Efficient Neural Architecture Search: A Broad Version

Zixiang Ding 1,2, Yaran Chen 1,2, Nannan Li 1,2, Dongbin Zhao 1,2*, and C.L. Philip Chen 3,4

1 State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
2 University of Chinese Academy of Sciences, Beijing 100049, China
3 School of Computer Science & Engineering, South China University of Technology, Guangzhou, Guangdong 510006, China
4 College of Navigation, Dalian Maritime University, Dalian 116026, China
{dingzixiang2018, chenyaran2013, linannan2017, dongbin.zhao}@ia.ac.cn, philip.chen@ieee.org

Abstract

Efficient Neural Architecture Search (ENAS) achieves novel efficiency for learning architecture with high-performance via parameter sharing, but suffers from an issue of slow propagation speed of search model with deep topology. In this paper, we propose a Broad version for ENAS (BENAS) to solve the above issue, by learning broad architecture whose propagation speed is fast with reinforcement learning and parameter sharing used in ENAS, thereby achieving a higher search efficiency. In particular, we elaborately design Broad Convolutional Neural Network (BCNN), the search paradigm of BENAS with fast propagation speed, which can obtain a satisfactory performance with broad topology, i.e. fast forward and backward propagation speed. The proposed BCNN extracts multi-scale features and enhancement representations, and feeds them into global average pooling layer to yield more reasonable and comprehensive representations so that the achieved performance of BCNN with shallow topology can be promised. In order to verify the effectiveness of BENAS, several experiments are performed and experimental results show that 1) BENAS delivers 0.23 day which is 2x less expensive than ENAS, 2) the architecture learned by BENAS based small-size BCNNs with 0.5 and 1.1 millions parameters obtain state-of-the-art performance, 3.63% and 3.40% test error on CIFAR-10, respectively, which are 25.3% top-1 error on ImageNet just using 3.9 millions parameters.

1 Introduction

Recently, Neural Architecture Search (NAS) [25] which automates the process of model designing is gaining around in past several years. Computer vision tasks (e.g. image classification [14, 19, 25]) can all be solved by NAS with surprising performance. However, early approaches [20, 25, 26] suffer from the issue of inefficiency. To solve this issue, some one-shot approaches [1, 6, 11, 14, 19] are proposed. Generally speaking, one-shot NAS approaches sample cells, a micro search space presented in [26], from a family of predefined candidate operations depending on a policy, and treat the sampled cells as building block of deep architecture, i.e. child model, whose performance is used for policy’s parameters update. These one-shot approaches avoid retraining each candidate deep architecture from scratch so that high efficiency can be promised.

In particular, Efficient Neural Architecture Search (ENAS) [19] delivers state-of-the-art efficiency, 0.45 GPU day, using parameter sharing and reinforcement learning. However, for promising the performance of learned architecture, ENAS has to use a deep child model whose propagation speed is slow as search paradigm. Therefore, ENAS suffers greatly from the issue of slow propagation speed of search model (child model) with deep topology.

In this paper, we propose a Broad version for ENAS (BENAS), an automatic architecture search approach with state-of-the-art efficiency. Different from other NAS approaches, in BENAS, an elaborately designed Broad Convolutional Neural Network (BCNN) instead of a deep one is discovered in a one-shot model by parameter sharing and reinforcement learning for solving the aforementioned limitation of ENAS. Particularly, we propose a new paradigm of search model, BCNN, which can obtain satisfactory performance with shallow topology, i.e. fast forward and backward propagation speed. The proposed BCNN extracts multi-scale features and enhancement representations, and feeds them into global average pooling layer to yield more reasonable and comprehensive representations so that the achieved performance of BCNN can be promised. Our contributions can be summarized as follows:

- We propose a broad version of ENAS named BENAS to further improve the efficiency of ENAS by replacing the search model with BCNN which is elaborately designed for promising satisfactory performance and fast propagation speed simultaneously.
- We achieve 2x less search cost (with a single GeForce
2.2 Neural Architecture Search

As a powerful tool for solving the architecture engineering issue with respect to some artificial intelligence related tasks, especially computer vision tasks, NAS achieves amazing performance in past several years. The unprecedented success of NAS is depending on the unacceptable computational resources.

There exist recent efforts introducing various methods to improve the search efficiency of NAS \cite{chen2018}. For example, based on a Sequential Model-Based Optimization (SMBO) strategy, Progressive Neural Architecture Search (PNAS) \cite{liu2018} searches the structure of convolutional neural networks in order of increasing complexity. A multi-objective evolutionary algorithm is proposed for improving the efficiency of architecture search in LEMONADE \cite{liu2019}. However, these approaches are still not efficient enough due to need to retrain each child model from scratch.

A great number of one-shot approaches \cite{han2016, liu2019, zoph2018} which define all possible candidate architectures in one-shot model for avoiding the issue of each child model retraining from scratch have been presented for improving the efficiency of NAS further. SMASH \cite{han2016} uses a hypernetwork to generate the weights of a designed architecture so that the search process can be accelerated greatly. Furthermore, Liu et al. \cite{liu2019} propose Differentiable ARchiTecture Search (DARTS) which discovers the computation cells within a continuous domain for formulating NAS in a differentiable way. DARTS achieves three orders of magnitude less expensive than previous approaches \cite{wu2019, zoph2017}. In particular, a NAS approach with novel efficiency (uses a single GeForce GTX 1080Ti GPU for 0.45 day which is 3x faster than DARTS) named ENAS \cite{zoph2018} is presented. In order to improve the efficiency, ENAS uses parameter sharing for avoiding each candidate deep architecture retraining from scratch.

3 The Proposed Approach

3.1 Efficient Neural Architecture Search

In the reinforcement learning based ENAS, an Long Short-Term Memory (LSTM) \cite{hochreiter1997} controller with parameter $\theta$ is trained in a loop: the LSTM first generates two types of cells, Normal cell and Reduction cell (more details can be found in previous works \cite{zoph2018, zoph2016}), with a list of tokens $a_{1:T}$ according a sampling policy $\pi(\cdot)$ for stacking up into a relative deep child model $m$, and then the child model whose weights $w$ are inherited from the one-shot model is trained in a single step for measuring its validation accuracy $R(m; w)$ on the desired task. Subsequently, the $R(m; w)$ is treated as the reward of reinforcement learning to guide the LSTM controller for discovering various cells with better performance. Moreover, ENAS asks the LSTM controller to maximize the expected reward $J(\theta)$, where

$$J(\theta) = \mathbb{E}_{\pi(a_{1:T}; \theta)}[R(m; w)].$$

(1)

Moreover, a gradient policy algorithm, REINFORCE \cite{williams1992} is applied to compute the policy gradient $\nabla_\theta J(\theta)$, where
ENAS can improve the efficiency but lead to performance loss. In order to solve the above issue, we propose a broad version of ENAS named BENAS where a novel paradigm of child model named BCNN is elaborately designed as the search model of ENAS.

### 3.3 Broad Convolutional Neural Network

In BENAS, we propose BCNN who can deliver satisfactory performance and fast propagation speed simultaneously with broad topology as the search paradigm and also child model for automatic architecture designing. Moreover, a two-layers LSTM controller, reinforcement learning and parameter sharing (more details can be found in ENAS [19]) are also adopted for architecture sampling, controller’s parameter updating and accelerating architecture search process, respectively. As aforementioned, the proposed BCNN is a developed CCFBLS [3] which is not only broad but also deep. For intuitional understanding, the structure of BCNN and its two important components, convolution and enhancement blocks are depicted in Figure 1.

BCNN consists of \( u \) convolution blocks denoted as \( \text{Conv}_i \) \((i = 1, 2, \ldots, u)\) and \( v \) enhancement blocks denoted as \( \text{En}_j \) \((j = 1, 2, \ldots, v)\) which are used for feature extraction and enhancement, respectively. In the convolution block, there are \( k + 1 \) convolution cells: \( k \) deep cells and a single broad cell which are utilized to deep and broad features extraction, respectively. Moreover, \( u \) is determined by the size of input images. For example, we set \( u = 2 \) for the experiments on CIFAR-10 with \( 32 \times 32 \) pixels. The other two parameters \( k \) and \( v \) need to be defined by user. For convenient expression below, a simple notation, \( u  \@  k  \@  v \) is defined to indicate these three parameters in the BCNN. For instance, \( 3@2@2 \) means that there are 3 convolution blocks where each one contains 2 deep cells, and 2 enhancement blocks.

\[
\nabla_\theta J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi(a_t; \theta)}[\nabla_\theta \log \pi(a_t|a_{t-1}; \theta) R(m; w)].
\]

(2)

After many iterations of this loop are repeated, novel cells with satisfactory performance can be found.

#### 3.2 Problem Analysis

ENAS suffers from an issue of slow propagation speed of child model with deep topology. Below, we will discuss the reasons of that in details.

First of all, a priori knowledge should be given. As we all know, two deep neural networks with same parameters but different depths have various propagation speeds where the shallow one is faster than the deep one. Moreover, the performance of neural network with deep topology is better than the shallow one.

Furthermore, ENAS has to employ a child model with deep topology as search paradigm. Loosely speaking, there are two phases, architecture search and architecture deriving in ENAS. On one hand, in the state of architecture search, the cells sampled by LSTM are stacked up as building blocks of child model with \( l_0 \) layers. On the other hand, a deeper model with \( l_1(l_1 > l_0) \) layers is stacked up by the sampled cells in the architecture deriving phase. Without loss of generality, the number of layers \( l_1 \) is set as large as possible to achieve a high accuracy. In the meanwhile, the number of search model’s layers \( l_0 \) should be set to a relative large value for reducing the differences between the models constructed in the above two phases, i.e. promising the stability and rationality of ENAS.

From the above, we can draw a conclusion that depth reduction of child model in the architecture search state of

![Figure 1: Broad convolutional neural network. Top: the topology of the proposed BCNN. Bottom Left: the structure of convolution block. Bottom Right: the structure of enhancement block.](image-url)
in BCNN.

In each convolution block, the deep cells and broad cell have same topologies but various strides: one for the deep and two for the broad. In order to extract broad features from the output features of final deep cell, the broad cell returns the feature maps with half width, half height and double channels, i.e. broad features. In each enhancement block, there is a single enhancement cell with one stride and different topology from those convolution cells. The proposed BCNN cascades $u$ convolution blocks one after another, and feeds the output of final convolution block into each enhancement block as the input for obtaining enhancement feature representations. The convolution and enhancement features from every convolution and enhancement block are all connected with the global average pooling layer to yield more reasonable and comprehensive representations for achieving promised performance of the proposed BCNN. For clear understanding, the formulaic expressions of BCNN are given below.

For convolution block $Conv_i$, its deep feature mapping $Z^{(i)}_h(h = 1, 2, \ldots, k)$ and broad feature mapping $Z^{(i)}_{k+1}$ can be defined as

$$Z^{(i)}_h = \phi(Z^{(i)}_{h-2}, Z^{(i)}_{h-1}, \{W^{(i)}_{h,\text{deep}}, \beta^{(i)}_{h,\text{deep}}\}), i = 1, 2, \ldots, u,$$  

where $\{W^{(i)}_{h,\text{deep}}, \beta^{(i)}_{h,\text{deep}}\}$ and $\{W^{(i)}_{k+1,\text{broad}}, \beta^{(i)}_{k+1,\text{broad}}\}$ are the weight, bias matrices of deep cells and broad cell in convolution block $i$, respectively. Moreover, $\phi(\cdot)$ is a set of transformations (e.g. depthwise-separable convolution [9], pooling, skip connection) by the deep cells and broad cell. In other words, each cell in the convolution block uses the outputs of its previous two cells as the inputs for combining various features. However, there is a doubt in [4] that $Z^{(i-1)}_n$ and $Z^{(i)}_0$ are not defined. A complementary expression is given as

$$\{Z^{(i)}_{-1}, Z^{(i)}_0\} = \{Z^{(i-1)}_k, Z^{(i-1)}_{k+1}\}, i = 2, 3, \ldots, u.$$  

Moreover, as aforementioned, a convolution with $3 \times 3$ kernel size is inserted in the front of BCNN to provide the input information for the first and second convolution cell. As a result, the output of the $3 \times 3$ convolution can be represented as $Z^{(i)}_{\xi}$, where $\xi \leq 0$.

For enhancement block $En_j$, its enhancement feature representations $H^{(j)}$ can be mathematically expressed as

$$H^{(j)} = \varphi(Z^{(i)}_k, Z^{(i)}_{k+1}; \{W^{(j)}, \beta^{(j)}\}), j = 1, 2, \ldots, v$$

where $W^{(j)}$ and $\beta^{(j)}$ are the weight and bias matrices of enhancement cell in enhancement block $j$, respectively. Moreover, $\varphi(\cdot)$ is a set of transformations by the enhancement cell.

In order to ensure as much as convolution and enhancement features can be aggregated as more reasonable and comprehensive representations for achieving promised performance of BCNN, all outputs of each convolution and enhancement block are connected directly with the global average pooling layer.

**Algorithm 1 Training Strategy for BENAS**

**Input:** Search space $S$, an Long-Short-Time-Memory LSTM with parameters $\theta$, the number of nodes $N$, selective probability distribution $\pi(\cdot)$, optimizer SGD, standard cross-entropy loss function $\mathcal{L}$, validation data $val$, empty set $\Omega$ and the number of candidate deriving architectures $\eta$

**Output:** $\Omega$ with $\eta$ candidate deriving architectures

1: Build one-shot model $m$ following the paradigm of BCNN based on $S$
2: for each training epoch do // train one-shot model
3: for each training step do
4: $cell \leftarrow LSTM(S, N; \theta)$ // generate cell
5: $m(w) \leftarrow \text{Stack}(cell, l)$ // construct child model
6: for each weight $\omega$ in $w$ do //restore weights
7: if $\omega$ in $m$ then
8: Restore $\omega$ for $m$
9: else
10: Initialize $\omega$ for $m$
11: end if
12: end for
13: $loss \leftarrow \mathbb{E}_{w \sim \pi(m; \theta)}[\mathcal{L}(m; \theta)]$ // compute loss
14: Minimize loss using SGD to update $w$
15: $m \leftarrow \text{Store}(w)$ // store $w$ in $m$
16: end for
17: while should training controller do // train controller
18: for each controller training step do
19: $cell \leftarrow LSTM(S, N; \theta)$
20: $m(w) \leftarrow \text{Stack}(cell, l)$
21: for each weight $\omega$ in $w$ do
22: if $\omega$ in $m$ then
23: Restore $\omega$ for $m$
24: else
25: Initialize $\omega$ for $m$
26: end if
27: end for
28: $R(m; w) \leftarrow m(val)$ // achieve accuracy
29: Treat $cell$ as a list of actions $a_{1:N}$
30: Treat $R(m; w)$ as reward
31: $J(\theta) \leftarrow \mathbb{E}_{\pi(a_{1:N}; \theta)}[R(m; w)]$
32: Compute $\nabla_{\theta} J(\theta)$ using REINFORCE
33: Update $\theta$
34: end for
35: end while
36: end for
37: for $i = 0 \rightarrow \eta$ do
38: $cell \leftarrow LSTM(S, N; \theta)$
39: $\Omega \leftarrow cell$ // add cell to $\Omega$
40: end for
41: return $\Omega$
applied a series of standard data augment techniques which can

Similarly, CIFAR-10 is chosen as the search dataset and ap-

4.1 Architecture Search on CIFAR-10

4 Experiments and Analysis

An overview of training strategy of BENAS can be found in
Algorithm [1]. It is obvious that we are not only need to

First of all, the parameters of LSTM $\theta$ are fixed in the first

be found in ENAS [19] for details. In BENAS, we chose five
candidate operations: $3 \times 3$ depthwise-separable convolution,
$5 \times 5$ depthwise-separable convolution, $3 \times 3$ max pooling,
$3 \times 3$ average pooling and skip connection as the components
of convolution cell and enhancement cell with 7 nodes.

In the architecture search phase, for training the broad
model with topology of $2@0@2$ (the definition of this no-
tation refers to Section [3.3]), the Nesterov momentum is
adopted and the learning rate follows the cosine schedule with
$l_{\text{max}}=0.05$, $l_{\text{min}}=0.0005$, $T_0=10$ and $T_{\text{mul}}=2$ [16]. Further-
more, the experiment runs for 150 epochs with batch size 128.
For updating the parameters $\theta$ of LSTM, the Adam optimizer
with a learning rate of 0.0035 is applied.

Figure 2: The optimal architecture discovered by BENAS: (a) The
convolution cell. (b) The enhancement cell.

and output a group of feature maps with different importance.
Moreover, the importance is represented by the number of
output channels which the larger is the more important it is.
Furthermore, these $1 \times 1$ convolutions have different strides
for concatenating all input feature maps with same size.

Just because of the above, the proposed BCNN can achieve
high performance with a shallow topology so that the ex-

treme fast forward and backward propagation speed needed
by ENAS can be promised.

3.4 Training Strategy

An overview of training strategy of BENAS can be found
in Algorithm [1]. It is obvious that we are not only need to

train the controller for generating better BCNNs but also child
models with different paradigms. The training of BENAS is

a dual optimization problem due to the interrelation between
the complete model and controller. In order to reduce the
computation complexity of the above optimization issue, we
divide the training procedure of BENAS into two interleaving
phases.

First of all, the parameters of LSTM $\theta$ are fixed in the first
phase. And then each child model is sampled and trained on
45000 images of CIFAR-10. At last, the trained weights of
child model are stored into the complete model for next child
model restoring. In the second phase, the parameters of com-
plete model $w$ are fixed firstly. Subsequently, the LSTM pre-
dicts a list of tokens with $T$ length which can be regard as a
list of actions $a_{1:T}$ to represent a cell. And then, the sampled
cell is stacked as the building block of child model follow-
ing the paradigm of BCNN shown in Fig. [1]. Moreover, the
child model’s weights is restored from the complete model.
Finally, the accuracy of the model on 5000 validation images
of CIFAR-10 is consider as the loss function of LSTM and a
policy gradient algorithm is applied for optimizing $\theta$.

4 Experiments and Analysis

4.1 Architecture Search on CIFAR-10

Similarly, CIFAR-10 is chosen as the search dataset and ap-
plicated a series of standard data augment techniques which can

(GAP) layer. Here, the output of the last deep cell in each
convolution block is connected for feeding all-scales features
into the GAP layer so that the final output of GAP layer can
be expressed as

$$ O = \psi(z_{k_1}^{(1)}, z_{k_2}^{(2)}, \ldots, z_{k_u}^{(u)}, h^{(1)}, h^{(2)}, \ldots, h^{(v)}), \quad (7) $$

where $\psi(\cdot)$ is a function combination of $1 \times 1$ convolution,
concatenating and global average pooling. Here, a priori
knowledge is incorporated into BCNN. Depending on a great
number of experiments, we find that those low-pixels feature
maps are more important than those feature maps with high
resolutions for achieving high performance. In other words,
for designing BCNN with novel performance, more deep and
broad feature maps of $\text{Conv}_r$ instead of $\text{Conv}_s$ should be
fed into the GAP layer, where $r > s$ and $0 < s, r \leq u$.
In order to insert the above priori knowledge into BCNN, a
group of convolutions with $1 \times 1$ kernel size are employed
in each connection between the convolution block and GAP
layer. These $1 \times 1$ convolutions accept those feature rep-
resentations from the final deep cell in each convolution block
and output a group of feature maps with different importance.
Moreover, the importance is represented by the number of
output channels which the larger is the more important it is.
Furthermore, these $1 \times 1$ convolutions have different strides
for concatenating all input feature maps with same size.

Just because of the above, the proposed BCNN can achieve
high performance with a shallow topology so that the ex-
treme fast forward and backward propagation speed needed
by ENAS can be promised.

be found in ENAS [19] for details. In BENAS, we chose five
candidate operations: $3 \times 3$ depthwise-separable convolution,
$5 \times 5$ depthwise-separable convolution, $3 \times 3$ max pooling,
$3 \times 3$ average pooling and skip connection as the components
of convolution cell and enhancement cell with 7 nodes.

In the architecture search phase, for training the broad
model with topology of $2@0@2$ (the definition of this no-
tation refers to Section [3.3]), the Nesterov momentum is
adopted and the learning rate follows the cosine schedule with
$l_{\text{max}}=0.05$, $l_{\text{min}}=0.0005$, $T_0=10$ and $T_{\text{mul}}=2$ [16]. Further-
more, the experiment runs for 150 epochs with batch size 128.
For updating the parameters $\theta$ of LSTM, the Adam optimizer
with a learning rate of 0.0035 is applied.

The diagrams of the top performing convolution cell and
enhancement cell discovered by BENAS are shown in Figure
2. Based on the learned cells, a family of BCNNs with same
topologies of $2@1@1$ but different parameters by changing
the number of channels are constructed. The comparisons
of BENAS with other NAS approaches on CIFAR-10 for
different-size models under identical training conditions are
shown in Table [1]. Moreover, a popular data augmentation
Table 1: Comparison of the proposed BENAS with other NAS approaches on CIFAR-10 for different-size models under identical training conditions.

| Architecture | Error (%) | Params (M) | Search Cost (GPU Days) | Cost Ratio† | Topology |
|--------------|-----------|------------|------------------------|-------------|----------|
| LEMONADE + cutout [8] | 4.57 | 0.5 | 80 | 347.8 | deep |
| DPP-Net + cutout [7] | 4.62 | 0.5 | 4.00 | 17.4 | deep |
| BENAS(2@1@1) + cutout(ours) | 3.63 | 0.5 | 0.23 | 1 | broad |
| LEMONADE + cutout [8] | 3.69 | 1.1 | 80 | 347.8 | deep |
| DPP-Net + cutout [7] | 4.78 | 1.0 | 4.00 | 17.4 | deep |
| BENAS(2@1@1) + cutout(ours) | 3.40 | 1.1 | 0.23 | 1 | broad |
| AmoebaNet-A + cutout [20] | 3.34 ± 0.06 | 3.2 | 3150 | 13695.7 | deep |
| AmoebaNet-B + cutout [20] | 2.55 ± 0.05 | 2.8 | 3150 | 13695.7 | deep |
| NASNet-A + cutout [26] | 2.65 | 3.3 | 1800 | 7826.1 | deep |
| NASNet-B + cutout [26] | 3.73 | 2.6 | 1800 | 7826.1 | deep |
| Hierarchical Evo [12] | 3.75 ± 0.12 | 15.7 | 300 | 1304.3 | deep |
| PNAS [13] | 3.41 | 3.2 | 225 | 978.3 | deep |
| LEMONADE + cutout [8] | 3.05 | 4.7 | 80 | 347.8 | deep |
| DARTS(second order) + cutout [14] | 2.83 ± 0.06 | 3.3 | 4.00 | 17.4 | deep |
| DARTS(first order) + cutout [14] | 3.00 | 2.9 | 1.50 | 6.5 | deep |
| SMASH + cutout [1] | 4.03 | 16.0 | 1.50 | 6.5 | deep |
| ENAS + cutout [19] | 2.89 | 4.6 | 0.45 | 2.0 | deep |
| BENAS(2@1@1) + cutout(ours) | 2.95 | 4.1 | 0.23 | 1 | broad |

† The search cost of our approach is chosen as the baseline.

Table 2: Comparison of BENASNet with other state-of-the-art image classifiers on ImageNet

| Architecture | Top-1 (%) | Top-5 (%) | Params (M) |
|--------------|-----------|-----------|------------|
| Inception-v1 [21] | 30.2 | 10.1 | - |
| MobileNet-224 [10] | 29.4 | - | 6 |
| ShuffleNet (2x) [24] | 29.1 | 10.2 | 10 |
| AmoebaNet-A [20] | 25.5 | 8.0 | 5.1 |
| AmoebaNet-B [20] | 26.0 | 8.5 | 5.3 |
| NASNet-A [26] | 26.0 | 8.4 | 5.3 |
| NASNet-B [26] | 27.2 | 8.7 | 5.3 |
| NASNet-C [26] | 27.5 | 9.0 | 4.9 |
| PNASNet [13] | 25.8 | 8.1 | 5.1 |
| LEMONADE [8] | 26.9 | 9.0 | 4.9 |
| DARTS [14] | 26.7 | 8.7 | 4.7 |
| FBNet-B [23] | 25.9 | - | 4.5 |
| BENASNet(5@1@1)(ours) | 25.7 | 8.5 | 3.9 |

4.2 Transferability of Learned Architecture on ImageNet

A large scale image classification model stacked by the learned cells named BENASNet is built for ImageNet 2012. This experiment is not only performed for verifying the transferability of discovered architecture by BENAS, but also proving the powerful multi-scale features extraction capacity of the proposed BCNN.

Like the experiments on CIFAR-10, some data augment techniques, for instance, randomly cropping and flipping are applied on the input images whose size is 224 × 224. In this experiment, the BENASNet consists of five convolution blocks and a single enhancement block. Moreover, there are only one deep cell in the convolution block, i.e. the topology of BENASNet is 5@1@1. We train the BENASNet for 150 epochs with batch size 256 by using SGD optimizer with momentum 0.9 and weight decay 3 × 10⁻⁵. The initial learning rate is set to 0.1 and decayed by a factor of 0.1 when arriving at epoch 70, 100 and 130. Other hyperparameters, e.g. label smoothing, gradient clipping bounds can be found in DARTS [14] in details.

Table 2 summaries the results from the point of view of accuracy and parameter, and compares with other state-of-the-art image classifiers on ImageNet.

4.3 Results Analysis

Performance

For the experiments on CIFAR-10, based on the learned architecture of BENAS, three models with same topologies but various parameters, 0.5, 1.1 and 4.1 millions are constructed. In the first and second block of Table 1, DPP-Net [7] and LEMONADE [8] are chosen as the comparative NAS approaches. It is obvious that BENAS can deliver small-size BCNNs with the best accuracy for small scale image classification task. In particular, for the models with 0.5 million parameters, BENAS exceeds those comparative NAS methods almost 1% which is a great promotion. Furthermore, in the third block of Table 1 a large-size model is constructed and several state-of-the-art NAS approaches, AmoebaNet [20], NASNet [26], DARTS [14] and ENAS [19] are chosen for comparing with the proposed method. Apparently, BENAS achieves a competitive result which is 2.95% test error with...
4.1 millions parameters.

Furthermore, two aspects, accuracy and parameter are compared for the experiment on ImageNet. Moreover, we not only choose the NAS approaches (second block of Table 2) but also manual design models (first block of Table 2) as the comparative methods. From the point of view of accuracy, BENASNet achieves 25.7% top-1 test error which is only 0.2% worse than state-of-art model designed by NAS, AmoebaNet-A [20]. The transferability of learned architecture and the powerful multi-scale features extraction capacity of BCNN for large scale image classification task can be proven. For the perspective with respect to parameter, BENASNet obtains the above competitive accuracy with 3.9 millions parameters which is state-of-the-art for NAS approaches. Here, the multi-scale features extracted by BCNN are fused to yield more reasonable and comprehensive representations for image classification so that BENASNet can make more exact decisions for image classification problem with few parameters.

In addition to the above discussion, we find an interesting phenomenon in the learned cells is that there are all convolution operation and skip connection without any pooling operations as shown in Figure 2. One possible reason is that the convolution and skip connection are more suitable for broad topology where each block needs more convolution operations for extracting multi-scale features.

Efficiency

The extreme fast search speed of BENAS, 0.23 day on a single GeForce GTX 1080Ti GPU is state-of-the-art for NAS.

As shown in Table 1 the efficiency of BENAS is about 14000x and 8000x which are almost five orders of magnitude faster than AmoebaNet and NASNet, respectively. Compared BENAS with those relative efficient NAS methods, Hierarchical Evo [12], PNAS [13] and LEMONADE [8]. BENAS uses about 1300x, 1000x and 350x less computational resources, respectively. Furthermore, several state-of-the-art efficient NAS approaches, DPP-Net [7], SMASH [1], DARTS [14] and ENAS [19] are compared in detail with the proposed BENAS below.

First of all, the comparisons of DPP-Net and SMASH between BENAS are given. It is obvious that BENAS is about 17x and 6.5x faster than the above two approaches, respectively. Moreover, the performance of BENAS is better as aforementioned. SMASH suffers from a low-rank restriction discussed in [19] so that the architecture discovered by SMASH can not outperform BENAS. Compared with DARTS, a novel gradient-based NAS approach, BENAS is about 6.5x and 17x faster than the above method with first-order and second-order approximation, respectively. However, the performance of BENAS exceeds DARTS with first-order approximation rather than second-order approximation which uses 17x more computational resources than our approach.

In particular, the search cost of BENAS is about 2x less than ENAS. As aforementioned, BENAS also uses LSTM controller, reinforcement learning and parameter sharing for architecture sampling, controller’s parameter updating and accelerating architecture search process, respectively. As a result, we can draw a conclusion that the proposed BCNN contributes to improve the efficiency of cell based NAS approach not merely ENAS.

5 Conclusions

In this paper, we propose a broad version for ENAS named BENAS. The core idea is designing a novel BCNN for replacing the deep search model in ENAS to accelerate the search process further. For efficiency, our approach delivers 0.23 GPU day on CIFAR-10, 2x less than ENAS. For performance, our approach achieves state-of-the-art performance for both small and large scales image classification task in particular for small-size model on CIFAR-10.

We only develop a broad learning system named CCF-BLS as the search paradigm of BENAS in this paper. However, some other structural variations of BLS which possibly achieve more novel performance are also presented in [3]. In the future, we will expand all variations of BLS for proposing better NAS approach.

Acknowledgments

This work was supported in part by Youth research fund of the state key laboratory of complex systems management and control No. 20190213 and No. GJHZ1849 International Partnership Program of Chinese Academy of Sciences, and was supported in part by No FA2018111061SOW12 Programe of the Huawei Technologies Co Ltd, and also was supported in part by Noahs Ark Lab, Huawei Technologies.

References

[1] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344, 2017.

[2] CL Philip Chen and Zhulin Liu. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. IEEE transactions on neural networks and learning systems, 29(1):10–24, 2017.

[3] CL Philip Chen, Zhulin Liu, and Shuang Feng. Universal approximation capability of broad learning system and its structural variations. IEEE transactions on neural networks and learning systems, (99):1–14, 2018.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1251–1258, 2017.

[5] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

[6] Zixiang Ding, Yaran Chen, Nannan Li, and Dongbin Zhao. Simplified space based neural architecture search. In The 2019 IEEE Symposium Series on Computational Intelligence, 2019.
[7] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In Proceedings of the European Conference on Computer Vision (ECCV), pages 517–531, 2018.

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. arXiv preprint arXiv:1804.09081, 2018.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[10] 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.

[11] Nannan Li, Yaran Chen, Zixiang Ding, and Dongbin Zhao. Light-weight neural architecture search for resource-constrained device. In 2019 Chinese Automation Congress, 2019.

[12] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436, 2017.

[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.

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

[15] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. arXiv preprint arXiv:1901.02985, 2019.

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

[17] Y-H Pao and Yoshiyasu Takefuji. Functional-link net computing: theory, system architecture, and functionalities. Computer, 25(5):76–79, 1992.

[18] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. Neurocomputing, 6(2):163–180, 1994.

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

[20] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. arXiv preprint arXiv:1802.01548, 2018.