Stacked BNAS: Rethinking Broad Convolutional Neural Network for Neural Architecture Search

Zixiang Ding, *Graduate Student Member, IEEE*, Yaran Chen, *Member, IEEE*, Nannan Li, Dongbin Zhao, *Fellow, IEEE*, and C.L.Philip Chen, *Fellow, IEEE*

***Abstract*—Different from other deep scalable architecture-based NAS approaches, Broad Neural Architecture Search (BNAS) proposes a broad scalable architecture which consists of convolution and enhancement blocks, dubbed Broad Convolutional Neural Network (BCNN), as the search space for amazing efficiency improvement. BCNN reuses the topologies of cells in the convolution block so that BNAS can employ few cells for efficient search. Moreover, multi-scale feature fusion and knowledge embedding are proposed to improve the performance of BCNN with shallow topology. However, BNAS suffers some drawbacks: 1) insufficient representation diversity for feature fusion and enhancement and 2) time consumption of knowledge embedding design by human experts. In this paper, we propose Stacked BNAS, whose search space is a developed broad scalable architecture named Stacked BCNN, with better performance than BNAS. On the one hand, Stacked BCNN treats mini BCNN as a basic block to preserve comprehensive representation and deliver powerful feature extraction ability. For multi-scale feature enhancement, each mini BCNN feeds the outputs of deep and broad cells to the enhancement cell. For multi-scale feature fusion, each mini BCNN feeds the outputs of deep, broad and enhancement cells to the output node. On the other hand, we propose Knowledge Embedding Search (KES) to learn appropriate knowledge embeddings in a differentiable way. Moreover, the basic unit of KES is an over-parameterized knowledge embedding module that consists of all possible candidate knowledge embeddings. Experimental results show that 1) Stacked BNAS obtains better performance than BNAS-v2 on both CIFAR-10 and ImageNet, 2) the proposed KES algorithm contributes to reducing the parameters of the learned architecture with satisfactory performance, and 3) Stacked BNAS delivers a state-of-the-art efficiency of 0.02 GPU days.**

***Index Terms*—broad neural architecture search, stacked broad convolutional neural network, knowledge embedding search, image classification.**

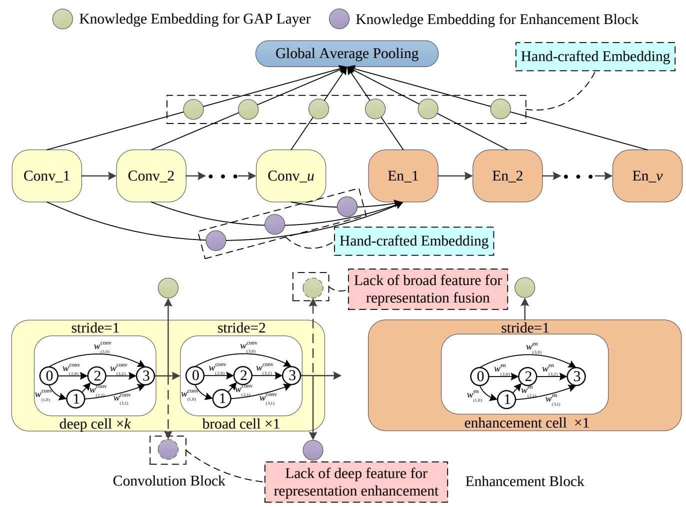


Fig. 1: Issues of BCNN. 1) Lack of representation diversity for feature fusion and enhancement: only deep and broad feature information is fed into the Global Average Pooling (GAP) layer for representation fusion and the first enhancement block for representation enhancement, respectively. 2) It is extremely time-consuming to design knowledge embeddings by human experts.

I. Introduction

NEURAL Architecture Search (NAS) has achieved unprecedented accomplishments for various structures (e.g., convolutional neural network [1], tensor ring [2], language model [3]) design. However, it needs enormous computational requirements, e.g., more than 22000 GPU days for vanilla NAS [1]. A micro search space dubbed cell [4] which is treated as the basic block of deep scalable architecture, is proposed to mitigate the above time-consuming issue and delivers a higher efficiency of 1800 GPU days. However, the search cost is unbearable yet. Subsequently, many cell-based NAS approaches were proposed to further improve the efficiency. Reinforcement Learning (RL) based ENAS [3] needs only 0.45 GPU days via parameter sharing. Gradient-based DARTS [5] employs a continuous relaxation strategy to transfer the search space from discrete to continuous, and delivers a novel efficiency of 0.45 GPU days. Furthermore, based on the DARTS framework, PC-DARTS [6] delivers a state-of-the-art search efficiency of 0.1 GPU days via partial channel connections.

Different from the above NAS approaches with deep scalable architecture, BNAS [7] proposed a broad scalable architecture named Broad Convolutional Neural Network (BCNN). Benefiting of BCNN, RL-based BNAS delivers a state-of-the-art efficiency of 0.2 GPU days, which is 2.2× faster than ENAS [3] whose efficiency ranks the best in RL-based NAS approaches. However, the training mechanism of BNAS following ENAS leads to terrible unfair training issue [8].

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62006226 and 62173324.

Z. Ding, Y. Chen, N. Li and D. Zhao are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (email : dingzixiang2018@ia.ac.cn, chenyaran2013@ia.ac.cn, linannan2017@ia.ac.cn, dongbin.zhao@ia.ac.cn).

C. L. P. Chen is with the School of Computer Science & Engineering, South China University of Technology, Guangzhou, Guangdong 510006, China, and also with the College of Navigation, Dalian Maritime University, Dalian 116026, China (e-mail: philip.chen@ieee.org).

BNAS-v2 [9] was proposed to solve the unfair training issue by a continuous relaxation strategy with a state-of-the-art efficiency of 0.05 GPU days. Admittedly, BNASs achieve satisfactory performance, especially in terms of efficiency. Nevertheless, BCNN suffers two drawbacks as shown in Fig. 1: 1) insufficient representation diversity for feature fusion and enhancement and 2) time-consuming knowledge embedding design.

In this paper, we propose Stacked BNAS whose scalable architecture is named Stacked BCNN, which treats mini BCNN as its basic block. Moreover, each mini BCNN can feed sufficient representations to the GAP layer and enhancement block for feature fusion and enhancement, respectively. As a new paradigm of neural networks, we prove also the universal approximation ability of Stacked BCNN in the Appendix. Furthermore, we transfer the knowledge embedding design task from discrete to continuous space, and learn appropriate knowledge embeddings in a differentiable way to solve the second issue as above mentioned.

Our contributions can be summarized as follows:

- **Stacked BNAS:** We propose Stacked BNAS that employs Stacked BCNN as the search space, to further improve the performance of NAS.
- **Stacked BCNN:** We not only propose a new broad search space dubbed Stacked BCNN for NAS, but also mathematically analyze the universal approximation ability of the proposed Stacked BCNN in the Appendix.
- **Knowledge Embedding Search:** We also propose a differentiable algorithm for designing knowledge embedding in an automatic way.
- **Efficiency:** Contributing to the proposed Stacked BCNN and optimization algorithm, Stacked BNAS delivers a state-of-the-art efficiency of 0.02 days on a single NVIDIA GTX 1080Ti GPU.

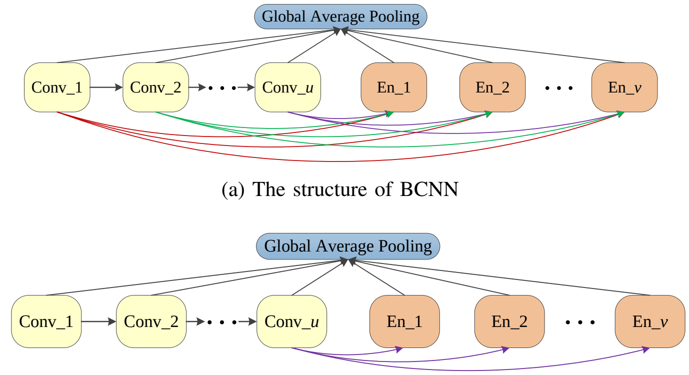
II. Related Work

A. Neural Architecture Search

Elsken et al. [10] claimed that NAS consisted of three components: search space, optimization strategy and estimation strategy. The search space referred to not only primitive operators, but also the combination paradigm of those candidate operations [11]. As described in [1], there were mainly five types of search spaces: entire-structured, cell-based [12, 13, 14], hierarchical [15, 16, 17], morphism-based [18, 19] and broad [7, 9]. We briefly introduce these search spaces as below.

1) Entire-structured Search Space: In the entire-structured search space, each layer with a specified operation (e.g., various convolutions and average pooling) was stacked one after another [1]. Beyond that, the skip connection was used in the above search space to explore more complex neural architectures. This search space had two disadvantages: computationally expensive and insufficient transferability [11].

2) Cell-based Search Space: To mitigate the above issues of the entire-structured search space, Zoph et al. (2018) proposed a cell-based search space where each cell with a list of operations was stacked one after another to construct a deep



(a) The structure of BCNN

(b) The structure of BCNN-CCLE

Fig. 2: Broad search space [7].

scalable architecture. The cell-based search space consists of normal and reduction cells that have different architectures and strides. Moreover, each cell treats the outputs of two predecessors as its inputs. Due to the effectiveness of cell-based search space in terms of efficiency and transferability, many cell-based NAS approaches were proposed, e.g., weight-sharing ENAS [3], differentiable DARTS [5] and partially connected PC-DARTS [6].

3) Hierarchical Search Space: Most cell-based NAS approaches [3, 5] followed a two-level hierarchy: the inner cell level and the outer network level. On the one hand, the inner level chose operation and connection of each intermediate node in the cell. On the other hand, the outer level controlled the spatial-resolution changes. A general formulation [15] was proposed to learn the network-level structure. Liu et al. [20] proposed a hierarchical architecture representation to avoid manually predefining the network block number.

4) Morphism-based Search Space: The morphism-based search space directly modified the existing architecture. Net2Net [18] employed identity morphism (IdMorph) to design architecture based on the existing model. He et al. [21] claimed that there were several issues in IdMorph: 1) limited width and depth changes and 2) identity layer drawback. To solve the above issue, a developed approach named network morphism [19] was proposed. Network morphism adopted a parameter sharing strategy [3] to inherit the knowledge from the parent network to the child network, which grew into a more robust model. Furthermore, network morphism was improved to deliver better performance in terms of optimization algorithm [22] and its level [23]. Furthermore, Chen et al. [24] used hand-crafted and learned blocks to discover novel architecture via parameter inheriting.

5) Broad Search Space: To improve the efficiency of NAS, Ding et al. [7] proposed a broad search space named BCNN, which employs broad topology to obtain extreme fast search speed and satisfactory performance. As described in [7], BCNN belonged to the cell-based network-level search space. There are three broad search spaces: BCNN (see Fig. 2 (a)), BCNN-CCLE (see Fig. 2 (b)) and BCNN-CCE (see Fig. 1). BCNNs consist of convolution and enhancement blocks which densely connect with the GAP layer for multi-scale feature fusion. The main difference among BCNNs is the connection

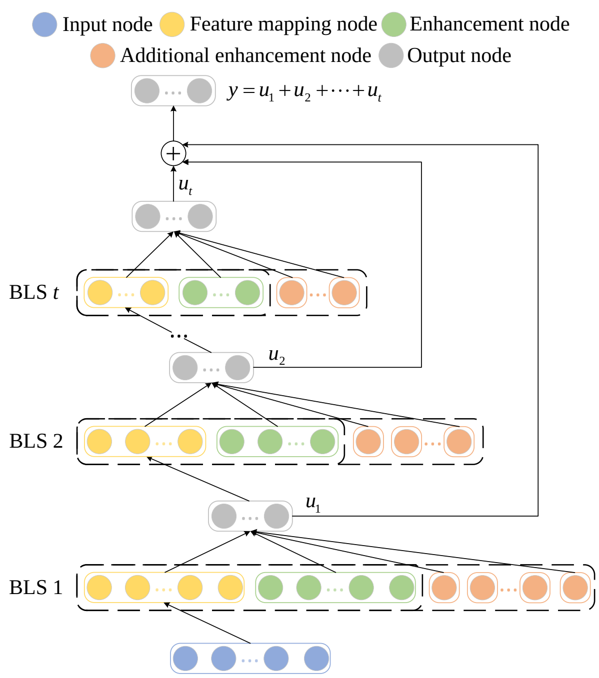
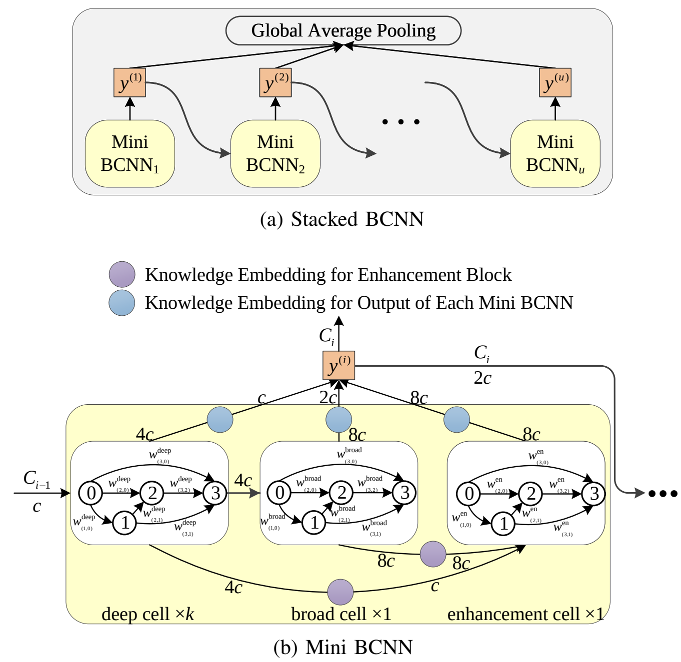


Fig. 3: The structure of Stacked BLS [27] with i BLSs.



(a) Stacked BCNN

(b) Mini BCNN

Fig. 4: Search space of Stacked BNAS.

between convolution and enhancement blocks. Similarly, Fang et al. (2020) proposed a network-level deep search space named dense search space where the MBConv [26] is treated as its basic block rather than cell. In contrast, each block of dense search space connects to several subsequent blocks. Based on the broad search space, BNAS [7] delivered a novel efficiency of 0.2 GPU days via RL. However, BNAS suffers from unfair training issue, so its optimization mechanism does not take full advantage of the BCNN, i.e., the efficiency improvement is limited. Furthermore, a differentiable over-parameterized broad search space was proposed to solve the unfair training issue in BNAS-v2 [9]. Experimental results show that BCNN can deliver a state-of-the-art search efficiency of 0.05 GPU days when its all advantages works.

B. Broad Learning System

Inspired by Random Vector Functional Link Neural Network (RVFLNN) [28] and incremental learning strategy [29], Chen and Liu [30] proposed Broad Learning System (BLS) and several variants [31]. Subsequently, BLS and its variants were widely used in various fields, e.g., image classification [32], industrial control [33] and intelligent transportation systems [34]. To combine the superiority of deep model and BLS, Liu et al. [27] proposed Stacked BLS, whose structure is shown in Fig. 3. BLS was the basic block of Stacked BLS whose output was the combination of all BLSs' outputs. Liu et al. [27] claimed that Stacked BLS only efficiently optimized trainable weights via an incremental learning algorithm, but also extracted deep representation using multiple BLSs.

III. Stacked Broad Neural Architecture Search

In this section, we first propose a developed broad search space named Stacked BCNN to solve the first issue of vanilla BCNN, i.e., insufficient presentation diversity for feature fusion and enhancement. Then, Knowledge Embedding Search (KES) algorithm is proposed to solve the second issue of vanilla BCNN, i.e the time consumption of knowledge embedding design by human experts. Finally, we show the optimization algorithm of Stacked BNAS.

A. Search Space: Stacked Broad Convolutional Neural Network

To improve the feature diversity, we propose Stacked BCNN, which treats mini BCNN as its basic block. The structures of Stacked BCNN and mini BCNN are shown in Fig. 4.

1) Stacked BCNN: As shown in Fig. 4 (a), the proposed Stacked BCNN consists of u mini BCNNs, where u is determined by the input size of the first mini BCNN (mini BCNN$_1$). To preserve the multi-scale feature fusion ability of the vanilla BCNN, we feed the output of each mini BCNN into GAP layer.

2) Mini BCNN: Fig. 4 (b) shows the structure of mini BCNN. Mini BCNN consists of $k+1$ convolution cells (k 1-stride deep cells, single 2-stride broad cell) and 1 enhancement cell with stride 1. Deep and broad cells are used for deep and broad feature extraction, respectively. The enhancement cell treats both the outputs of deep and broad cells as inputs to obtain a comprehensive enhancement representation. Furthermore, a family of 1×1 convolutions is inserted into fixed locations as the knowledge embeddings.

There are two main differences between BCNN and Stacked BCNN: 1) the output of each mini BCNN is the combination of all outputs of three cells rather than the outputs of deep and enhancement cells, and 2) the enhancement cell treats the outputs of convolution cells as inputs. The above two differences provide sufficient feature diversity to the GAP layer and enhancement block for representation fusion and enhancement so that better performance can be obtained.

3) Mathematical Information Flow: We employ a 3×3 convolution as the stem layer of the Stacked BCNN, and its output is treated as the two inputs of the first deep cell of mini BCNN_1, denoted as $y_{-1}^{(1)}$ and $y_0^{(1)}$.

For mini $\mathrm{BCNN}_i (i = 1, 2, \ldots, u)$, its output $y^{(i)}$ can be obtained by the outputs of three cells:

$$y^{(i)} = \phi(\delta_{do}^{(i)}(y_k^{(i)}), \delta_{bo}^{(i)}(y_{k+1}^{(i)}), \delta_{eo}^{(i)}(y_{k+2}^{(i)})), \tag{1}$$

where, ϕ and $\delta_{*o}^{(i)}$ are concatenation of the channel dimension and knowledge embeddings with respect to mini BCNN_i's output, respectively. Moreover, $y_k^{(i)}$ is the output of the last deep cell with a list of operations φ_d and can be computed by

$$y_k^{(i)} = \varphi_d(y_{k-2}^{(i)}, y_{k-1}^{(i)}; \boldsymbol{W}_d^{(i)}, \boldsymbol{\theta}_d^{(i)}), \tag{2}$$

$y_{k+1}^{(i)}$ is the output of a broad cell with a list of operations φ_b that can be obtained by

$$y_{k+1}^{(i)} = \varphi_b(y_{k-1}^{(i)}, y_k^{(i)}; \boldsymbol{W}_b^{(i)}, \boldsymbol{\theta}_b^{(i)}), \tag{3}$$

and $y_{k+2}^{(i)}$ is the output of the enhancement cell with a list of operations φ_e that can be calculated by

$$y_{k+2}^{(i)} = \varphi_e(\delta_{de}^{(i)}(y_k^{(i)}), \delta_{be}^{(i)}(y_{k+1}^{(i)}); \boldsymbol{W}_e^{(i)}, \boldsymbol{\theta}_e^{(i)}), \tag{4}$$

where, $\delta_{*e}^{(i)}$ represents knowledge embeddings with respect to the enhancement cell of mini BCNN_i and $\boldsymbol{W}_*^{(i)}$ and $\boldsymbol{\theta}_*^{(i)}$ are the weight and bias matrices of corresponding cells in mini BCNN_i, respectively.

The output of Stacked BCNN can be computed by

$$\boldsymbol{y} = \phi(y^{(1)}, y^{(2)}, \ldots, y^{(u)}). \tag{5}$$

4) Channel Flow Graph: As shown in Fig. 4 (a), for mini BCNN_i, the channel number of the deep cell's output $c_{deep}^{(i)}$ can be obtained by

$$c_{deep}^{(i)} = N_{in} \times 2^{i-1} \times c, \quad i = 1, 2, \ldots, u \tag{6}$$

where N_{in} represents the pre-defined intermediate node number of the cell, and c is the input channel number of mini BCNN_1. Moreover, the channel number of broad and enhancement cells' outputs in mini BCNN_i are both $2 \times c_{deep}^{(i)}$. For those direct-connected cells and output nodes (e.g., broad and enhancement cells, enhancement cells and output nodes), corresponding knowledge embedding does not reduce the feature significance. In contrast, the significance of the indirect-connected features is reduced by a factor of a quarter, as shown in Fig. 5 (a).

B. Knowledge Embedding Search

Different from vanilla BNAS, we propose a Knowledge Embedding Search (KES) algorithm, which treats an over-parameterized knowledge embedding module as a basic unit, to learn appropriate knowledge embedding in a differentiable way rather than designing by human. Moreover, KES plays a role of parameter redundancy reduction like network pruning [35].

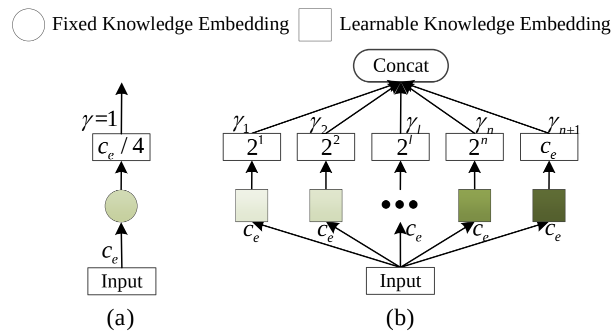


Fig. 5: Embedding between indirectly connected cells and the output node. (a) Hand-crafted knowledge embedding and (b) over-parameterized knowledge embedding module.

1) Over-parameterized Knowledge Embedding Module: For each knowledge embedding on an indirect edge, we construct an over-parameterized knowledge embedding module, as shown in Fig. 5 (b), to discover appropriate knowledge embeddings.

We assume that c_e channels are fed into the indirect-connected knowledge embedding. The over-parameterized knowledge embedding consists of n learnable knowledge embeddings with $2^i (i = 1, 2, \ldots, n)$ output channels and a single learnable knowledge embedding with c_e output channels, where n is the largest power of 2 less than c_e determined by

$$\underset{n}{\operatorname{argmax}} \ 2^n \quad \text{s.t. } 2^n < c_e. \tag{7}$$

Subsequently, the output of the over-parameterized knowledge embedding module y_e is obtained by the channel-dimension concatenation of weighted $n+1$ learnable knowledge embedding outputs as

$$y_e = \phi(\gamma_1 y_e^{(1)}, \gamma_2 y_e^{(2)}, \ldots, \gamma_{n+1} y_e^{(n+1)}), \tag{8}$$

where, $y_e^{(l)}$ and γ_l $(l = 1, 2, \ldots, n+1)$ represent the weighted output and weight of lth learnable knowledge embedding, respectively.

2) Learning Strategy: After over-parameterized knowledge embedding module construction, we aim to jointly optimize the knowledge embedding weights γ and network weights w. The goal of KES is to discover γ^* that minimizes the validation loss $\mathcal{L}_{val}(w^*, \gamma^*)$, where w^* is obtained by minimizing the training loss $\mathcal{L}_{train}(w, \gamma^*)$. The bilevel optimization problem with lower-level variable w and upper-level variable γ can be represented as

$$\begin{aligned} \min_{\gamma} \quad & \mathcal{L}_{val}(w^*(\gamma), \gamma), \\ \text{s.t.} \quad & w^*(\gamma) = \underset{w}{\operatorname{argmin}} \ \mathcal{L}_{train}(w, \gamma). \end{aligned} \tag{9}$$

The optimization of (9) exactly is prohibitive because $w^*(\gamma)$ needs to be recomputed by the second term of (9) whenever γ takes place any change [5]. Therefore, we propose an approximate iterative optimization process as follows. For each gradient descent step, network weights w and knowledge embedding weights γ are optimized by alternating in the

Algorithm 1: Stacked BNAS

Define p as the early stopping epoch number, q as the stable epoch number with zero initialization, kes as the flags represented using KES or not, $arch_{prev}$ as previous architecture with None initialization;
For each edge (i, j), use (13) and (14) for continuous relaxation as $f^{PC}_{(i,j)}(x^{PC}_{(j)}; M_{(i,j)})$ parameterized by $\alpha_{(i,j)}$ and $\beta_{(i,j)}$;
Define architecture weight set $\Theta = [\alpha, \beta]$;
if kes **then**
  Use (7) and (8) to construct over-parameterized knowledge embedding module parameterized by γ;
  $\Theta = [\alpha, \beta, \gamma]$;
end
while *not converged* **do**
  Optimize w by descending $\nabla_w \mathcal{L}_{train}(w, \Theta)$;
  Optimize Θ by descending $\nabla_\Theta \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \Theta), \Theta)$;
  Determine current architecture $arch_{curr}$ by taking argmax;
  $arch_{prev} = arch_{curr}$;
  if $arch_{curr} == arch_{prev}$ **then**
    $q = q + 1$;
    if $q \geq p$ **then**
      break
    end
  else
    $q = 1$;
  end
end
Output $arch_{curr}$ as the best architecture.

network and knowledge embedding spaces, respectively. At step t, given the current knowledge embedding γ_{t-1}, we update w_t by descending the training loss $\mathcal{L}_{train}(w_{t-1}, \gamma_{t-1})$. Subsequently, we keep w_t fixing and optimize the over-parameterized knowledge embedding module with learning rate ξ by descending

$$\nabla_\gamma \mathcal{L}_{val}(w_t - \xi \nabla_w \mathcal{L}_{train}(w_t, \gamma_{t-1}), \gamma_{t-1}). \tag{10}$$

Finally, we replace every over-parameterized knowledge embedding module as the knowledge embedding with the largest weight by taking argmax.

C. Optimization Algorithm

To discover a high-performance Stacked BCNN, we employ a gradient-based optimization pipeline: 1) a continuous relaxation strategy for over-parameterized Stacked BCNN construction, 2) partial channel connections (PC) for memory reduction and 3) early stopping for efficiency improvement. The optimization algorithm of Stacked BNAS is shown in **Algorithm 1**.

1) Continuous Relaxation: In mini BCNN, each cell consists of 2 input nodes $\{x_{(0)}, x_{(1)}\}$, $N-3$ intermediate nodes $\{x_{(2)}, \ldots, x_{(N-2)}\}$ and a single output node $\{x_{(N-1)}\}$. Each intermediate node $x_{(i)}$ can be computed by

$$x_{(i)} = \sum_{j<i} o_{(i,j)}(x_{(j)}), \tag{11}$$

where, $o_{(i,j)}$ is the operator between $x_{(i)}$ and $x_{(j)}$ chosen from candidate operation set $\mathcal{O}$. Moreover, the outputs of all intermediate nodes are combined to deliver $x_{(N-1)}$ by channel-dimension concatenation.

Subsequently, the over-parameterized Stacked BCNN is constructed by the strategy of continuous relaxation [5]. Particularly, we relax edge (i, j) of each cell for mini BCNN by

$$f_{(i,j)}(x_{(j)}) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha^o_{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha^{o'}_{(i,j)})} o(x_{(j)}), \tag{12}$$

where operation $o(\cdot)$ is weighted by architecture weight α^o.

2) Partial Channel Connections: We employ PC [6] to improve the memory efficiency of Stacked BNAS. The continuous relaxation of Stacked BNAS with PC can be obtained by

$$f^{PC}_{(i,j)}(x_{(j)}; M_{(i,j)}) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha^o_{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha^{o'}_{(i,j)})} o(M_{(i,j)} * x_{(j)}) + (1 - M_{(i,j)}) * x_{(j)}, \tag{13}$$

where, $M_{(i,j)}$ represents the mask of the channel sample whose values are chosen from $\{0, 1\}$.

Moreover, we use edge normalization to mitigate the undesired fluctuation in the search phase via $\beta_{(i,j)}$ as

$$x^{PC}_{(i)} = \sum_{j<i} \frac{\exp(\beta_{(i,j)})}{\sum_{j'<i} \exp(\beta_{(i,j')})} \cdot f_{(i,j)}(x_{(j)}). \tag{14}$$

Finally, each operation of the best architecture is obtained by taking argmax as

$$o_{(i,j)} = \operatorname*{argmax}_{o \in \mathcal{O}} \frac{\exp(\alpha^o_{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha^{o'}_{(i,j)})} \cdot \frac{\exp(\beta_{(i,j)})}{\sum_{j'<i} \exp(\beta_{(i,j')})}. \tag{15}$$

3) Early Stopping (ES): As described in DARTS+ [36], there are two indices for ES: 1) the *skip connection* number in a single cell and 2) the number of stable epochs. On the one hand, the search procedure is stopped when there is more than one *skip connection* in one cell to avoid the performance collapse issue. PC contributes to reducing the predominance of *skip connections*, so we do not choose the first index in this paper. On the other hand, the search procedure is stopped when the ranking of architecture weights is no longer changed for a determined number of epochs. This index means that the search procedure stops when arriving at a saturated state. Above all, the second index is chosen for early stopping of Stacked BNAS using the following criterion:

Criterion 1: *Stop the search procedure when the rank of architecture weights is no longer changed for three epochs.*

IV. EXPERIMENTS AND RESULTS

In this section, we perform a set of experiments to examine several novel properties of Stacked BNAS, including the Stacked BCNN and the KES. First, the datasets used and implementation details are given. Next, architecture search experiments of Stacked BNAS without/with KES are introduced. Moreover, qualitative and quantitative analysis are given for the experimental results on CIFAR-10. Then, the best-performing architecture learned by Stacked BNAS on CIFAR-10 is transferred to solve large-scale image classification task on ImageNet, and the experimental results are analysed. Furthermore, the learned architecture is also transferred to four image classification datasets (e.g., MNIST, FashionMNIST, NORB and SVHN) to verify the generalization ability of the proposed Stacked BNAS. Finally, two groups of ablation experiments are performed to verify the effectiveness of Stacked BNAS in terms of efficiency and performance.

A. Datasets and Implementation Details

1) Datasets: We employ CIFAR-10 and ImageNet to verify the effectiveness of the proposed Stacked BNAS. CIFAR-10 is a small-scale image classification dataset with 32×32 pixels that contains 50K training images and 10K test images. ImageNet contains approximately 1.3 M training data and 50K validation data with various pixels over 1000 object categories.

2) Implementation Details: The data preprocessing technique follows BNAS-v2 for CIFAR-10 and ImageNet. We implement architecture search without or with KES. For architecture search, we repeat the implementation five times. For architecture evaluation, the mean value of three repetitive retraining experiments is treated as the index to determine the best architecture. Furthermore, the best architecture learned on CIFAR-10 is transferred to ImageNet, MNIST, FashionMNIST, NORB and SVHN.

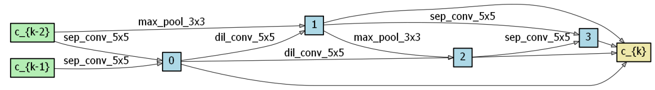
B. Experiments on CIFAR-10

1) Experimental Settings: As mentioned above, we implement two experiments on CIFAR-10: 1) Stacked BNAS without KES and 2) Stacked BNAS with KES.

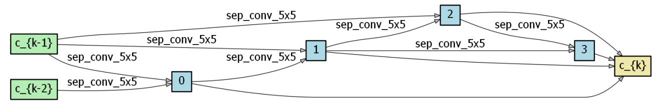
The above two experiments use many identical experimental settings for architecture search, as shown below. The initial input channel number is set to 16. We train the above over-parameterized Stacked BCNN for 50 epochs. All training images are equally divided into two parts. On the one hand, 25K images are treated as the training data to update the network weights w. On the other hand, another part is used as the validation data to optimize the architecture/embedding weights. To optimize the network weights w, we employ the SGD optimizer with a dynamic learning rate using an annealed decay manner, momentum of 0.9 and weight decay of 3×10^{-4}. Beyond that, the Adam optimizer is used to update the architecture/embedding weights, with momentum of (0.5, 0.999) and weight decay of 1×10^{-3}. We implement architecture search on a single NVIDIA GTX 1080Ti GPU.

For architecture search of Stacked BNAS without KES, the over-parameterized Stacked BCNN consists of 2 mini BCNNs, where each one contains 1 broad cell and 1 enhancement cell. The batch size and network learning rate are set to 512 and 0.2, respectively. Moreover, we set the architecture learning rate to 6×10^{-4}. For architecture search of Stacked BNAS with KES, the over-parameterized Stacked BCNN consists of 2 mini BCNNs, where each one contains 1 deep cell, 1 broad cell and 1 enhancement cell. Due to more memory requirements of KES, we set the batch size and network learning rate to 128 and 0.05, respectively. Beyond that, we set a larger learning rate of 2×10^{-3} for architecture/embedding weights when knowledge embedding is learned.

For architecture evaluation, we employ identical settings except the number of deep cell which is 2 and 3 for Stacked BNAS without/with KES, respectively. The stacked BCNN consists of two mini BCNNs, where each one contains 1 broad cell and 1 enhancement cell. Following BNAS-v2, the learned Stacked BCNN with 44 input channels is trained for 2000 epochs using the SGD optimizer with a batch size of 128, learning rate of 0.05, momentum of 0.9 and weight decay of 3×10^{-4}. Moreover, we implement architecture evaluation on a single NVIDIA Tesla P100 GPU.

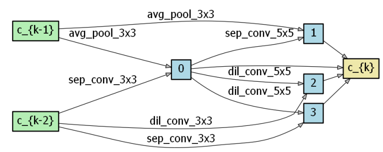


(a) Convolution cell

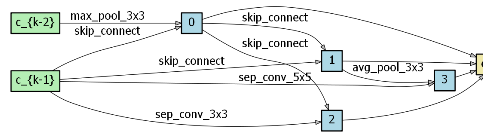


(b) Enhancement cell

Fig. 6: Architecture learned by Stacked BNAS without KES.



(a) Convolution cell



(b) Enhancement cell

Fig. 7: Architecture learned by Stacked BNAS with KES.

TABLE I: Hand-crafted and learned knowledge embedding with $c = 44$ input channels.

Feature	Enhancement		Output	
	Design	Search	Design	Search
Deep_1	c	2 ($\downarrow$)	c	8 ($\downarrow$)
Broad_1	$4c$	$4c$	$2c$	$8c$ ($\uparrow$)
Deep_2	$2c$	128 ($\uparrow$)	$2c$	64 ($\downarrow$)
Broad_2	$8c$	$8c$	$4c$	4 ($\downarrow$)

TABLE II: Comparison of the proposed Stacked BNAS and other state-of-the-art NAS approaches on CIFAR-10.

Architecture	**Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Number of Cells**	**Search Method**	**Topology**
LEMONADE [37]	3.05	4.7	80	-	evolution	deep
DARTS (1st order) [5]	3.00	3.3	0.45†	20	gradient-based	deep
DARTS (2nd order) [5]	2.76	3.3	1.50†	20	gradient-based	deep
DARTS (random) [5]	3.49	3.1	-	20	-	deep
SNAS + mild constraint [38]	2.98	2.9	1.50	20	gradient-based	deep
SNAS + moderate constraint [38]	2.85	2.8	1.50	20	gradient-based	deep
SNAS + aggressive constraint [38]	3.10	**2.3**	1.50	20	gradient-based	deep
P-DARTS [39]	**2.50**	3.4	0.30	20	gradient-based	deep
GDAS-NSAS [40]	2.73	3.5	0.40	20	gradient-based	deep
PC-DARTS [6]	2.57	3.6	0.10	20	gradient-based	deep
ENAS [3]	2.89	4.6	0.45	17	RL	deep
BNAS [7]	2.97	4.7	0.20	**5**	RL	broad
BNAS-CCLE [7]	2.95	4.1	0.20	**5**	RL	broad
BNAS-CCE [7]	2.88	4.8	0.19	8	RL	broad
BNAS-v2 [9]	2.79	3.7	0.05	8	gradient-based	broad
Random	3.12	3.1	-	8	-	broad
Stacked BNAS w/o KES (Ours)	2.71	3.7	**0.02**	8	gradient-based	broad
Stacked BNAS w/ KES (Ours)	2.73	3.1	0.15	10	gradient-based	broad

† Obtained by DARTS using the code publicly released by the authors at https://github.com/quark0/darts on a single NVIDIA GTX 1080Ti GPU.

2) Results and Analysis: For Stacked BNAS, we visualize the learned architecture in Fig. 6. For Stacked BNAS with KES, we show the best architecture and knowledge embedding in Fig. 7 and TABLE I, respectively. Furthermore, TABLE II summarizes the comparison of the proposed Stacked BNAS with other novel NAS approaches on CIFAR-10.

Contributing to the combination of Stacked BCNN and optimization strategy, Stacked BNAS delivers a state-of-the-art efficiency of 0.02 GPU days and competitive test accuracy of 97.29% with 3.7 M parameters. Beyond that, Stacked BNAS with KES discovers a Stacked BCNN with 2.73% test error and just 3.1 M parameters (approximately 16.2% less than Stacked BNAS). Moreover, the over-parameterized knowledge embedding module leads to more trainable parameters and memory requirements than vanilla Stacked BNAS, so larger costs are needed. As shown in TABLE I, one indirect-connected knowledge embedding of each mini BCNN has more output channels than hand-crafted embedding. The above knowledge embedding changes lead to parameter reduction of the architecture learned by Stacked BNAS with KES.

Compared with NAS approaches with deep topology, Stacked BNAS obtains the best efficiency and competitive accuracy. In terms of random architecture, Stacked BCNN obtains 0.38% better accuracy than DARTS, which further examines the effectiveness of the proposed Stacked BCNN. Furthermore, Stacked BNAS is 5× faster than PC-DARTS whose efficiency ranks the best among all NAS approaches. Compared with BNASs, the proposed Stacked BNAS delivers better efficiency and accuracy. On the one hand, the efficiency of Stacked BNAS is approximately 10× and 2.5× faster than BNAS-v1 and BNAS-v2, respectively. On the other hand, the accuracy of Stacked BNAS is approximately 0.2% and 0.1% better than BNAS-v1 and BNAS-v2, respectively. Compared with previous BNASs, Stacked BNAS with KES can deliver better accuracy with approximately 16% and 33% parameter reduction, respectively.

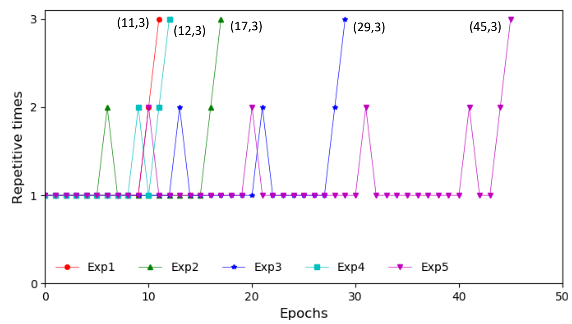


Fig. 8: Early stopping for Stacked BNAS without KES.

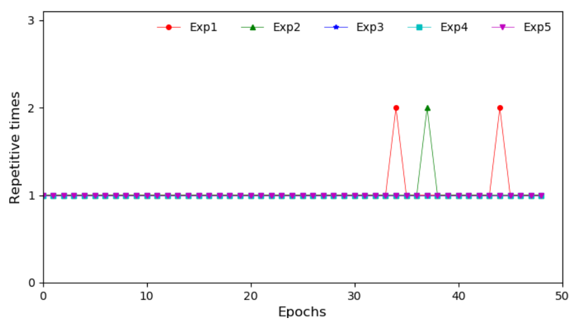
3) Efficiency Difference between Stacked BNAS without or with KES: As shown in TABLE II, the efficiencies of Stacked BNAS without/with KES are 0.02 and 0.15 GPU days, respectively. For that, there are two main reasons: 1) different structures of used Stacked BCNNs in the search phase, and 2) various degrees of difficulty meeting **Criterion 1**.

On the one hand, in the architecture search phase of Stacked BNAS with KES, each mini BCNN contains a deep cell for learning appropriate knowledge embedding. Consequently, the first advantage of fast single-step training speed does not work. On the other hand, Stacked BNAS with KES has difficult satisfying **Criterion 1** for early stopping.

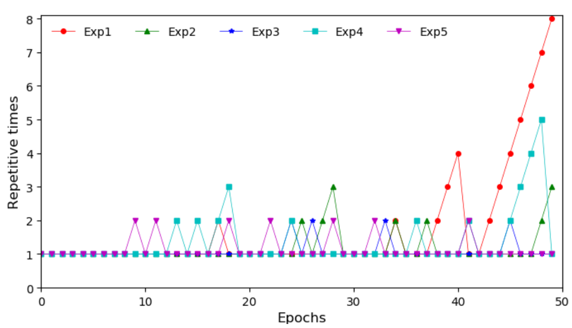
- In the architecture search phase of Stacked BNAS, both the repetitive times of convolution and enhancement cells should be larger than 2. The architecture repetitive times of Stacked BNAS are shown in Fig. 8. Each implementation can stop early before arriving at the maximum epoch

TABLE III: Comparison of the proposed Stacked BNAS and other state-of-the-art NAS approaches on ImageNet.

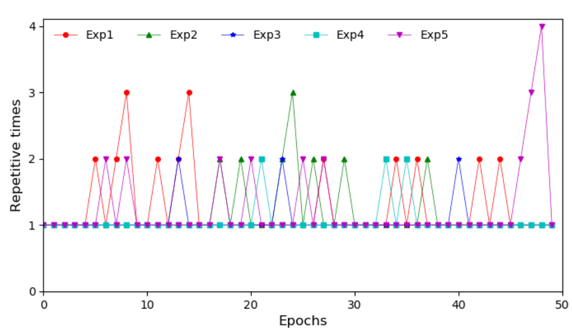
Architecture	**Test Err. (%)**		**Params**	**Search Cost**	**FLOPs**	**Search Method**	**Topology**
	top-1	**top-5**	**(M)**	**(GPU days)**	**(M)**		
AmoebaNet-A [16]	25.5	8.0	5.1	3150	555	evolution	deep
NASNet-A [4]	26.0	8.4	5.3	1800	564	RL	deep
PNAS [41]	25.8	8.1	5.1	225	588	SMBO	deep
GHN [42]	27.0	8.7	6.1	0.84	569	SMBO	deep
DARTS (2nd order) [5]	26.7	8.7	4.7	1.50	574	gradient-based	deep
SNAS (mild) [38]	27.3	9.2	4.3	1.50	522	gradient-based	deep
P-DARTS [39]	24.4	7.4	4.9	0.30	557	gradient-based	deep
GDAS-NSAS [40]	24.7	7.5	5.1	0.30	577	gradient-based	deep
BayesNAS [43]	26.5	8.9	**3.9**	0.20	-	gradient-based	deep
PC-DARTS [6]	25.1	7.8	5.3	0.10	586	gradient-based	deep
PC-DARTS (ImageNet) [6]	**24.2**	**7.3**	5.3	3.80	597	gradient-based	deep
BNAS-v2 (PC) (2nd order) [9]	27.2	8.8	4.6	0.09	**475**	gradient-based	broad
Stacked BNAS (Ours)	26.4	8.9	4.7	**0.02**	485	gradient-based	broad



(a) Repetitive times of both architecture and embedding



(b) Repetitive times of architecture



(c) Repetitive times of embedding

Fig. 9: Early stopping for Stacked BNAS with KES where both architecture and embedding weight should satisfy the **Criterion 1** *simultaneously*.

so that Stacked BNAS delivers state-of-the-art efficiency.

- For Stacked BNAS with KES, we show the architecture and embedding repetitive times in Fig. 9. In this experiment, the ES strategy is that the convolution cell, enhancement cell, and knowledge embedding do not change in three epochs. Each implementation cannot satisfy the aforementioned early stopping criterion. Beyond that, over-parameterized knowledge embedding without a partial channel connections strategy leads to more memory usages than vanilla BNAS. As a result, we set batch size as 128 instead of 256. Above all, the efficiency of Stacked BNAS is not satisfactory when using KES.

C. Experiments on ImageNet

1) Experimental Settings: To transfer the architecture learned by Stacked BNAS on ImageNet, we treat three 3×3 convolutions as stem layers that reduce the input size from 224×224 to 28×28. Subsequently, a Stacked BCNN is constructed by 2 mini BCNNs, where each one contains 2 deep cells, 1 broad cell and 1 enhancement cell.

For architecture evaluation, we choose the SGD optimizer with a learning rate of 0.1 followed by a linear decay method, momentum of 0.9 and weight decay of 3×10^{-5}. Moreover, we train the Stacked BCNN for 250 epochs with a batch size of 512 on 4 NVIDIA Tesla P100 GPUs. Other experimental settings follow PC-DARTS, e.g., label smoothing is set to 0.1, and gradient clip bound is set to 5.0.

2) Results and Analysis: TABLE III summarizes the comparison of the proposed Stacked BNAS with other novel NAS approaches on ImageNet.

Compared with previous impactful NAS approaches (e.g., AmoebaNet, NASNet, PNAS), the proposed Stacked BNAS delivers competitive or better performance with a state-of-the-art efficiency of 0.02 GPU days which is 5 or 6 magnitudes faster. For efficient NAS approaches (e.g., DARTS, SNAS, PC-DARTS), Stacked BNAS also obtains competitive or better performance with 1 or 2 magnitudes faster efficiency. Compared with BNAS-v2, Stacked BNAS obtains better performance in terms of top-1 and top-5 accuracy. As

TABLE IV: Comparison of Stacked BNAS and other novel classifiers on four image classification datasets

Architecture	Params (M)	Accuracy (%)				Number of Cells	Search Cost (GPU Days)
		MNIST	FashionMNIST	NORB	SVHN		
NASNet [4]	1.5/1.3†	**99.64** (1)	**95.47** (1)	93.34 (4)	96.87 (4)	20	1800
AmoebaNet [16]	1.5	99.62 (4)	95.33 (3)	**93.73** (1)	96.85 (5)	20	3150
DARTS [5]	1.5	99.58 (6)	95.24 (6)	91.83 (6)	96.76 (6)	20	0.45
PC-DARTS [6]	1.4	99.61 (5)	95.26 (5)	93.00 (5)	96.98 (2)	20	0.1
BNAS-v2 [9]	1.5	99.63 (3)	95.33 (3)	93.37 (3)	96.98 (2)	**8**	0.09
Stacked BNAS (ours)	1.5	**99.64** (1)	95.35 (2)	93.52 (2)	**97.12** (1)	10	**0.02**

† The error of out of memory arises on a NVIDIA Tesla P100 GPU when using 1.5 M parameters for NORB and SVHN classification.

mentioned above, the main difference between BNAS-v2 and the proposed Stacked BNAS is the broad scalable architecture, so that the effectiveness of Stacked BCNN can be promised. Moreover, the search cost of BNAS-v2 is 4.5× higher than Stacked BNAS. Due to the broad topology of Stacked BCNN, Stacked BNAS delivers the best performance in terms of FLOPs, which is an important index to show the hardware friendliness of deep models.

D. Experiments on Other Datasets

To further verify the effectiveness of Stacked BNAS, we transfer the architecture learned on CIFAR-10 to other image classification datasets, e.g., MNIST, FashionMNIST, NORB and SVHN. For the above four datasets, we employ identical experimental settings for Stacked BNAS as follows. The BCNN consists of 2 mini BCNNs where each one has 3 deep cells, 1 broad cells and 1 enhancement cells. A single NVIDIA Tesla P100 GPU with 2.20-GHz Intel Xeon E5-2650 CPUs is used to train the classifier from scratch for 300 epochs. We choose SGD as the optimizer with a batch size of 128, an initial learning rate of 0.05 following the cosine decay method, a momentum of 0.9 and a weight decay of 3×10^{-4}. The drop path probability is set to 0.3. Similar to BNAS-v2, deep scalable architecture-based NAS approaches (e.g., NASNet [4], AmoebaNet [16], DARTS [5] and PC-DARTS [6]) consist of 20 cells and employ identical settings to Stacked BNAS except the learning rate of 0.025. Moreover, BNAS-v2 consists of 8 cells and sets the learning rate to 0.025. Experimental results are shown in TABLE IV.

Compared with state-of-the-art NAS approaches, the proposed Stacked BNAS delivers the best generalization ability. For MNIST, FashionMNIST and NORB, the accuracies of BNAS-v2 are second-ranked. For SVHN, BNAS-v2 and PC-DARTS are tied for the first place with 96.98% accuracy. Furthermore, the architecture learned by BNAS-v2 on CIFAR-10 shows powerful generalization ability. NASNet and AmoebaNet deliver satisfactory performance for all datasets. However, the performance of DARTS and PC-DARTS is unsatisfactory.

E. Ablation Studies

Here, we implement two groups of experiments: 1) one is to examine the effectiveness of PC and ES for efficiency improvement of Stacked BNAS, and 2) the other is to verify the effectiveness of two scales of information, i.e., the output of the broad cell to the output node is denoted as b2o and the output of the deep cell to the enhancement cell is denoted as d2e.

TABLE V: Ablation experiments for the efficiency of Stacked BNAS on CIFAR-10.

Case	PC	ES	Epochs	Batch Size	Efficiency (GPU days)
1	✗	✗	50	128	0.140
2	✓	✗	50	**512**	0.068
3	✗	✓	15	128	0.047
4	✓	✓	**12**	**512**	**0.018**

1) PC and ES for Efficiency Improvement: In this group of experiments, there are four cases: 1) using neither PC nor ES, 2) using only PC, 3) using only ES and 4) using both PC and ES. Each case is repeatedly performed five times following the experimental setting used for architecture search of Stacked BNAS without KES. Experimental results are shown in TABLE V.

In case 1, the search cost of Stacked BNAS without PC and ES is 0.14 GPU days under a single NVIDIA GTX 1080Ti GPU. In case 2, Stacked BNAS employs PC to deliver an efficiency of 0.068 GPU days, which is 2× faster than case 1, because the strategy of PC contributes to improving a higher memory efficiency (using a larger batch size, i.e., 512) of Stacked BNAS than case 1. In the case of using ES, Stacked BNAS can stop the search phase at the 15th epoch and obtain an efficiency of 0.047 GPU days. When using both PC and ES, Stacked BNAS can not only search architecture with efficient memory of 512 batch size, but also stop early at the 12th epoch and delivers a state-of-the-art efficiency of 0.018 GPU days. Above all, both PC and ES play important roles in the efficiency improvement of Stacked BNAS.

2) Multi-scale Feature Fusion for Performance Improvement: In this group of experiments, there are four cases: 1) using neither b2o nor d2e, 2) using only b2o, 3) using only d2e and 4) using both b2o and d2e. Each case is repeatedly performed three times following the experimental setting used for architecture evaluation of Stacked BNAS without KES. Moreover, the mean accuracy of three repetitive experiments is treated as the final result. Experimental results are shown in TABLE V.

When using neither b2o nor d2e, Stacked BCNN degrades into vanilla BCNN, which lacks feature diversity for repre-

TABLE VI: Ablation experiments for multi-scale feature fusion of Stacked BCNN on CIFAR-10.

Case	**Scale Information**		**Parameters**	**Test Error**
	b2o	**d2e**	**(M)**	**(%)**
1	✗	✗	3.56	3.14
2	✓	✗	**3.36**	3.05
3	✗	✓	3.64	2.88
4	✓	✓	3.70	**2.71**

sentation fusion and enhancement, so its test error is 3.14%. For case 2, each mini BCNN employs the scale information from the broad cell to the output node and delivers 96.95% accuracy, which is approximately 0.1% higher than case 1. Compared with the scale information of b2o, d2e shows a greater contribution to performance improvement and obtains 97.12% accuracy, which is approximately 0.3% higher than case 1. In the last case, the combination of b2o and d2e further improves the performance of Stacked BCNN, i.e., 2.71% test error. Above all, multi-scale feature fusion is effective for performance improvement of Stacked BCNN.

V. Conclusions

BNASs deliver state-of-the-art efficiency via a novel broad scalable architecture named BCNN, which employs multi-scale feature fusion and hand-crafted knowledge embedding to yield satisfactory performance with shallow topology. However, there are two issues in BCNN: 1) feature diversity loss for representation fusion and enhancement and 2) time consumption of knowledge embedding design.

To solve the above issues, this paper proposes Stacked BNAS. On the one hand, Stacked BNAS proposes a new broad scalable architecture named Stacked BCNN that can provide more feature diversities for representation fusion and enhancement than vanilla BCNN. On the other hand, a differentiable algorithm named KES is also proposed to learn appropriate knowledge embedding for Stacked BCNN in an automatic way instead of designing by machine learning experts. Benefiting from the combination of Stacked BCNN and an efficient optimization algorithm, the proposed Stacked BNAS delivers a state-of-the-art efficiency of 0.02 GPU days with competitive performance. Moreover, KES contributes to discovering a high-performance Stacked BCNN with fewer parameter counts that plays a similar role as network pruning [44, 45]. Moreover, the proposed Stacked BNAS shows powerful generalization ability on four image classification datasets.

References

[1] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2017.

[2] N. Li, Y. Pan, Y. Chen, Z. Ding, D. Zhao, and Z. Xu, “Heuristic rank selection with progressively searching tensor ring network,” *Complex & Intelligent Systems*, pp. 1–15, 2021.

[3] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in *International Conference on Machine Learning (ICLR)*, 2018, pp. 4095–4104.

[4] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8697–8710.

[5] H. Liu, K. Simonyan, Y. Yang *et al.*, “DARTS: Differentiable architecture search,” in *International Conference on Learning Representations (ICLR)*, 2018.

[6] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong, “PC-DARTS: partial channel connections for memory-efficient architecture search,” in *International Conference on Learning Representations (ICLR)*, 2020.

[7] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. P. Chen, “BNAS: Efficient neural architecture search using broad scalable architecture,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[8] X. Chu, B. Zhang, R. Xu, and J. Li, “FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search,” *arXiv preprint arXiv:1907.01845*, 2019.

[9] Z. Ding, Y. Chen, N. Li, and D. Zhao, “BNAS-v2: Memory-efficient and performance-collapse-prevented broad neural architecture search,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[10] T. Elsken, J. H. Metzen, F. Hutter *et al.*, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, pp. 1–21, 2019.

[11] X. He, K. Zhao, and X. Chu, “AutoML: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[12] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Practical block-wise neural network architecture generation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2423–2432.

[13] N. Li, Y. Chen, Z. Ding, and D. Zhao, “Light-weight neural architecture search for resource-constrainted device,” in *2019 Chinese Automation Congress (CAC)*, 2019.

[14] Z. Ding, Y. Chen, N. Li, and D. Zhao, “Simplified space based neural architecture search,” in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 43–49.

[15] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 82–92.

[16] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, 2019, pp. 4780–4789.

[17] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2820–2828.

[18] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[19] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 564–572.

[20] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *International Conference on Learning Representations (ICLR)*, 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[22] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.

[23] T. Wei, C. Wang, and C. W. Chen, "Modularized morphing of neural networks," *arXiv preprint arXiv:1701.03281*, 2017.

[24] Y. Chen, R. Gao, F. Liu, and D. Zhao, "Modulenet: Knowledge-inherited neural architecture search," *arXiv preprint arXiv:2004.05020*, 2020.

[25] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Densely connected search space for more flexible neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 628–10 637.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[27] Z. Liu, C. P. Chen, S. Feng, Q. Feng, and T. Zhang, "Stacked broad learning system: From incremental flatted structure to deep model," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 209–222, 2020.

[28] Y. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[29] C. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 1, pp. 62–72, 1999.

[30] C. P. Chen and Z. 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*, vol. 29, no. 1, pp. 10–24, 2017.

[31] C. P. Chen, Z. Liu, S. Feng *et al.*, "Universal approximation capability of broad learning system and its structural variations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1191–1204, 2018.

[32] H. Zhao, J. Zheng, W. Deng, and Y. Song, "Semi-supervised broad learning system based on manifold regularization and broad network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 3, pp. 983–994, 2020.

[33] F. Chu, T. Liang, C. P. Chen, X. Wang, and X. Ma, "Weighted broad learning system and its application in nonlinear industrial process modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3017–3031, 2019.

[34] C. P. Chen and B. Wang, "Random-positioned license plate recognition using hybrid broad learning system and convolutional networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[35] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 14 913–14 922.

[36] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+: Improved differentiable architecture search with early stopping," *arXiv preprint arXiv:1909.06035*, 2019.

[37] T. Elsken, J. H. Metzen, F. Hutter *et al.*, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations (ICLR)*, 2018.

[38] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in *International Conference on Learning Representations (ICLR)*, 2018.

[39] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE International Conference on Computer Vision (ECCV)*, 2019, pp. 1294–1303.

[40] M. Zhang, H. Li, S. Pan, X. Chang, and S. Su, "Overcoming multi-model forgetting in one-shot nas with diversity maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7809–7818.

[41] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. FeiFei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[42] C. Zhang, M. Ren, and R. Urtasun, "Graph hypernetworks for neural architecture search," in *International Conference on Learning Representations (ICLR)*, 2019.

[43] H. Zhou, M. Yang, J. Wang, and W. Pan, "BayesNAS: A bayesian approach for neural architecture search," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 7603–7613.

[44] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 264–11 272.

[45] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, “Variational convolutional neural network pruning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2780–2789.
[46] W. Rudin, *Real and complex analysis*. Tata McGraw-hill education, 2006.
[47] B. Igelnik and Y. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

Appendix: Universal Approximation Ability of Stacked BCNN

As a new paradigm of neural networks, similar to BNAS [7], we also provide theoretical demonstration for the proposed Stacked BCNN in terms of universal approximation ability as follows.

Given the initial input channel number c, the output of mini BCNN_i with C_i channels, i.e., (1) can be rewritten as

$$y^{(i)} = \phi(\boldsymbol{x}; \{\delta^{(i)}, \varphi^{(i)}, \boldsymbol{W}_d^{(i)}, \boldsymbol{\theta}_d^{(i)}, \boldsymbol{W}_b^{(i)}, \boldsymbol{\theta}_b^{(i)}, \boldsymbol{W}_e^{(i)}, \boldsymbol{\theta}_e^{(i)}\}), \tag{16}$$

where $\boldsymbol{x}$ represents input data. After GAP, each channel of $y^{(i)}$ is transformed into a single-pixel neuron-like feature map, so that we can treat it as C_i neurons.

Given standard hypercube $\mathbf{I}^d = [0; 1]^d \in \mathbb{R}^d$ and any continuous function $f \in \mathrm{C}(\mathbf{I}^d)$, the proposed Stacked BCNN can be equivalently represented as

$$f_{\boldsymbol{p}_{k,u}} = \sum_{z=1}^{Z} w_z \sigma(\boldsymbol{x}; \{\phi, \delta, \varphi, \boldsymbol{W}^{(1)}, \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{W}^{(u)}, \boldsymbol{\theta}^{(u)}\}), \tag{17}$$

where $Z = \sum_{i=1}^{u} C_i$ is the neuron number of the GAP output, w represents the weight of the fully connected layer, $\boldsymbol{p}_{k,u} = (k, u, c, w_1, \ldots, w_Z, \boldsymbol{W}, \boldsymbol{\theta})$ represents the set of overall parameters for the Stacked BCNN, and σ is the activation function. Given the probability measure $\zeta_{k,u}$, we define randomly generated variables on $\boldsymbol{\xi}_{k,u} = (w_1, \ldots, w_Z, \boldsymbol{W}, \boldsymbol{\theta})$. For compact set Ω of $\mathbf{I}^d$, the distance between any continuous function and Stacked BCNN can be calculated as

$$\chi_{\Omega}(f, f_{\boldsymbol{p}_{k,u}}) = \sqrt{\mathbb{E}\left[\int_{\Omega} (f(\boldsymbol{x}) - f_{\boldsymbol{p}_{k,u}}(\boldsymbol{x}))^2 d\boldsymbol{x}\right]}. \tag{18}$$

Based on the above hypotheses, a theorem with proof of Stacked BCNN is given below.

Theorem 1: Given any continuous function $f \in \mathrm{C}(\mathbf{I}^d)$ and any compact set $\Omega \in \mathbf{I}^d$, Stacked BCNN with nonconstant bounded functions ϕ, δ, φ, and absolutely integrable activation function σ whose definition domain is $\mathbf{I}^d$ so that $\int_{\mathbb{R}_d} \sigma^2(\boldsymbol{x}) d\boldsymbol{x} < \infty$, has a sequence of $\{f_{\boldsymbol{p}_{k,u}}\}$ with probability measures $\zeta_{k,u}$ satisfied that

$$\lim_{u \to \infty} \chi_{\Omega}(f, f_{\boldsymbol{p}_{k,u}}) = 0. \tag{19}$$

Moreover, the trainable parameters $\boldsymbol{\xi}_{k,u}$ are generated by $\zeta_{k,u}$.

Proof: Define input data $\boldsymbol{x}$, nonconstant bounded functions ϕ, δ, φ, approximation function $f_{\boldsymbol{p}_{k,u'}}$ of Stacked BCNN with u' mini BCNNs, probability distribution $\zeta_{k,u'}$ for trainable parameter generation, the weight matrix of fully connected layer $\boldsymbol{w}' = [w'_1, \ldots, w'_{Z'}]^{\mathrm{T}}$ where $Z' = \sum_{z=1}^{u'} C_z$ and supplement weight $\boldsymbol{w}'' = [w''_1, \ldots, w''_{C_{u'+1}}]^{\mathrm{T}}$.

For Stacked BCNN with u' (any integer) mini BCNNs, we compute its output by

$$f_{\boldsymbol{w}'} = \sum_{z=1}^{Z'} w'_z \sigma(\boldsymbol{x}; \{\phi, \delta, \varphi, \boldsymbol{W}^{(1)}, \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{W}^{(u')}, \boldsymbol{\theta}^{(u')}\}). \tag{20}$$

Subsequently, Stacked BCNN with input data $\boldsymbol{x}$ can approximate continuous function f with bounded and integrable resident function $f_{r_{u'}} \in \mathbf{I}^d$ as

$$f_{r_{u'}}(\boldsymbol{x}) = f(\boldsymbol{x}) - f_{\boldsymbol{w}'}(\boldsymbol{x}). \tag{21}$$

As described in previous work [46], for $\forall \varepsilon > 0$, a function $f_{b_{u'}} \in \mathrm{C}(\mathbf{I}^d)$ can always be found to satisfy the following expression:

$$\chi_{\Omega}(f_{b_{u'}}, f_{r_{u'}}) < \frac{\varepsilon}{2}. \tag{22}$$

We define an extra mini BCNN (i.e., mini $\mathrm{BCNN}_{u'+1}$) to approximate $f_{b_{u'}}$ with $C_{u'+1}$ channels. Mini $\mathrm{BCNN}_{u'+1}$ can be equivalently expressed as

$$f_{\boldsymbol{w}''} = \sum_{z=1}^{C_{u'+1}} w''_z \underbrace{\sigma(\boldsymbol{x}; \{\phi, \delta, \varphi, \boldsymbol{W}^{(u'+1)}, \boldsymbol{\theta}^{(u'+1)}\})}_{\vartheta}, \tag{23}$$

Similarly, we can conclude that the composition function ϑ in (23) is absolutely integrable. According to *Theorem 1* in [47], for $\forall \varepsilon > 0$, a sequence of $f_{\boldsymbol{w}''}$ can be found to satisfy the following expression:

$$\chi_{\Omega}(f_{b_{u'}}, f_{\boldsymbol{w}''}) < \frac{\varepsilon}{2}. \tag{24}$$

Moreover, the output of Stacked BCNN can be rewritten as

$$f_{\boldsymbol{p}_{k,u}} = f_{\boldsymbol{w}'} + f_{\boldsymbol{w}''}. \tag{25}$$

Above all, we can obtain the distance between f and $f_{\boldsymbol{p}_{k,u}}$ by

$$\begin{aligned}
\chi_{\mathbf{I}^d}(f, f_{\boldsymbol{p}_{k,u}}) &= \sqrt{\mathbb{E}\left[\int_{\Omega} (f(\boldsymbol{x}) - f_{\boldsymbol{p}_{k,u}}(\boldsymbol{x}))^2 d\boldsymbol{x}\right]} \\
&= \sqrt{\mathbb{E}\left[\int_{\Omega} \left((f(\boldsymbol{x}) - f_{\boldsymbol{w}'}(\boldsymbol{x})) - f_{\boldsymbol{w}''}(\boldsymbol{x})\right)^2 d\boldsymbol{x}\right]} \\
&= \sqrt{\mathbb{E}\left[\int_{\Omega} \left(f_{r_{u'}} - f_{\boldsymbol{w}''}\right)^2 d\boldsymbol{x}\right]} \\
&= \chi_{\Omega}(f_{r_{u'}}, f_{\boldsymbol{w}''}) \\
&\leq \chi_{\Omega}(f_{b_{u'}}, f_{r_{u'}}) + \chi_{\Omega}(f_{b_{u'}}, f_{\boldsymbol{w}''}) \\
&< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \\
&< \varepsilon
\end{aligned} \tag{26}$$

Therefore, we can conclude that

$$\lim_{u,v\to\infty} \chi_\Omega(f, f_{\boldsymbol{p}_{k,u}}) = 0. \tag{27}$$

In other words, the proposed Stacked BCNN can completely appropriate any continuous function.