial flip(2^{\overline{P}}, \overline{S})}{\partial 2^{\overline{P}}} \frac{\partial 2^{\overline{P}}}{\partial \overline{P}} \\ &\approx flip(2^{\overline{P}}, \overline{S}) 2^{\overline{P}} ln2 = w 2^{\overline{P}} ln2\end{aligned} \tag{5}$$

As a result, the gradients of the trainable parameters (P, S) with respect to the model loss $\mathcal{L}$ are:

$$\frac{\partial \mathcal{L}}{\partial P} \approx \frac{\partial \mathcal{L}}{\partial Y} \frac{\partial Y}{\partial w} w 2^{\overline{P}} ln2, \quad \frac{\partial \mathcal{L}}{\partial S} \approx \frac{\partial \mathcal{L}}{\partial Y} \frac{\partial Y}{\partial w} \tag{6}$$

Based on the above constructed forward and backward pass of bit-shift operations, we can achieve searching and training a NAS model in the bit-shift domain.

4.2 Topology-related Search Strategy

The dominance of skip connections caused by the gradient-based search strategy is a major restriction for applying NAS techniques to quantized networks (Bulat et al., 2020). Besides, ignoring the model topology during a search in some NAS methods also limits the generation of optimal network architectures. Hence, we determine to decouple the operation search and topology search. This search strategy can efficiently suppress the dominance of skip-connections and also improve the performance of searched networks.

Operation search. As introduced in Section 4.1, the 8 candidate operations in the shift-oriented search space can be divided into two groups: $\mathcal{O}_t$ contains topology-related operations that can explicitly affect the model topology (e.g., *skip*), while operations in $\mathcal{O}_c$ do not have such impact. Therefore, the operation search space $\mathcal{O}$ is split into two subspaces $\mathcal{O} = \{\mathcal{O}_t, \mathcal{O}_c\}$, and each operation subspace is relaxed to be continuous independently. Then a bi-level optimization is applied to train the model weight w and operation weight α. With the trained α, we retain the operation with the maximum weight in each operation subspace, which can be formulated as:

$$o_t^{(i,j)} = arg \max_{o_t \in \mathcal{O}_t} \alpha_{o_t}^{(i,j)}, \quad o_c^{(i,j)} = arg \max_{o_c \in \mathcal{O}_c} \alpha_{o_c}^{(i,j)} \tag{7}$$

Such group operation scheme can avoid the elimination of potential topology choices during the operation search, which then allows the subsequent topology search to find out the optimal topology. Finally, all the retained operations are collected to construct a new operation search space $\mathcal{O}_N = \{o_t^{(i,j)}, o_c^{(i,j)}\}$ on each edge (i, j), which is used for the topology search.

Topology search. The previous operation search step aims to determine the best operations on each edge. In this topology search step, we try to search for the optimal combinations of model edges. It can well prevent *skips* from dominating the searched model topology.

First, a topology search space is constructed. Following previous works, we restrict two input edges for each node in the cell *supernet*, so the topology search space $\mathcal{E}_{x_j}$ for node x_j can be represented as a set of all possible pairwise combinations of its incoming edges: $\mathcal{E}_{x_j} = \{\langle (i_1, j), (i_2, j) \rangle | 0 < i_1 < i_2 < j\}$. The topology search space contains $C_n^2 = \frac{n!}{2!(n-2)!}$ candidates, where n denotes the number of incoming edges for node x_j. Similar to the operation search, we also relax the topology search space $\mathcal{E}_{x_j}$ to be continuous:

$$\beta_{x_j}^c = \frac{exp({\beta'}_{x_j}^c / T_\beta)}{\sum_{c' \in \mathcal{E}_{x_j}} exp({\beta'}_{x_j}^{c'} / T_\beta))} \tag{8}$$

where $\beta_{x_j}^c$ is the topology weight that denotes the normalized probability of the edge combination $c \in \mathcal{E}_{x_j}$. $T_\beta(t) = T_0 \theta^t$ is the temperature for architecture annealing, which can efficiently bridge the optimization gap between the supernet and child networks (Xie et al., 2018).

Then, the importance weight $\gamma^{(i,j)}$ for each edge (i, j) can be computed from those combinations containing this edge, which can be formulated as:

$$\gamma^{(i,j)} = \sum_{c \in \mathcal{E}_{x_j}, (i,j) \in c} \frac{1}{N(c)} \beta_{x_j}^c \tag{9}$$

where $N(c)$ is the number of edges in the edge combination c. As a result, the feature map of node x_j can be obtained by summing all the incoming edges weighted by the edge importance weight $\gamma^{(i,j)}$:

$$x_j = \sum_{i<j} \gamma^{(i,j)} \overline{o}^{(i,j)}(x_i) \tag{10}$$

where $\overline{o}^{(i,j)}(x_i)$ denotes the mixed operations on edge (i, j) obtained from the operation search. In the topology search, as the number of candidate operations is largely reduced (i.e., 2 in $\mathcal{O}_N$), we can directly use the one-level optimization to update three weights (w, α, β) in the search.

Determining the architecture. After the operation and topology search, we select the edge combination c with the maximal weight in topology weight β to construct the model topology, and then attach to each edge the operation with the maximal weight in the operation weight α.

4.3 Search Regularization and Stabilisation

Based on the shift-oriented search space and topology-related search strategy, an efficient bit-shift network architecture can be identified for each specific task automatically. However, the adoption of bit-shift weights makes the architecture search much more unstable and also leads to more difficult model training. The search process usually converges to a sub-optimal solution, sometimes even cannot converge. So we need to regularize and stabilize the optimization of the three trainable weights during search: network weight w, operation weight α and topology weight β.

For the optimization of the network weight w, note that w consists of the bitwise shift P and sign flip S, i.e., $w \leftarrow \{P, S\}$. We use an *adaptive L2 regularization* term to regularize the gradient descent of P, which is defined as $\sum W^2 = \sum(2^P S)^2$ rather than the conventional formulation $\sum(P^2 + S^2)$. While most weights in a trained model are rarely larger than 1 (i.e., $|2^P| < 1$), the range of the value of P is also empirically set to be smaller than 0. As a negative parameter, a smaller P instead leads to a larger P^2, which gives a reverse activation to the training loss. Hence, the regularization term should be modified to avoid misguiding the direction of the gradient descent. Formally, the regularized loss $\mathcal{L}'$ can be formulated as: $\mathcal{L}' = \mathcal{L} + \frac{\lambda}{2}\sum(2^P S)^2$, where $\mathcal{L}$ denotes the original model loss and λ is the fixed weight decay. Our experiments in Section 5.4 show that this adaptive L2 regularization improves the accuracy of searched architectures.

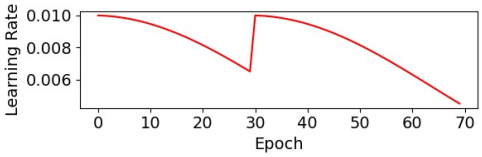


Figure 2: Learning rate curve in the search

To stabilize the optimization of the operation weight α and topology weight β, in addition to using the temperature regularization in Eq.(8), we also carefully implement a *learning rate reset* scheme. Since bit-shift networks are extremely sensitive to large learning rates, we need to use a much smaller initial learning rate than that in previous NAS techniques to avoid model convergence failure. Besides, while previous works (Gu et al., 2021) adopt the annealed learning rate from the previous operation search step for following topology search, we find that resetting the learning rate to an initial value again at the start of topology search allows to obtain a better network architecture. Figure 2 shows the learning rate curve in the search with the cosine annealing: the learning rate is reset at the 30*th* epoch, when the topology search starts.

5 Evaluation

We implement `AutoShiftNet` with Pytorch. Following previous works (Elhoushi et al., 2021; Zhou et al., 2017), we emulate the precision of an actual bit-shift hardware implementation by rounding the operation input and bias to the 32-bit fixed-point format precision (16-bit for the integer part and 16-bit for the fraction part). The shift parameter P is constrained in [-15, 0], i.e., the absolute value of the model weight is within $[2^{-15}, 1]$, which only needs 4 bits to represent. The model weight also needs an extra bit to denote its sign S.

We run evaluations on CIFAR10, CIFAR100 and ImageNet datasets. We comprehensively compare `AutoShiftNet` with a variety of state-of-the-art CNN models (e.g., ResNet, VGG, MobileNet, ShuffleNet, GoogleNet, SqueezeNet) and NAS models (e.g., NASNet, AmoebaNet, DARTS, GDAS, DOTS). For fair comparisons, these baseline models are trained in the bit-shift domain, unless otherwise specified.

Architecture	Top-1 Acc. (%) CIFAR10	Params (M) CIFAR10	Top-1 Acc. (%) CIFAR100	Params (M) CIFAR100	Search Cost (GPU-days)	Search Method
ResNet18 (He et al., 2016)	93.20	11.2	69.11	11.2	-	-
ResNet20 (He et al., 2016)	88.84	0.3	60.12	0.3	-	-
ResNet50 (He et al., 2016)	93.89	23.6	70.64	23.6	-	-
ResNet56 (He et al., 2016)	91.11	0.9	65.57	0.9	-	-
ResNet101 (He et al., 2016)	93.43	42.8	69.18	42.8	-	-
ResNet152 (He et al., 2016)	93.17	58.5	65.58	58.5	-	-
MobileNet-v2 (Sandler et al., 2018)	92.64	2.4	70.24	2.4	-	-
VGG19 (Simonyan & Zisserman, 2014)	91.57	20.1	64.88	20.1	-	-
ShuffleNet-v2 (Ma et al., 2018)	87.51	0.5	58.26	0.5	-	-
NASNet (Zoph et al., 2018)	95.28	3.3	75.33	3.3	1800	RL
AmoebaNet (Real et al., 2019)	95.22	2.3	75.05	2.3	3150	EA
DARTS-v1 (Liu et al., 2018b)	94.39	3.2	74.93	3.2	0.4	GD
DARTS-v2 (Liu et al., 2018b)	94.80	3.5	75.17	3.5	0.4	GD
GDAS (Dong & Yang, 2019)	94.62	2.5	74.87	3.4	0.2	GD
P-DARTS (Chen et al., 2019)	94.21	3.4	74.54	3.6	0.3	GD
DARTS- (Xu et al., 2019)	93.87	3.4	70.85	3.5	0.4	GD
DOTS (Gu et al., 2021)	95.12	3.7	75.05	4.2	0.3	GD
`AutoShiftNet` (Best)	**95.58**	3.3	**76.35**	3.8	0.23*	GD
`AutoShiftNet` (Avg)†	95.43±0.12	3.3	76.08±0.23	3.8	0.23*	GD

Table 1: Evaluation results on CIFAR10/100. The results of conventional CNNs are obtained by running open code of DeepShift (Elhoushi et al., 2021). †: The results are computed from four individual runs with random seeds. *: The search cost can be much smaller on the dedicated hardware as we emulate the bit-shift operations with software.

5.1 Evaluation on CIFAR

Search settings. The entire search process on CIFAR 10/100 consists of two steps: operation search for 30 epochs and then topology search for 40 epochs. The network skeleton consists of 8 cells (6 normal cells and 2 reduction cells) with the initial channel size of 16. The learning rate is scheduled from 0.01 following the reset scheme in Section 4.3. The search process takes about 5.5 hours on one GeForce RTX 3090 GPU. However, since we emulate the hardware bit-shift operations with software implementation, the search time actually can be significantly shortened on the dedicated hardware platforms. We will discuss more about the search efficiency in Section 5.5. The best cells searched from CIFAR are shown in Appendix C.

Evaluation settings. The evaluation network is composed of 20 cells, including 18 normal cells and 2 reduction cells. We set the initial channel size as 36 and optimize the network via the RAdam optimizer (Liu et al., 2019) with an initial learning rate of 0.01 (cosine annealing to 0) and weight decay of $3e$-4. Following the setting in DeepShift, the netowrk is trained from scratch with bit-shift weights for 200 epochs. The batch size is set as 128. Cutout and drop-path with a rate of 0.2 are used to prevent overfitting. The training accuracy curves can be found in Appendix D.

Results analysis. Table 1 shows the evaluation results on CIFAR 10/100 datasets. The bit-shift networks searched by `AutoShiftNet` achieve 95.58% and 76.35% accuracy on CIFAR10 and CIFAR100, respectively. Compared to conventional manually designed CNNs, `AutoShiftNet` models lead to a significant performance improvement in the bit-shift domain, where the prediction accuracy increases (1.69∼8.07)% on CIFAR10 and (5.71∼18.09)% on CIFAR100. Moreover, the parameter size of searched networks is also much smaller than most conventional CNNs. Hence, in contrast to directly transferring those CNNs into bit-shift counterparts, `AutoShiftNet` is a more efficient approach to generate high-quality bit-shift networks, with the improved accuracy, reduced parameter size and automatic design process. We also compare `AutoShiftNet` with state-of-the-art NAS techniques searched in the real domain, and the results show that our method can find out architectures more compatible to the bit-shift domain. We will discuss more details in Section 5.3.

5.2 Evaluation on ImageNet

Evaluation settings. Following previous works (Liu et al., 2018b; Dong & Yang, 2019), we construct the network for ImageNet with the best cells searched from the CIFAR dataset. The evaluation follows the ImageNet-mobile setting, in which the input size is 224×224. The network consists of 14 cells (12 normal cells and 2 reduction cells) with the initial channel size of 46. We train the network in the bit-shift domain for 90 epochs with a batch size of 1024. The RAdam optimizer with an initial learning rate of 0.01 (warming up in the first 5 epochs and cosine annealing to 0) is used. The training accuracy curves can be found in Appendix D.

Results analysis. Table 2 shows the evaluation results on the ImageNet dataset. It can be found that although some conventional CNNs (e.g., ResNet) still perform well when converted to the

Architecture	Acc. (%)		Params	Multi	Add
	Top-1	Top-5	(M)	(M)	(M)
ResNet18	62.25	83.79	11.7	0	987
ResNet50	69.04	88.61	25.8	0	2053
VGG16*	0.10	0.98	138.5	0	8241
GoogleNet	62.81	84.81	6.6	0	752
MobileNet-v2*	40.03	65.13	4.7	0	206
ShuffleNet-v2*	37.32	62.26	7.4	0	306
SqueezeNet1_0	29.08	51.96	3.8	0	412
NASNet	66.24	86.24	5.6	0	317
DARTS-v2	64.98	85.18	4.7	0	287
GDAS	65.87	85.95	5.3	0	291
DOTS	66.36	86.23	5.2	0	302
AutoShiftNet (Ours)	67.17	87.38	5.1	0	298

Table 2: Evaluation results on ImageNet. *: The results are the highest accuracy in the training while networks fail to converge.

Architecture	Domain	Acc. (%)			
		C10	Diff.	C100	Diff.
ResNet18	R	94.45	-	72.53	-
	BS	93.20	-1.25	69.11	-3.42
ResNet50	R	95.12	-	74.19	-
	BS	93.89	-1.23	70.65	-3.54
DARTS(v2)	R	96.48	-	78.78	-
	BS	94.80	-1.68	75.17	-3.61
DARTS-	R	95.61	-	76.02	-
	BS	93.87	-1.74	70.85	-5.17
DOTS	R	96.55	-	78.87	-
	BS	95.13	-1.42	75.05	-3.82
AutoShiftNet	R	96.19	-	78.26	-
	BS	**95.58**	**-0.61**	**76.35**	**-1.91**

Table 3: Accuracy of various architectures on CIFAR10 (C10) and CIFAR100 (C100) in the real (R) and bit-shift (BS) domains.

bit-shift domain, there are many more state-of-the-art CNNs giving much lower prediction accuracy or even failing to converge, including VGG16, MobileNet-v2 and ShuffleNet-v2, whose final top-1 accuracy drops to 0.09%, 1.18% and 9.27%, respectively. In contrast, AutoShiftNet can converge robustly and achieve 67.17% top-1 accuracy, which is (4.36∼67.07)% higher than conventional CNNs except ResNet50. Note that the high accuracy of ResNet50 is obtained at the price of much larger parameter size (5×) and more operations (7×). Hence, compared to conventional CNNs, bit-shift networks searched by AutoShiftNet perform better with fewer parameters and operations. The comparison with previous NAS techniques also shows that AutoShiftNet can generate more compatible architectures for bit-shift networks. Given all multiplications in networks are replaced with bit shifts, the number of multi-operations would be 0, which greatly reduces the resource cost and speeds up the model inference.

5.3 Real-valued and Bit-shift Network Comparisons

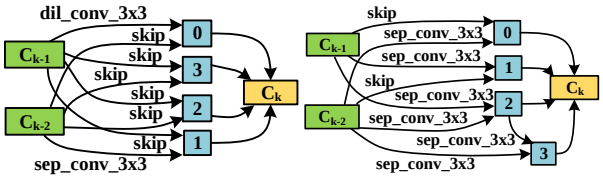
We compare the accuracy of the same network trained in the real and bit-shift domains, aiming to investigate the accuracy drop of conventional CNNs and NAS models caused by the bit-shift quantization. Table 3 shows the results of some representative networks on the CIFAR datasets. Comparison on ImageNet can be found in Appendix E. We can observe that AutoShiftNet not only achieves the highest accuracy of bit-shift networks, but also leads to the smallest accuracy drop (-0.61% and -1.91%) when the network is quantized from the real to bit-shift domains. In comparison, conventional CNNs have lower accuracy in the real domain, and the accuracy drops more significantly during the bit-shift quantization.

We further compare AutoShiftNet with previous NAS techniques. From Table 3, AutoShiftNet is able to obtain network architectures with better performance in the bit-shift domain, even their accuracy in the real domain is slightly lower. It indicates that transferring existing NAS models directly to the corresponding bit-shift networks normally just achieves sub-optimal solutions. The networks searched by AutoShiftNet are more compatible to the bit-shift quantization.

5.4 Ablation Study

Impact of the shift-oriented search space. The superiority of AutoShiftNet in the bit-shift domain actually has indicated the effectiveness of the shift-oriented search space, which avoids converging to sub-optimal solutions for searching bit-shift network architectures. To further validate the importance of this new search space, we replace the search space with the classical real-valued one in AutoShiftNet, and then check the performance of the searched results. Four experiments are run individually with random seeds, where the searched architectures achieve average accuracy of 94.97% on CIFAR10 and 75.03% on CIFAR100. It drops 0.63% and 1.32% from that with the shift-oriented search space. Besides, as a by-product, the shift-oriented search space significantly reduces the resource cost in the search process, as it replaces dense multiplications with much cheaper bit shifts. Hence, AutoShiftNet can generate better bit-shift networks automatically with much less resource budget.

Impact of the topology-related search strategy. We take DARTS as the baseline strategy to derive cell structures from the shift-oriented search space. The result is shown in Figure 3a. It can be seen that the searched cell is dominated by the skip connections and only achieves 69.58% accuracy on



(a) DARTS cell (b) `AutoShiftNet` cell

Figure 3: Normal cells searched from the bit-shift domain

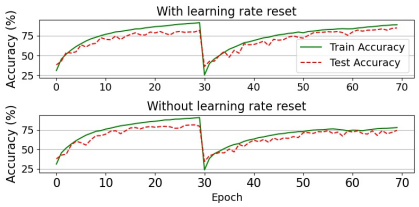


Figure 4: Search accuracy on CIFAR10

CIFAR100. This is because the drawback of the traditional gradient-based search strategy is amplified in the bit-shift domain. By integrating our topology-related search strategy, this drawback can be effectively mitigated and the searched result is shown in Figure 3b. Since the edge connections are further inspected, the topology-related search strategy can generate more stable architectures and achieve 76.21% accuracy, which is 6.63% improvement over DARTS.

Impact of regularization and stabilization. To evaluate the effectiveness of our modified *L2 regularization (L2R)* and *learning rate reset (LRR)* schemes, we compare the performance of networks searched with various scheme combinations (Table 4). We find that while both schemes increase the accuracy of the searched architecture, LRR contributes more than L2R. Figure 4 shows the accuracy curves of the search process on CIFAR10 with or without LRR. It shows that LRR scheme significantly improves the model accuracy from 74.58% to 84.68%, which makes it more possible to search for better bit-shift networks. Note that at the start of topology search (the 30*th* epoch), the model gets pruned and retrained, so the accuracy has a sharp drop.

ID	Scheme		Acc. (%)			
	L2R	LRR	C10	Diff.	C100	Diff.
1	✓	✓	95.58	-	76.35	-
2	✓	✗	95.17	-0.41	73.86	-2.49
3	✗	✓	95.43	-0.15	74.93	-1.42
4	✗	✗	94.91	-0.67	73.04	-3.31

Table 4: Accuracy of various schemes.

5.5 Efficiency Analysis

Given that modern computer architectures use the binary format to store and calculate data, bitwise operations like bit shift and addition are the atomic units for performing complex computations, including the multiplication. According to (Agner Fog), the floating-point multiplication takes at least 5× of clock cycles than the bit shift. Besides, compared to the hardware implementation of bit shift on the circuit, the multiplier takes at least 9.7× of average power, 1.45× of area and 4.32× of transistors (Asati, 2009). Hence, by replacing floating-point weights with bit shift and sign flip operations, the efficiency of architecture search can be significantly improved over previous NAS techniques that search in the real domain. While our software emulation of `AutoShiftNet` just takes 5.5 hours, where the bit shift is simulated by multiplying the power of 2, the actual search cost on the dedicated hardware platforms (e.g., FPGA accelerators) would be largely decreased. We deem that accelerating the NAS process with bit shift on the FPGA board is a promising research direction. Besides, since the searched architectures are trained as bit-shift networks, it also reduces the resource cost and time of model training and inference. `AutoShiftNet` also greatly compresses the storage size of searched networks, as it represents model weights with fewer bits (i.e., 5 bits). This promotes the applications of NAS models on the edge devices, where the memory storage and energy consumption are the main constraints.

6 Conclusion and Future Work

In this paper, we propose to automatically generate advanced bit-shift networks with a dedicated NAS method `AutoShiftNet`. We overcome the challenges of applying existing NAS techniques in the bit-shift domain with three innovations: shift-oriented search space, topology-related search strategy and search regularization and stabilization. Experimental results show that `AutoShiftNet` can search for architectures with higher compatibility for bit-shift operations, and better performance than state-of-the-art CNNs and NAS models.

While replacing model multiplications with bit shifts can efficiently reduce the running cost, it is essentially a coarse-grained representation of model weights, which naturally results in the non-trivial drop of prediction accuracy. To address this, we can further introduce additions into the search space of `AutoShiftNet`, which are also efficient substitutes of multiplications (Chen et al., 2020) and more importantly, can achieve finer-grained weight manipulation (You et al., 2020). Since current CUDA kernels lack optimization of intensive additions, we leave it as future work.

REFERENCES

Agner Fog. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for intel, amd and via cpus. `https://www.agner.org/optimize/instruction_tables.pdf`. Online; accessed 14 January 2022.

Abhijit Rameshwar Asati. *A Comparative Study of High Performance CMOS Multipliers, Barrel Shifters and Modeling of NBTI Degradation in Nanometer Scale Digital VLSI Circuits*. PhD thesis, BITS Pilani, 2009.

Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. Bats: Binary architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pp. 309–325. Springer, 2020.

Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1468–1477, 2020.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1294–1303, 2019.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

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

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2359–2368, 2021.

Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12311–12320, 2021.

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.

Xinlin Li, Bang Liu, Yaoliang Yu, Wulong Liu, Chunjing Xu, and Vahid Partovi Nia. S^3: Sign-sparse-shift reparametrization for effective training of low-bit shift networks. *Advances in Neural Information Processing Systems*, 34:14555–14566, 2021.

Xinlin Li, Bang Liu, Rui Heng Yang, Vanessa Courville, Chao Xing, and Vahid Partovi Nia. Denseshift: Towards accurate and transferable low-bit shift network. *arXiv preprint arXiv:2208.09708*, 2022.

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 *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018a.

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

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

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.

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

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

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. *arXiv preprint arXiv:2010.12785*, 2020.

Haoran You, Baopu Li, Shi Huihong, Yonggan Fu, and Yingyan Lin. Shiftaddnas: Hardware-inspired search for more accurate and efficient neural networks. In *International Conference on Machine Learning*, pp. 25566–25580. PMLR, 2022.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

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.

A ARCHITECTURE SEARCH DETAILS

For the operation search, the official CIFAR training dataset is divided into two halves: training set $\mathbb{D}_T$ and validation set $\mathbb{D}_V$, which are used to optimize network weights w and operation weights α, respectively. The topology search directly uses the whole official training set to optimize the topology weight β with one-level optimization, where the initial temperature T_0 is set as 10 and decay to 0.02. We adopt Rectified Adam (RAdam) optimizer with initial learning rate of 0.01 and weight decay of 3e-4 to optimize model weight w and Adam optimizer with initial learning rate of 3e-4 and weight decay of 1e-3 to optimize operation weight α and topology weight β. The learning rate is scheduled with cosine scheduler following our proposed learning rate reset scheme. The search process consists of 70 epochs with the batch size of 128, including 30 epochs for operation search and 40 epochs for topology search.

B ARCHITECTURE EVALUATION DETAILS

Training on CIFAR. We train the evaluation network for 200 epochs with the batch size of 128. The network is optimized by RAdam optimizer with initial learning rate of 0.01 and weight decay of 3e-4. The learning rate is scheduled by a cosine annealing scheduler to 0. Cutout and drop-path with a rate of 0.2 are used for preventing overfitting.

Training on ImageNet. The network is trained by 90 epochs with the batch size of 1024. The RAdam optimizer is adopted, whose initial learning rate is set as 0.01 and weight decay is set as 3e-4. The learning rate is cosine annealed to 0. Label smoothing and an auxiliary loss tower is used to enhance model training.

Dataset	Cell	Node	Genotype
CIFAR10	Normal Cell	1	('skip_connect', 0), ('skip_connect', 1)
		2	('sep_conv_3x3', 0), ('sep_conv_3x3', 1)
		3	('sep_conv_3x3', 0), ('sep_conv_3x3', 1)
		4	('sep_conv_3x3', 0), ('dil_conv_5x5', 4)
	Reduction Cell	1	('skip_connect', 0), ('skip_connect', 1)
		2	('sep_conv_3x3', 0), ('max_pool_3x3', 1)
		3	('sep_conv_3x3', 0), ('sep_conv_5x5', 1)
		4	('skip_connect', 0), ('dil_conv_5x5', 2)

Table 5: Genotype of Best Archtiecture on CIFAR10

Dataset	Cell	Node	Genotype
CIFAR100	Normal Cell	1	('sep_conv_3x3', 0), ('skip_connect', 1)
		2	('skip_connect', 0), ('sep_conv_3x3', 1)
		3	('sep_conv_3x3', 0), ('sep_conv_3x3', 1)
		4	('sep_conv_3x3', 0), ('sep_conv_5x5', 4)
	Reduction Cell	1	('max_pool_3x3', 0), ('skip_connect', 1)
		2	('sep_conv_5x5', 0), ('sep_conv_5x5', 1)
		3	('max_pool_3x3', 0), ('dil_conv_5x5', 3)
		4	('sep_conv_5x5', 0), ('sep_conv_3x3', 3)

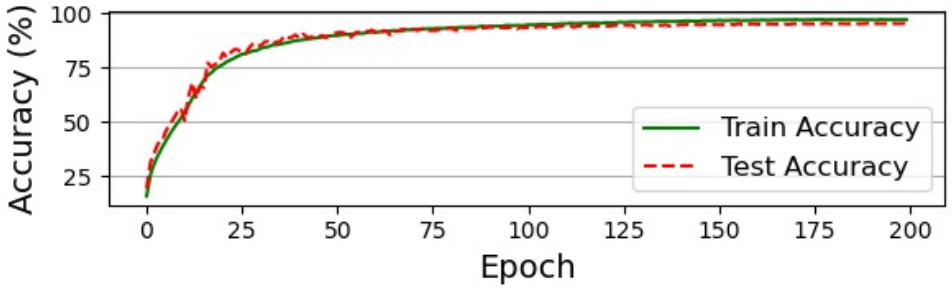
Table 6: Genotype of Best Archtiecture on CIFAR100

C BEST SEARCHED CELL STRUCTURES

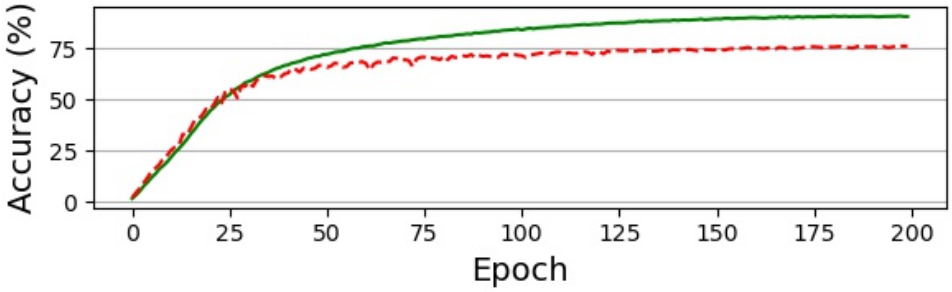
Table 5 and 6 show the best searched architectures for CIFAR10 and CIFAR100. The evaluation on ImageNet adopts cells searched from CIFAR10 (Table 5).

D TRAINING RESULTS

Figure 5 shows the accuracy traces of training on CIFAR10 and CIFAR100. Figure 6 shows the accuracy traces of training on ImageNet, where (a) takes batch size of 1024 and (b) takes 256. It can be seen that training with batch size of 256 converges earlier and is also more stable, where the final top-1 accuracy is slightly higher (68.67% vs. 67.17%).

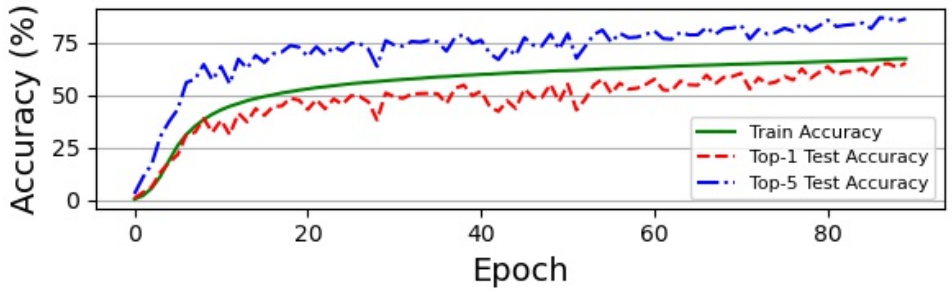


(a) CIFAR10

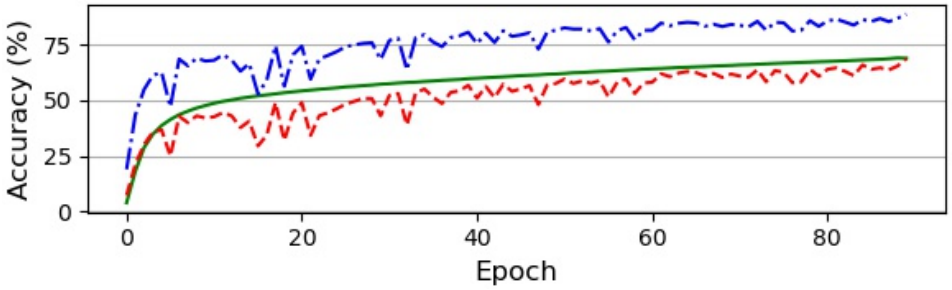


(b) CIFAR100

Figure 5: Training result on CIFAR



(a) Batch size = 1024



(b) Batch size = 256

Figure 6: Training result on ImageNet

E COMPARISON WITH REAL-VALUED COUNTERPARTS ON IMAGENET

Due to the limitation of resource and time, we just select each a model from conventional CNNs (i.e., ResNet18) and previous NAS methods (i.e., DOTS) to compare the accuracy drop from the real-valued counterparts on the ImageNet with our proposed `AutoShiftNet`. Table 7 shows the results. It can be found that the architecture searched by `AutoShiftNet` achieves the highest accuracy as a bit-shift network, and also has the lowest accuracy drop from the counterpart training in the real domain. Compared to other conventional CNNs and even most state-of-the-art NAS models, ResNet have more robust performance even training with bit-shift weights. However, it is still worse than our proposed `AutoShiftNet`, and more importantly, ResNets are much more heavy than NAS searched models.

Architecture	**Domain**	**Acc. (%) on ImageNet**			
		Top-1	**Diff.**	**Top-5**	**Diff.**
ResNet18	R	68.14	-	88.67	-
	BS	62.25	-5.89	83.79	-4.88
DOTS	R	72.75	-	90.96	-
	BS	66.36	-6.39	86.23	-4.73
`AutoShiftNet`	R	72.18	-	90.61	-
	BS	67.17	-5.01	87.38	-3.23

Table 7: Model accuracy on ImageNet of various architectures in the real (R) and bit-shift (BS) domains, and their differences (Diff.).