

Searching for Energy-Efficient Hybrid Adder-Convolution Neural Networks

Wenshuo Li¹, Xinghao Chen¹, Jinyu Bai^{1,3}, Xuefei Ning^{2,4}, Yunhe Wang¹

¹Huawei Noah's Ark Lab²Huawei TCS Lab.

³School of Integrated Circuit Science and Engineering, Beihang University.

⁴Department of Electronic Engineering, Tsinghua University.

fliwenshuo, xinghao.chen, yunhe.wang

g@huawei.com

Abstract

As convolutional neural networks (CNNs) are more and more widely used in computer vision area, the energy consumption of CNNs has become the focus of researchers. For edge devices, both the battery life and the inference latency are critical and directly affect user experience. Recently, great progress has been made in the design of neural architectures and new operators. The emergence of neural architecture search technology has improved the performance of network step by step, and liberated the productivity of engineers to a certain extent. New operators, such as AdderNets, make it possible to further improve the energy efficiency of neural networks. In this paper, we explore the fusion of new adder operators and common convolution operators into state-of-the-art light-weight networks, GhostNet, to search for models with better energy efficiency and performance. Our proposed search equilibrium strategy ensures that the adder and convolution operators can be treated fairly in the search, and the resulting model achieves the same accuracy of 73.9% with GhostNet on the ImageNet dataset at an extremely low power consumption of 0.612 mJ. When keeping the same energy consumption, the accuracy reaches 74.3% which is 0.4% higher than original GhostNet.

1. Introduction

With the development of convolutional neural networks (CNNs), the applications on the edge devices are becoming more and more popular. However, deployment of neural networks on edge devices is a highly challenging task due to their limited memory storage and low computational capability. For real-time responses and long battery life, the neural networks deployed on edge devices should have high accuracy but low power consuming and latency. To achieve this goal, great efforts have been dedicated for designing

Figure 1. Accuracy-Energy Pareto frontier. Our ACGhostNet models achieve the best performance in all energy ranges.

light-weight neural architectures and hardware-friendly operators.

Designing architectures for neural networks has been an important topic for deep learning. Recently, the emergence of Neural Architecture Search (NAS) [33] brings new paradigm for model design, which automatically discovers optimal networks given the search space without the burden of human experts. NAS has shown great performance for various tasks like image classification [30, 10], segmentation [12], object detection [5] etc. More importantly, NAS can search for hardware-friendly models [27] for different platforms based on the actual resource constraints. Recently, with the introduction of concepts such as shared weights [21], the efficiency of NAS algorithms has been improved significantly. At present, the NAS algorithm has been able to obtain considerable performance improvement at the cost of a small search time.

Meanwhile, the basic operation of convolutional neural network is evolving towards more efficient energy cost and less computational burden. For example, operations without multiplication have also received widespread attentions. AdderNet [2, 29] first proposes an addition op-

Corresponding author

erations which achieve significantly high energy efficiency while maintaining a satisfying accuracy. There are some variants of AdderNet, such as ShiftAddNet [31], to find different energy-accuracy trade-off between CNNs and AdderNets. Besides, Binary Neural Network (BNN) [7] is another kind of multiplication-free networks. Calculations in BNN could be implemented by XNOR operations. However, BNN now still suffer from severe accuracy loss in most cases. For example, state-of-the-art binary algorithm ReActNet [15] achieves 65.9% top-1 accuracy on ImageNet dataset with model ResNet-18, which is 4% lower than CNN and AdderNet [3]. Given these energy efficient networks such as AdderNets, it is therefore a quite interesting topic to explore whether we could discover a hybrid networks via convolutional and adder layers, to achieve better trade-off between accuracy and efficiency.

At the same time, the idea of manual network structure design for lightweight and mobile-based networks is also improving continuously, mainly due to their highly compact architectures and low computational cost. For example, MobileNets [11] utilizes the inverse bottleneck structure to achieve good accuracy and latency on mobile devices. GhostNet [9] series propose to use low-cost operations, such as depthwise convolution or shift [18], instead of expensive convolution operations, and they achieve state-of-the-art performance. Therefore, we aim to build upon these lightweight and mobile-based networks and investigate a proper way to utilize the adder layers to achieve better accuracy.

We aim to explore the design of ultra-low-power network architecture models. In the work of AdderNet, the author only gives the network under the convolution operation, and does not study the model under the mobile settings. In this paper, we propose a novel method to search for a hybrid of convolution and adder operators to obtain an extremely low-power architecture based on the lightweight GhostNet model. We first analyze the difficulties of applying adder operations to energy-efficient models like GhostNet. Then we propose several searching strategies to ensure we find satisfying hybrid adder-convolution neural networks, including search space and search object design, separate warmup strategy and adaptive learning rate trick. We also scale our searched models to different energy range to get models that adapt to different situations. Extensive experiments on CIFAR-100 and ImageNet demonstrate that our searched Adder-Convolution GhostNet (abbreviated as ACGhostNet) achieves better energy-accuracy tradeoff in almost every energy range. For example, ACGhostNet achieves the same accuracy of 73.9% with GhostNet on the ImageNet dataset at an extremely low power consumption of 0.612 mJ. When keeping the same energy consumption, the accuracy reaches 74.3% which is 0.4% higher than original GhostNet.

2. Related Work

2.1. Neural Architecture Search

NAS is often considered as a subfield of AutoML. In 2016, Zoph et al. [33] first propose to use reinforcement learning to search for optimal architectures. In their paper, they believe that the NAS framework consists of controller and evaluator. There is a pre-defined search space including optional operations and connections of neural networks. The controller samples architectures from the search space and then the evaluator gives a performance back, which is used to update the controller. The original NAS algorithm was very slow, taking thousands of hours to complete a search. To improve the efficiency of search process, researchers have made plenty of efforts.

The controller is responsible for selecting architectures in every search iteration, so the efficiency of the controller directly affects the number of iterations required for the search and the final results. NAS [33], MNasNet [24] and ENAS [21] use reinforcement learning with RNN model as the controller. With the development of weight-sharing NAS, differentiable controllers are more popular now, including DARTS [14] and FBNet [27]. At the meanwhile, evolutionary algorithm is also widely used. AmoebaNet [22] proposes to use aging evolution to encourage the exploration. NSGA-Net [17] gives a method to do multi-objective search. CARS [30] improves the NSGA-Net to mitigate the correlation problem brought by weight sharing.

For the evaluator, the most important improvement is the weight sharing strategy proposed by ENAS [21], which reduces the cost of evaluation significantly. Besides, zero-shot NAS is another research hotspot. Zero-shot NAS [28, 20] proposes to train a predictor to predict the performance of architectures instead of evaluating them.

In terms of search space, Zoph et al. propose to apply transferable cell search space to reduce the cost [34]. PNAS [13] gives a progressive way to accelerate the search process. DARTS [14] gives a multi-step two-edge cell search space and it is widely used now. MNasNet [24] introduces a single-path search space which is more friendly to mobile devices.

The effectiveness evaluation of NAS methods is one of the hotspots in recent years. Ning et al. [19] give some suggestions on how to achieve better results with one-shot NAS and zero-shot NAS. Yu et al. [32] also propose some methods to help to train the supernet. Researchers have developed some NAS benchmarks, such as NasBench-101, NasBench-201 and NasBench301, which provide the possibility to evaluate the strategies of NAS.

2.2. Operations and Architectures Design

The efficiency and accuracy of convolution neural networks are the focus of industry. Depthwise convolution [6]

has become a standard operation of light-weight neural net. To stabilize the training process, AdderNet proposes to use works on mobile devices. Based on the depthwise convolution operation, MobileNet series [11] become a benchmark for deploying neural networks on mobile devices. GhostNet [9] puts forward a new idea of generating ghost features by cheap operations, such as depthwise convolution and shift operation [18], which largely reduce the calculation cost.

In addition to the traditional convolution operation, there are also some multiplication-free operations which deliver significant power consumption benefits. One of them is BNN [7], which uses XNOR operations instead of multiplications. However, the accuracy has always been a problem that plagues the applications of BNN. State-of-the-art BNN algorithm, like ReActNet [15] ResNet-18, only achieves 65.9% top-1 accuracy based on ImageNet.

Another work that has received much attention is AdderNet. AdderNet [2] introduces to use distance instead of convolutions, which converts a multiplication operation to an addition operation. This replacement greatly reduces energy costs. With the help of following work [29, 3], AdderNet ResNet-18 can achieve 69.8% top-1 accuracy on ImageNet, which is comparable with CNNs. There are also weight researches on the hardware design of AdderNet [26, 25], making it possible to real-world applications. AdderNet has been applied in various computer vision tasks, like super-resolution [23] and detection [4].

3. Proposed Methods

3.1. Preliminary

In AdderNet, authors use distance instead of convolutions to represent feature maps. The calculations of Adder Networks can be formulated as:

$$Y(m; n; t) = \sum_{i=0}^X \sum_{j=0}^X \sum_{k=0}^X jX(m+i; n+j; k) \cdot F(i; j; k; t);$$

Where X denotes the input feature map and n denotes the filters. According to the investigation in [8], 32-bit floating point multiplication requires 3.7 pJ energy, while 32-bit floating point addition only requires 0.9 pJ. As for 8-bit x point number, multiplication requires 0.2 pJ energy, which is 6-7 times larger than addition (0.03 pJ). So using distance significantly reduces the power consuming.

As mentioned in [2], the gradient $\frac{\partial Y}{\partial F}$ and F is approximate by $\frac{\partial Y}{\partial X}$ distance and HardTanh, respectively.

$$\frac{\partial Y(m; n; t)}{\partial F(i; j; k; t)} = X(m+i; n+j; k) \cdot F(i; j; k; t)$$

$$\frac{\partial Y(m; n; t)}{\partial X(m+i; n+j; k)} = \text{HT}(F(i; j; k; t) \cdot X(m+i; n+j; k))$$

$$\eta = \frac{\eta_0}{k \cdot L(F_1)k_2};$$

in which η is a hyper-parameter which is actually the learning rate of adder layers, and k represents the number of weights in this layer. With adaptive learning rate, the norms of gradients of each layer are almost the same, so the converge process could be stabilized.

3.2. Search Space

Table 1. Comparison of CNN and AdderNet on GhostNet models

	Accuracy		gap
	CNN	AdderNet	
GhostNet-1.0x	75.56%	73.59%	1.97%
GhostNet-0.5x	72.92%	68.30%	4.62%
GhostNet-0.25x	69.53%	56.80%	12.73%

While AdderNet achieves great success with the large models like ResNet and VGG, the performances on light-weight models including GhostNet and MobileNet series of Adder GhostNet on CIFAR-100 dataset is significantly worse than original GhostNet. And the accuracy gap is becoming larger when the size of models are shrunk. This phenomenon may indicate that the presentative ability of depthwise adder operation is not enough. We have to take this into account while designing our search space and search strategies.

Table 2. Configurations of candidate modules in our search space

Module Type	1st Block (normal conv)	2nd Block (cheap op)	Groups
AAg1	adder	adder	N
AAg2	adder	adder	N/2
ACg1	adder	conv	N
ACg2	adder	conv	N/2
CAG1	conv	adder	N
CAG2	conv	adder	N/2
CCg1	conv	conv	N
CCg2	conv	conv	N/2

To alleviate the accuracy loss caused by depthwise adder operations, we provide group convolutions as a substituted choice. We choose state-of-the-art light-weight model GhostNet as our backbone. The GhostNet model contains [1; 2; 2; 4; 2; 5] blocks for six stages. In each block, there are two GhostModules and one SE Module. For the first block in stage 1-2-3-5, there is a stride=2 layer to downsample the feature maps.

Based on the GhostNet, we apply a single-path block-wise search space. The unit of our search space is shown

Figure 2. Structure of our backbone model and search space.

in the right of Figure 2, which we called ACGhostModule. Operation conv3x3 and adder3x3 actually refer to a conv3x3/adder3x3-BatchNorm-ReLU sequence. Operation dwcv3x3g1 and dwcv3x3g2 refers to the cheap operations in Ghostnet. To specify, they are conv3x3 with groups equal to channels and conv3x3 with groups equal to half of the channels, respectively. Same with dwadd3x3g1 and dwadd3x3g2. So there are $2 \times 4 = 8$ choices in each block, and the size of search space is $2 \times 8 \times 10^{14}$. For ease of description, we use ACg1 to present the block which has adder3x3(A) and dwcv3x3g1(Cg1) operations and the others are described with the same method, shown in Table 2.

operations in searching are just the same as that in training. However, for adder operations, the rankings are obviously different. Adder5x5 operation performs worse than adder3x3 operation in some cases. If we apply kendall-tau (the larger, the better) as a quantifiable indicator to evaluate the rank correlation between the super-net and single models, we can find that the kendall-tau of adder operation is only 0.33 while the kendall-tau of convolution operation is 1. The result means that AdderNets are more difficult to converge than CNNs and are maybe influenced by the small model trap problem [30] more seriously, which claims that smaller models have more opportunities to win due to their fast convergence.

3.3. Search Object

The original search object used in CARS is the number of parameters. In this paper, we need a different search object which could tell the difference between adder operations and convolution operations with the same number of parameters. Therefore, we choose energy as our search object. We use the data with 32-bit floating point number in [8]. So we model the addition operation as 0.9 pJ energy and the multiplication operation as 3.7 pJ energy. Then the energy cost of a multiply accumulate (MAC) operation is 4.6 pJ and an adder accumulate operation is 1.8 pJ.

3.4. Search Strategies

Different from vanilla convolution neural networks, it takes more training time for AdderNets to converge. This proposed a powerful evolutionary algorithm called pNSGA-causes to some kind of unfairness when we search for all. pNSGA-III can help to overcome the small model trap hybrid architecture containing both convolution operations in some way, so it is a perfect choice for our purpose. and adder operations. We conduct a toy experiment to show the unfairness. The toy experiments contains two operation sets: (1) conv3x3 (c3) and conv5x5 (c5) / (2) adder3x3 (a3) and adder5x5 (a5). We only set two layers in our toy didates are randomly chosen as the initial population and single path super-net and we train these two toy super-nets on candidates for some epochs. After every for three times. Also we train each standalone model for t epochs of training, we insert a evolution process to update the population. For the evolution process, we apply in Table 3. We can find that the rankings of convolution

Table 3. Toy experiment of ranking correlation

	search		train	
	accuracy	rank	accuracy	rank
a3,a3	48.26 0.15	3	72.60 0.08	4
a5,a3	47.61 0.17	4	75.17 0.05	3
a3,a5	49.41 0.22	1	77.15 0.15	2
a5,a5	48.72 0.17	2	78.45 0.07	1
c3,c3	61.58 0.34	4	75.33 0.33	4
c5,c3	62.50 0.39	3	77.72 0.19	3
c3,c5	64.82 0.30	2	79.31 0.20	2
c5,c5	65.45 0.39	1	81.07 0.21	1

To alleviate the ranking correlation issues we mentioned above, we must find suitable search strategies. CARS [30] proposed a powerful evolutionary algorithm called pNSGA-III can help to overcome the small model trap hybrid architecture containing both convolution operations in some way, so it is a perfect choice for our purpose. There is a brief introduction of the search process. First we train the supernet by randomly picking paths, and this operation would give each path a initial performance. Then candidates are randomly chosen as the initial population and we train these two toy super-nets on candidates for some epochs. After every for three times. Also we train each standalone model for t epochs of training, we insert a evolution process to update the population. For the evolution process, we apply pNSGA-III algorithm to generate the next generation. We

Table 4. Results of CIFAR-10 dataset

Model	#Params	#FAdd	#FMul	energy(J)	Accuracy (%)
GhostNet-1.0x [9]	3.9M	43M	43M	196.3	94.9
Adder-GhostNet-1.0x	3.9M	81M	4M	88.4	93.3
ACGhostNet-A (Ours)	4.0M	71M	19M	135.3	94.8
ACGhostNet-B (Ours)	4.0M	61M	28M	156.8	95.2
ACGhostNet-C (Ours)	4.0M	61M	30M	165.3	95.1
ACGhostNet-D (Ours)	4.0M	56M	36M	185.2	95.1

use mutation and crossover to expand the population, and divide the whole supernet into three sub-supernets according to the ratio of adder operations, and then train each sub-supernet until it converges to a setting accuracy threshold [accuracy, 1/energy], respectively. The former one is the normal pareto frontier and latter one is used to alleviate the influence of small model trap. Then the results of two sortations are directly merged and the best candidates are reserved. We repeat these steps until reaching the maximum training epochs.

In our search space, one path refers to one candidate, and the mutation means one choice of ACGhostModule turns into another while crossover means one path take the choice of a block from the other path.

We have made some modifications to the search and training process to make it more suitable for the hybrid search of convolution operations and adder operations.

3.4.1 Training Settings of the Super-net

As we introduced in section 3.1, AdderNet uses adaptive learning rate to update the parameter. The reason why AdderNet needs adaptive learning rate is that the norm of gradients of AdderNets vary widely across different layers. Adaptive learning rate can stabilize the training process of AdderNet. Considering about our hybrid adder-convolution search, while training the super-net, convolution operation and adder operation are updated together. If we only apply adaptive learning rate to adder operations, the gradients would differ greatly between these two operations, which lead to the unfairness of converge speed. Therefore, we apply adaptive learning rate for both convolution operations and adder operations. Noted that convolution neural network with adaptive learning rate can also achieve similar accuracy compared with normal learning rate.

3.4.2 Seperate Warmup Strategy

In some exploratory experiments, we find that the warmup process can also bring unfairness between adder operations and convolution operations. With same epochs training, adder operations are significantly worse than convolution operations. So during the first one to two rounds of evolution, adder operation would soon become scarce. To solve this problem, we propose a separate warmup strategy.

4. Experiments

4.1. Experimental Settings

CIFAR-10 is a small dataset containing 60000 32 RGB images, in which 50000 images are used for training and 10000 images are used for evaluation. ImageNet is much larger, with 1281167 training and 50000 evaluation images. CIFAR-10 is generally used as the surrogate dataset in NAS process which means that we search the optimal architectures on CIFAR-10 dataset, and they are normally trained and used on ImageNet dataset.

Searching: We train the supernet for 500 epochs in total, in which the first 50 epochs are used to warmup. Then we repeat the evolving process every ten epochs. The size of population is set as 128, and we expand it to 256 for the evolving process. We use SGD optimizer with momentum to train supernet and the momentum is set as 0.9. The learning rate is initialized as 0.025 and then decays with a cosine scheduler. The batch size of training is 128 and the weight decay is set as $1e^{-4}$. Half of the training data is used to train the supernet and the other half is used to evaluate.

Training: We also use the SGD optimizer with momentum to train our final models. The initial learning rate is set as 0.4 and decays with a cosine scheduler too. We train the model with 8 Nvidia V100 GPUs using 1024 batch size. The weight decay is set as $1e^{-5}$ and dropout ratio is 0.2. We train the model for 800 epochs in total and 4 epochs are used to warmup. The data pre-processing is all the same as [9]. The hyper-parameter of adaptive learning rate is set as $\alpha=15$. We do not apply grad clip technique since grad clip may destroy the gradient after scaling by adaptive learning rate.

4.2. Results on CIFAR10

We train the searched models on CIFAR-10 to make a quick evaluation. The models are trained 600 epochs with 256 batch size on a single Nvidia V100 GPU, and the learn-

Table 5. Results of ImageNet dataset

Model	#Params	#FAdd	#FMul	resolution	energy(mJ)	Top-1 (%)	Top-5 (%)
CNNs							
MobileNetV3-Large0.75 [10]	4.0M	155M	155M	224 224	0.713	73.3	-
MobileNetV3-Large1.0 [10]	5.4M	219M	219M	224 224	1.007	75.2	-
SkipblockNet-XS [1]	2.3M	81M	81M	224 224	0.372	66.9	88.9
SkipblockNet-S [1]	3.6M	152M	152M	224 224	0.699	73.8	91.4
SkipblockNet-M [1]	5.5M	246M	246M	224 224	1.131	76.2	92.8
MUXNet-xs [16]	1.8M	66M	66M	224 224	0.304	66.7	86.8
MUXNet-s [16]	2.4M	117M	117M	224 224	0.538	71.6	90.3
MUXNet-m [16]	3.4M	218M	218M	224 224	1.003	75.3	92.5
GhostNet-0.5x [9]	2.6M	42M	42M	224 224	0.225	66.2	86.6
GhostNet-1.0x [9]	5.2M	141M	141M	224 224	0.649	73.9	91.4
GhostNet-1.3x [9]	7.3M	226M	226M	224 224	1.058	75.7	92.7
AdderNets							
Adder-GhostNet-1.0x	5.2M	267M	15M	224 224	0.297	68.9	88.1
Adder-GhostNet-1.55x	9.5M	606M	27M	224 224	0.644	72.8	90.9
Hybrid Add-Convolution Networks							
ACGhostNet-A (Ours)	5.2M	235M	64M	224 224	0.448	72.6	90.6
ACGhostNet-B (Ours)	5.2M	202M	96M	224 224	0.537	73.4	91.0
ACGhostNet-C (Ours)	5.2M	204M	101M	224 224	0.556	73.6	91.4
ACGhostNet-D (Ours)	5.2M	192M	119M	224 224	0.612	73.9	91.5
ACGhostNet-A-S (Ours)	3.8M	157M	44M	224 224	0.302	69.9	88.9
ACGhostNet-C-S (Ours)	5.2M	243M	120M	240 240	0.661	74.3	91.8
ACGhostNet-D-S1 (Ours)	5.9M	225M	140M	224 224	0.721	74.6	91.9
ACGhostNet-D-S2 (Ours)	5.9M	292M	182M	256 256	0.936	75.6	92.5

ing rate is initialized as 0.025 and then decay with a cosine scheduler. The results are shown in Table 4. We can find that our ACGhostNet-A model achieves comparable accuracy with original GhostNet-1.0x while the energy cost decreases by 30%. And ACGhostNet-B/C/D achieve even better accuracy with less energy cost.

4.3. Results on ImageNet

We finally evaluate our models on ImageNet dataset. The results are shown in Table 5. In addition to the four models we searched, we also evaluate the scaled models, to check the energy-accuracy tradeoff in different energy range. We scale the searched models by enlarging or shrinking the number of layers or input resolution.

We pick several state-of-the-art energy-efficient CNN models, like MobileNetV3 [10], MUXNet [16] and SkipblockNet [1], to compare with. Our ACGhostNet obviously achieves better accuracy under similar or even less energy cost. To specify, our ACGhostNet-D achieve the same accuracy 73.9% with GhostNet-1.0x while the energy cost is less, and our ACGhostNet-C-S achieve 0.4% better accuracy with the same energy cost. Considering about large model like GhostNet-1.3x and tiny model like GhostNet-0.5x, our ACGhostNet-D-S2 and ACGhostNet-

4.4. Ablation Study

Table 6. Kendall-tau of different training strategies

Training Strategy		Kendall-Tau
Adaptive LR	Seperate Warmup	
	p	-0.359
p		-0.153
p	p	0.296
		0.442

In this section, we evaluate the effectiveness of our seperate warmup strategy and training hyper-parameters. We evaluate them from two aspects: the final accuracy of searched standalone models and the ranking correlations on the subset of our search space.

First we show the search results under different warmup settings on CIFAR-10 dataset in the left of Figure 3. From the figure we can find that seperate warmup until accuracy reaches 70% achieves the best results. We also evaluate the search results of convolution operations with/without adaptive learning rate, shown in the right of Figure 3. We can find that convolution operations with adaptive learning rate is also better than without adaptive learning rate.

Figure 3. Energy-Accuracy curve under different settings. Left: different warmup settings. Right: different learning rate settings.

Figure 4. Scatter of four types of training strategies. The X axis represents the ground truth accuracy and the Y axis represents the predict accuracy.

From another aspect, we conduct a toy experiment on the subset of our search space, to show the influence of our strategies to the ranking correlations. We pick two operations in our search space as the subset, AAg1 and CCg1. And we set the block number of each stage as 1, so there are 6 blocks in total. The whole size of our toy search space is $2^6 = 64$. We train all standalone models in our toy search space to get the ground truth ranking. Then we train the super-net with different strategies and then evaluate every architecture to get a searched ranking. For no separate warmup scenario, we train the super-net for 50 epochs. Otherwise, we separately warmup different operations to certain accuracy threshold and then train the super-net for 20 epochs, to keep the fairness of comparison.

The scatters of predict-ground truth accuracy are shown in Figure 4. The scatters of upper two figures show a negative correlation between predict accuracy and ground truth accuracy, while the lower two show a positive correlation. We use kendall-tau to directly show the ranking correlation in Table 6. Training with both adaptive learning rate (ALR) and separate warmup strategy obviously achieve the best kendall-tau and significantly outperforms others. Noted that without adaptive learning rate, the kendall-tau even becomes negative, which means the predict results are opposite to the ground truth. This means that adding operations which should be worse than convolution operations are wrongly thought to be better with the help of adaptive learning rate.

Figure 5. Architecture of our searched models

Figure 6. Energy and FLOPs distribution in our searched models. The X axis represents the index of blocks and the Y axis represents the average energy or FLOPs.

4.5. Analysis of Searched Models

Our searched models are shown in Figure 5. There are something in common among our searched models. First, our sampler think the first block of each stage is more important than others. We can find that the first block of stage 2 and stage 5 are all formed by convolution operations, and most have groups=2. Second, for the models with less energy cost (like ACGhostNet-A and ACGhostNet-B), there are more AA and AC blocks, which have lower energy consuming. For larger models, there are more CC and CA blocks which requires more energy but have better accuracy. Third, there are more groups=2 blocks in the last two stages, which may means that these blocks need more model capacity to extract features. We also calculate the average number of FLOPs and energy cost of each block among 4 architectures, shown in Figure 6. We can find that block 1 and block 10 require the largest FLOPs since there are many groups=2 layers, but block 11 requires the most energy cost since all operations are convolution. We hope these knowledges would help in the future hybrid adder-convolution neural network design.

5. Conclusion

In this paper, we make a exploration of the searching for energy-efficient hybrid adder-convolution neural networks. We propose the hybrid search space which contains both adder operations and convolution operations. Besides, we analyse the difficulties to search with adder operations, and then we give corresponding guidance to deal with the unfairness in our search process. We apply CARS to alleviate the small model trap problem, and raise separate warmup strategy and some training recommends. With these techniques, our searched ACGhostNet models achieve comparable accuracy with original GhostNet with only 612 mJ energy, and the accuracy under the same energy cost as original GhostNet is improved by 0.4% on ImageNet dataset. We also make some ablation study to show the unfairness during searching and the effectiveness of our strategies. Finally we show our searched architectures and find some common knowledge for the future model design.

In the future, we would try larger hybrid search space, with more possible operations, to search for more energy-efficient models. Besides, we would attempt to search for different vision tasks and hardware platforms.

References

- [1] Lusine Abrahamyan, Valentin Ziatichin, Yiming Chen, and Nikos Deligiannis. Bias loss for mobile neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* pages 6556–6566, 2021. 6
- [2] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 1468–1477, 2020. 1, 3
- [3] Hanting Chen, Yunhe Wang, Chang Xu, Chao Xu, Chunjing Xu, and Tong Zhang. Universal adder neural network. *arXiv preprint arXiv:2105.14202* 2021. 2, 3
- [4] Xinghao Chen, Chang Xu, Mingjing Dong, Chunjing Xu, and Yunhe Wang. An empirical study of adder neural networks for object detection. In *NeurIPS* 2021. 3
- [5] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979* 1(2):4–1, 2019. 1
- [6] François Chollet. Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* pages 1251–1258, 2017. 2
- [7] 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 preprint arXiv:1602.02830* 2016. 2, 3
- [8] William Dally. High-performance hardware for machine learning. *NIPS Tutorial* 2, 2015. 3, 4
- [9] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 1580–1589, 2020. 2, 3, 5, 6
- [10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* pages 1314–1324, 2019. 1, 6
- [11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* 2017. 2, 3
- [12] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 82–92, 2019. 1
- [13] 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)* pages 19–34, 2018. 2
- [14] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* 2018. 2
- [15] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. *European Conference on Computer Vision* pages 143–159. Springer, 2020. 2, 3
- [16] Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Muxconv: Information multiplexing in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 12044–12053, 2020. 6
- [17] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference* pages 419–427, 2019. 2
- [18] Ying Nie, Kai Han, Zhenhua Liu, An Xiao, Yiping Deng, Chunjing Xu, and Yunhe Wang. Ghostsr: Learning ghost features for efficient image super-resolution. *arXiv preprint arXiv:2101.08525* 2021. 2, 3
- [19] Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. *arXiv preprint arXiv:2008.03064* 2021. 2
- [20] Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based nas. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII*, pages 189–204. Springer, 2020. 2
- [21] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* pages 4095–4104. PMLR, 2018. 1, 2
- [22] 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. 2
- [23] Dehua Song, Yunhe Wang, Hanting Chen, Chang Xu, Chunjing Xu, and Da-Cheng Tao. Addersr: Towards energy efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 15648–15657, 2021. 3
- [24] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 2820–2828, 2019. 2
- [25] Soyeon Um, Sangyeob Kim, Sangjin Kim, and Hoi-Jun Yoo. A 43.1 tops/w energy-efficient absolute-difference-accumulation operation computing-in-memory with computation reuse. *IEEE Transactions on Circuits and Systems II: Express Briefs* 68(5):1605–1609, 2021. 3

- [26] Yunhe Wang, Mingqiang Huang, Kai Han, Hanting Chen, Wei Zhang, Chunjing Xu, and Dacheng Tao. Addernet and its minimalist hardware design for energy-efficient artificial intelligence. arXiv preprint arXiv:2101.10015, 2021. 3
- [27] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10734–10742, 2019. 1, 2
- [28] Yixing Xu, Yunhe Wang, Kai Han, Yehui Tang, Shangling Jui, Chunjing Xu, and Chang Xu. Renas: Relativistic evaluation of neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4411–4420, 2021. 2
- [29] Yixing Xu, Chang Xu, Xinghao Chen, Wei Zhang, Chunjing Xu, and Yunhe Wang. Kernel based progressive distillation for adder neural network. arXiv preprint arXiv:2009.13044, 2020. 1, 3
- [30] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1829–1838, 2020. 1, 2, 4
- [31] 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. 2
- [32] Kaicheng Yu, Rene Ranftl, and Mathieu Salzmann. How to train your super-net: An analysis of training heuristics in weight-sharing nas. arXiv preprint arXiv:2003.04276, 2020. 2
- [33] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016. 1, 2
- [34] 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, pages 8697–8710, 2018. 2