SGAS: Sequential Greedy Architecture Search

Guohao Li*1, Guocheng Qian*1, Itzel C. Delgadillo*1, Matthias Müller2, Ali Thabet1, Bernard Ghanem1
1King Abdullah University of Science and Technology (KAUST), Saudi Arabia
2Intelligent Systems Lab, Intel Labs, Germany

Abstract

Architecture design has become a crucial component of successful deep learning. Recent progress in automatic neural architecture search (NAS) shows a lot of promise. However, discovered architectures often fail to generalize in the final evaluation. Architectures with a higher validation accuracy during the search phase may perform worse in the evaluation (see Figure 1). Aiming to alleviate this common issue, we introduce sequential greedy architecture search (SGAS), an efficient method for neural architecture search. By dividing the search procedure into sub-problems, SGAS chooses and prunes candidate operations in a greedy fashion. We apply SGAS to search architectures for Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN). Extensive experiments show that SGAS is able to find state-of-the-art architectures for tasks such as image classification, point cloud classification and node classification in protein-protein interaction graphs with minimal computational cost.

1. Introduction

Deep learning has revolutionized computer vision by learning features directly from data. As a result deep neural networks have achieved state-of-the-art results on many difficult tasks such as image classification [13], object detection [30], object tracking [37], semantic segmentation [11], depth estimation [15] and activity understanding [7], to name just a few examples. While there was a big emphasis on feature engineering before deep learning, the focus has now shifted to architecture engineering. In particular many novel architectures have been proposed, such as LeCun [26], AlexNet [25], VGG [44], GoogLeNet [46], ResNet [18], DenseNet [21], ResNeXt [54] and SENet [20]. Results on each of the above mentioned tasks keep improving every year by innovations in architecture design. In essence, the community has shifted from feature engineering to architecture engineering.

In recent years, many efforts have been made to reduce the manual intervention required to obtain better models for a particular task. As a matter of fact, a new area of research, commonly referred to as meta-learning, has emerged in order to tackle such problems. The idea of meta-learning is to leverage prior experience in order to quickly find good algorithm configurations, network architectures and any required parameters for a new learning task. Examples of recent meta-learning approaches include automatic hyper-parameter search [14], data-augmentation [12], finding novel optimizers [2] and architecture search [62]. In particular, architecture search has sparked a lot of interest in the community. In this task, the search space is huge and manual search is prohibitive.

Early work by Zoph et al. [62], based on Reinforcement Learning, has shown very promising results. However, its high computational cost has prevented widespread adoption. Recently, differentiable architecture search (DARTS) [33] has been proposed as an alternative which makes architecture search differentiable and much more efficient. This has opened up a path towards computationally feasible ar-

*equal contribution
architecture search. However, despite their success, current approaches still have a lot of limitations. During the search phase, network architectures are usually constructed from basic building blocks and evaluated on a validation set. Due to computational cost, the size of considered architectures is limited. In the evaluation phase, the best building blocks are used to construct larger architectures and they are evaluated on the test set. As a result there is a large discrepancy between the validation accuracy during search and the test accuracy during evaluation. In this work, we propose a novel greedy architecture search algorithm, SGAS, which addresses this discrepancy and searches very efficiently.

**Contributions.** Our contributions can be summarized as the following: (1) We propose SGAS, a greedy approach for neural architecture search with high correlation between the validation accuracy during the search phase and the final evaluation accuracy. (2) Our method discovers top-performing architectures with much less search cost than previous state-of-the-art methods such as DARTS. (3) Our proposed method is able to search architectures for both CNNs and GCNs across various datasets and tasks.

2. Related Work

In the past, considerable success was achieved with hand-crafted architectures. One of the earliest successful architectures was LeNet [26], a very simple convolutional neural network for optical character recognition. Other prominent networks include AlexNet [25], VGG [44] and GoogLeNet [46] which revolutionized computer vision by outperforming all previous approaches in the ImageNet [13] challenge by a large margin. ResNet [18] and DenseNet [21] were further milestones in architecture design. They showed the importance of residual and dense connections for designing very deep networks, an insight that influences modern architecture design to this day. Until recently, architecture innovations were a result of human insight and experimentation. The first successful attempts for architecture search were using reinforcement learning [62] and evolutionary algorithms [40]. These works were extended with NASNet [63] where a new cell-based search space and regularization technique were proposed. Another extension, ENAS [38], represents the entire search space as a single directed acyclic graph. A controller discovers architectures by searching for subgraphs that maximize the expected reward on the validation set. This setup allows for parameter sharing between child models making search very efficient. Further, PNAS [31] introduced a sequential model-based optimization (SMBO) strategy in order to search for structures of increasing complexity. PNAS needs to evaluate 5 times less models and reduces the computational cost by a factor of 8 compared to NASNet. Yet, PNAS still requires thousands of GPU hours. One shot approaches [6, 5, 8] further reduce the search time by training a single over-parameterized network with inherited/shared weights. In order to search in a continuous domain [41, 1, 43, 50], DARTS [33] proposes a continuous relaxation of the architecture representation, making architecture search differentiable and hence much more efficient. As a result, DARTS is able to find good convolutional architectures at a fraction of the computational cost making NAS broadly accessible. Owing to the large success of DARTS, several extensions have been proposed recently. SNAS [55] optimizes parameters of a joint distribution for the search space in a cell. The authors propose a search gradient which optimizes the same objective as RL-based NAS, but leads to more efficient structural decisions. P-DARTS [9] attempts to overcome the depth gap issue between search and evaluation. This is accomplished by increasing the depth of searched architectures gradually during the training procedure. PCDARTS [58] leverages the redundancy in network space and only samples a subset of channels in super-net during search to reduce computation.

3. Methodology

3.1. Preliminary - DARTS

By reducing the search problem to searching for the best cell structure, cell-based NAS methods [53, 31, 40] are able to learn scalable and transferable architectures. The networks are composed of layers with identical cell structure but different weights. A cell is usually represented as a directed acyclic graph (DAG) with $N$ nodes including two input nodes, several intermediate nodes and a single output node. Each node is a latent representation denoted as $x^{(i)}$, where $i$ is its topological order in the DAG. Each directed edge $(i, j)$ in the DAG is associated with an operation $o^{(i, j)}$ that transfers the information from node $x^{(i)}$ to node $x^{(j)}$. In Differentiable Architecture Search (DARTS) [33] and its variants [55, 5, 38], the optimal architecture is derived from a discrete search space by relaxing the selection of operations to a continuous optimization problem. During the search phase, the operation of each edge is parameterized by architectural parameters $\alpha^{(i, j)}$ as a softmax mixture over all the possible operations within the operation set $O$, i.e., $\tilde{o}^{(i, j)}(x^{(i)}) = \frac{\exp(\alpha^{(i, j)}_{o}))}{\sum_{o' \in O} \exp(\alpha^{(i, j)}_{o'}))} \cdot o(x^{(i)})$. The input nodes are represented by the outputs from the previous two cells. Each intermediate node aggregates information flows from all of its predecessors, $x^{(j)} = \sum_{i < j} o^{(i, j)}(x^{(i)})$. The output node is defined as a concatenation of a fixed number of its predecessors. The learning procedure of architectural parameters involves a bi-level optimization problem:

$$\min_{A} \mathcal{L}_{val}(\mathcal{W}^*(A), A)$$

(1)

s.t. $\mathcal{W}^*(A) = \text{argmin}_{\mathcal{W}} \mathcal{L}_{train}(\mathcal{W}, A)$

(2)
The Kendall $\tau$ is a widely used metric to determine the correlation. The corresponding architectural parameter $\alpha^{(i,j)}$ will be removed from the bi-level optimization. Operations which were not chosen in a mixture operation will be pruned. At the end of the search phase, a stand-alone architecture without weight sharing will be obtained.

$\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ denote the training and validation loss respectively. Owing to the continuous relaxation, the search is realized by optimizing a supernet. $W$ is the set of weights of the supernet and $O$ is the set of the architectural parameters. DARTS [33] proposed to solve this bi-level problem by a first/second order approximation. At the end of the search, the final architecture is derived by selecting the operation with highest weight for every mixture operation, $o^{(i,j)} = \arg\max_{o \in O} \alpha^{(i,j)}$.

### 3.2. Search-Evaluation Correlation

A popular pipeline of existing NAS algorithms [63, 33] includes two stages: a search phase and an evaluation phase. In order to reduce computational overhead, previous works [63, 33] first search over a pre-defined search space with a lightweight proxy model on a small proxy dataset. After the best architecture cell/encoding is obtained, the final architecture is built and trained from scratch on the target dataset. This requires that the true performance during evaluation can be inferred during the search phase. However, this assumption usually does not hold due to the discrepancy in dataset, hyper-parameters and network architectures between the search and evaluation phases. The best ranking derived from the search phase does not imply the actual ranking in the final evaluation. In practice, the correlation between the performances of derived architectures during the search and evaluation phases is usually low. In this paper, we refer to this issue as **degenerate search-evaluation correlation**. Recent work by Sciuto et al. [42] also analyzes this issue and suggests that the Kendall $\tau$ metric [22] could be used to evaluate the search phase. They show that the widely used weight sharing technique actually decreases the correlation. The Kendall $\tau$ metric [22] is a common measurement of the correlation between two rankings. The Kendall $\tau$ coefficient can be computed as $\tau = \frac{N_c - N_d}{\frac{1}{2}n(n-1)}$, where $N_c$ and $N_d$ are the number of concordant pairs and the number of discordant pairs respectively. It is a number in the range from $-1$ to $1$ where $-1$ corresponds to a perfect negative correlation and $1$ to a perfect positive correlation. If the Kendall $\tau$ coefficient is $0$, the rankings are completely independent. An ideal NAS method should have a high **search-evaluation Kendall $\tau$ coefficient**. We take DARTS [33] as an example and show its Kendall $\tau$ in Figure 1. It is calculated between the rankings during search phase and evaluation phase. The rankings are determined according to the validation accuracy and the final evaluation accuracy after 10 different runs on the CIFAR-10 dataset. The Kendall $\tau$ coefficients for DARTS are only $0.16$ and $-0.29$ for the 1st-order and 2nd-order versions respectively. Therefore, it is impossible to make reliable predictions regarding the final test accuracy based on the search phase.

### 3.3. Sequential Greedy Architecture Search

In order to alleviate the degenerate search-evaluation correlation problem, the core aspects are to reduce (1) the discrepancy between the search and evaluation phases and (2) the negative effect of weight sharing. We propose to solve the bi-level optimization (Equation 1, 2) in a sequential greedy fashion to reduce the model discrepancy and the weight sharing progressively. As mentioned in Section 3.1, DARTS-based methods [33, 9, 58] solve the relaxed problem fully and obtain all the selected operations at the end. Instead of solving the complete problem directly, we divide it into sub-problems and solve them sequentially with a greedy algorithm. The sub-problems are defined based on the directed edges in the DAG. We pick the operation

![Figure 2. Illustration of Sequential Greedy Architecture Search.](image_url)
for edges greedily in a sequential manner and solve the remaining sub-problem iteratively. The iterative procedure is shown in Algorithm 1. At each decision epoch, we choose one edge \((i^*, j^*)\) according to a pre-defined selection criterion. A greedy optimal choice is made for the selected edge by replacing the corresponding mixture operation \(\alpha_{i,j^*}^\dagger\) with \(\alpha_{i,j^*}^\dagger = \arg\max_{o \in O} \alpha_{i,j^*}^\dagger o\). The architectural parameters \(\alpha_{i,j^*}^\dagger\) and the weights of the remaining paths within the mixture operations are no longer needed; we prune and exclude them from the latter optimization. As a side benefit, the efficiency improves as parameters in \(A\) and \(W\) are pruned gradually in the optimization loop. The search procedure of the remaining \(A\) and \(W\) forms a new sub-problem which will be solved iteratively. At the end of the search phase, a stand-alone network without weight sharing is obtained, as illustrated in Figure 2. Therefore, the model discrepancy is minimized and the validation accuracy during the search phase reflects the final evaluation accuracy much better. To maintain the optimality, the design of the selection criterion is crucial. We consider three aspects of edges which are the edge importance, the selection certainty and the selection stability.

**Edge Importance.** Similar to DARTS [33], a zero operation is included in the search space to indicate a lack of connection. Edges that are important should have a low weight in the zero operation. Thus, the edge importance is defined as the summation of weights over non-zero operations:

\[
S_{EI}^{(i,j)} = \sum_{o \in O, o \neq \text{zero}} \frac{\exp(\alpha_{o}^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})} \tag{3}
\]

**Selection Certainty.** Entropy is a common measurement of uncertainty of a distribution. The normalized softmax weights of non-zero operations can be regarded as a distribution, \(p_{o}^{(i,j)} = \frac{\exp(\alpha_{o}^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})}, o \in O, o \neq \text{zero}.\) We define the selection certainty as the complement of the normalized entropy of the operation distribution:

\[
S_{SC}^{(i,j)} = 1 - \frac{-\sum_{o \in O, o \neq \text{zero}} p_{o}^{(i,j)} \log(p_{o}^{(i,j)})}{\log(|O| - 1)} \tag{4}
\]

**Selection Stability.** In order to incorporate the history information, we measure the movement of the operation distribution. Kullback–Leibler divergence and histogram intersection [45] are two popular methods to detect changes in distribution. For simplicity, we choose the latter one. The average selection stability at step \(T\) with a history window size \(K\) is computed as follows:

\[
S_{SS}^{(i,j)} = \frac{1}{K} \sum_{t=T-K}^{T-1} \sum_{o \in \Omega} \min(p_{ot}^{(i,j)}, p_{oT}^{(i,j)}) \tag{5}
\]

In our experiments, we consider the following two criteria:

**Criterion 1.** An edge \((i^1, j^1)\) with a high edge importance \(S_{EI}^{(i,j)}\) and a high selection certainty \(S_{SC}^{(i,j)}\) will be selected. We normalize \(S_{EI}^{(i,j)}\) and \(S_{SC}^{(i,j)}\), compute the final score and pick the edge with the highest score:

\[
S_{1}^{(i,j)} = \text{normalize}(S_{EI}^{(i,j)}) \ast \text{normalize}(S_{SC}^{(i,j)}) \tag{6}
\]

**Criterion 2.** In addition to Criterion 1, we also consider that the selected edge \((i^1, j^1)\) should have a high selection stability. The final score is defined as follows:

\[
S_{2}^{(i,j)} = S_{1}^{(i,j)} \ast \text{normalize}(S_{SS}^{(i,j)}) \tag{7}
\]

where \(\text{normalize}(\cdot)\) denotes a standard Min-Max scaling normalization. For a fair comparison with existing works [62, 40, 33], two incoming edges are preserved for every intermediate node in the DAG. Once a node has two determined incoming edges, its other incoming edges will be pruned. We refer to our method as **Sequential Greedy Architecture Search** (SGAS). Figure 1 shows that SGAS with Criterion 1 and 2 improves the Kendall \(\tau\) correlation coefficients to 0.56 and 0.42 respectively. As expected from the much higher search-evaluation correlation SGAS outperform DARTS in terms of average accuracy significantly.
4. Experiments

We use our SGAS to automatically find architectures for both CNNs and GCNs. The CNN architectures discovered by SGAS outperform the state-of-the-art (SOTA) in image classification on CIFAR-10 [24] and ImageNet [13]. Similarly, the discovered GCN architectures outperform the state-of-the-art methods for point cloud classification on ModelNet [53] and node classification in biological graphs using the PPI [61] dataset.

4.1. Searching CNN architectures with SGAS

4.1.1 Architecture Search on CIFAR-10

As is common practice, we first search for normal cells and reduction cells with a small network for image classification on CIFAR-10. CIFAR-10 is a small popular dataset containing $50K$ training images and $10K$ testing images. Then, a larger network is constructed by making necessary changes in channel size and stacking the searched cells multiple times. The larger network is retrained on CIFAR-10 to compare its performance with other state-of-the-art methods. Finally, we show the transferability of our SGAS by stacking even more cells and evaluating on ImageNet. We show that SGAS consistently achieves the top performance.

Search Space. We keep our search space the same as DARTS, which has 8 candidate operations: skip-connect, max-pool-3×3, avg-pool-3×3, sep-conv-3×3, sep-conv-5×5, dil-conv-3×3, dil-conv-5×5, zero. During the search phase, we stack 6 normal cells and 2 reduction cells to form a network. Two reduction cells are inserted at a network depth of 1/3 and 2/3 respectively. The stride of each convolution in normal cells is 1, so the spatial size of an input feature map does not change. In reduction cells, convolutions with stride 2 are used to reduce the spatial resolution of feature maps. There are 7 nodes with 4 intermediate nodes and 14 edges in each cell during search. The first and second input nodes of the cell are set equal to the outputs of the two previous cells respectively. The output node of a cell is the depth-wise concatenation of all the intermediate nodes.

Training Settings. We keep the training setting the same as in DARTS [33]. A small network consisting of 6 normal cells and 2 reduction cells with an initial channel size 16 is trained on CIFAR-10. We perform architecture search for 50 epochs with a batch size of 64. SGD is used to optimize the model weights $\mathcal{W}$ with an initial learning rate 0.025, momentum 0.9 and weight decay $3 \times 10^{-4}$. For architecture parameters $\mathcal{A}$, the Adam optimizer with an initial learning rate $3 \times 10^{-4}$, momentum (0.5, 0.999) and weight decay $10^{-3}$ is used. Instead of training the entire super-net throughout the search phase, SGAS makes decisions sequentially in a greedy fashion. After warming up for 9 epochs, SGAS begins to select one operation for one selected edge every 5 epochs using Criterion 1 or Criterion 2 as the selection criterion. For Criterion 2, we set the history window size $K$ to 4. The batch size is increased by 8 after each greedy decision, which further boosts the searching efficiency of SGAS. We provide a thorough discussion and ablation study on the choices of hyper-parameters in the supplementary material. The search takes only 0.25 day (6 hours) on a single NVIDIA GTX 1080Ti.

4.1.2 Architecture Evaluation on CIFAR-10

We run 10 independent searches to get 10 architectures with Criterion 1 or Criterion 2, as shown in Figure 1. To highlight the stability of the search method, we evaluate the discovered architectures on CIFAR-10 and report the mean and standard deviation of the test accuracy across those 10 models and the performance of the best model in Table 1. It is important to mention that other related works in Table 1 only report the mean and standard deviation for their best architecture with different runs on evaluation.

Training Settings. We train a large network of 20 cells with a initial channel size 36. The SGD optimizer is used during 600 epochs with a batch size of 96. The other hyper-parameters remain the same as the search phase. Cutout with length 16, auxiliary towers with weight 0.4 and path dropout with probability 0.3 are used as in DARTS [33].

Evaluation Results and Analysis. We compare our results with other methods in Table 1 and report the average and best performance for both Criterion 1 and Criterion 2. We outperform our baseline DARTS by a significant margin with test errors of 2.39% and 2.44% respectively while only using 0.25 day (6 hours) on a single NVIDIA GTX 1080Ti.

4.1.3 Architecture Evaluation on ImageNet

The architecture evaluation on ImageNet uses the cell architectures that we obtained after searching on CIFAR-10.

Training Settings. We choose the 3 best performing cell architectures on CIFAR-10 for each Criterion and train them on ImageNet. For this evaluation, we build a large network with 14 cells and 48 initial channels and train for 250 epochs with a batch size of 1024. The SGD optimizer with an initial learning rate of 0.5, a momentum of 0.9 and a weight decay of $3 \times 10^{-5}$ is used. We run these experiments on 8 Nvidia Tesla V100 GPUs for three days.

Evaluation Results and Analysis. In Table 2 we compare our models with SOTA hand-crafted architectures (manual) and models obtained through other search methods. We apply the mobile setting for ImageNet, which has an image size of $224 \times 224$ and restricts the number of multi-add operations to 600M. Our best performing models SGAS (Cri.1 best) and SGAS (Cri.2 best) outperform all the other methods with top-1 errors 24.2% and 24.1% respectively.
while using only a search cost of 0.25 GPU day on one NVIDIA GTX 1080Ti. SGAS (Cri.2) outperforms SGAS (Cri.1) showing the effectiveness of integrating selection stability into the selection criterion. The best performing cells of SGAS (Cri.2 best) are illustrated in Figure 3.

4.2. Searching GCN architectures with SGAS

Recently, GCNs have achieved impressive performance on point cloud segmentation [28], biological graph node classification [27] and video recognition [57] by training DeepGCNs [28, 27]. However, this hand-crafted architecture design requires adequate effort by an human expert. The main component of DeepGCNs is the GCN backbone. We explore an automatic way to design the GCN backbone using SGAS. Our backbone network is formed by stacking the graph convolutional cell discovered by SGAS. Our GCN cell consists of 6 nodes. We use fixed $1 \times 1$ convolutions in the first two nodes, and set the input to them equal to the output from the previous two layers. Our experiments on GCNs have two stages. First, we apply SGAS to search for the graph convolutional cell using a small dataset and obtain 10 architectures from 10 runs. Then, 10 larger networks are constructed by stacking each discovered cell multiple times. The larger networks are trained on the same dataset or a larger one to evaluate their performance. We report the best and average performance of these 10 architectures. We show the effectiveness of SGAS in GCN architecture search by comparisons with SOTA hand-crafted methods and Random Search.

### 4.2.1 Architecture Search on ModelNet10

ModelNet [53] is a dataset for 3D object classification with two variants, ModelNet10 and ModelNet40 containing objects from 10 and 40 classes respectively. We conduct GCN architecture search on ModelNet10 and then evaluate the final performance on ModelNet40.

**Search Space.** Our graph convolutional cell has 10 candidate operations: $conv-1 \times 1$, MRCov [28], EdgeConv [52], GAT [49], SemeGCN [23], GIN [56], SAGE [16], RelSAGE, skip-connection, and zero operation. Please refer to our supplementary material for more details of these GCN operators. We use $k$ nearest neighbor in the first operation of each cell to construct edges (we use $k = 9$ by default unless it is specified). These edges are then shared in the following operations inside the cell. Dilated graph convolutions with the same linearly increasing dilation rate schedule as proposed in DeepGCNs [28] are applied to the cells.

**Training Settings.** We sample 1024 points from the 3D models in ModelNet10. We use 2 cells with 32 initial channels and search the architectures for 50 epochs with batch size 28. SGD is used to optimize the model weights with initial learning rate 0.005, momentum 0.9 and weight decay $3 \times 10^{-4}$. The Adam optimizer with the same parameters as in the search for CNNs is used to optimize architecture parameters. After warming up for 9 epochs, SGAS begins to select one operation for a selected edge every 7 epochs. We experimented with both selection criteria, Criterion 1 and Criterion 2. We use a history window of 4 for Cri.2. The

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Test Err. (%)</th>
<th>Params (M)</th>
<th>Search Cost (GPU-days)</th>
<th>Search Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>DenseNet-BC [21]</td>
<td>3.46</td>
<td>25.6</td>
<td>-</td>
<td>manual</td>
</tr>
<tr>
<td>NASNet-A [63]</td>
<td>2.65</td>
<td>3.3</td>
<td>1800</td>
<td>RL</td>
</tr>
<tr>
<td>AmoebaNet-A [40]</td>
<td>3.34±0.06</td>
<td>3.2</td>
<td>3150</td>
<td>evolution</td>
</tr>
<tr>
<td>AmoebaNet-B [40]</td>
<td>2.55±0.05</td>
<td>2.8</td>
<td>3150</td>
<td>evolution</td>
</tr>
<tr>
<td>Hier-Evolution [32]</td>
<td>3.75±0.12</td>
<td>15.7</td>
<td>300</td>
<td>evolution</td>
</tr>
<tr>
<td>PNAS [31]</td>
<td>3.41±0.09</td>
<td>3.2</td>
<td>225</td>
<td>SMBO</td>
</tr>
<tr>
<td>ENAS [38]</td>
<td>2.89</td>
<td>4.6</td>
<td>0.5</td>
<td>RL</td>
</tr>
<tr>
<td>NAO-Net-WS [35]</td>
<td>3.53</td>
<td>3.1</td>
<td>0.4</td>
<td>NAO</td>
</tr>
</tbody>
</table>

Table 1. Performance comparison with state-of-the-art image classifiers on CIFAR-10. We report the average and best performance of SGAS (Cri.1) and SGAS (Cri.2). Criterion 1 and Criterion 2 are used in the search respectively. *Note that mean and standard derivation are computed across 10 independently searched architectures.
shows that SGAS (Cri.2 best), the best architecture is evaluated with the overall accuracy (OA). We also form a large backbone network for each and train them on ModelNet40. The performance of 3D point cloud classification is evaluated with the overall accuracy (OA). We sample 1024 points as input. Our architectures are all trained for 30 epochs with batch size of 400.

Training Settings. We stack the searched cell 9 times with channel size 128. We also form small networks by stacking the cell 3 times with the same channel size. We use $k = 20$ for all the large networks and $k = 9$ for the small ones. Adam is used to optimize the weights with initial learning rate $0.001$ and weight decay $1 \times 10^{-4}$. We sample 1024 points as input. Our architectures are all trained for 400 epochs with batch size of 32. We report the mean and standard deviation of the accuracy on the test dataset of the 10 discovered architectures; we also report the accuracy of the best performing model of the big and the small networks.

4.2.2 Architecture Evaluation on ModelNet40

After searching for 10 architectures on ModelNet10, we form a large backbone network for each and train them on ModelNet40. The performance of 3D point cloud classification is evaluated with the overall accuracy (OA). We also apply Random Search to the same search space to obtain 10 architectures as our random search baseline.

Training Settings. We stack the searched cell 9 times with channel size 128. We also form small networks by stacking the cell 3 times with the same channel size. We use $k = 20$ for all the large networks and $k = 9$ for the small ones. Adam is used to optimize the weights with initial learning rate $0.001$ and weight decay $1 \times 10^{-4}$. We sample 1024 points as input. Our architectures are all trained for 400 epochs with batch size of 32. We report the mean and standard deviation of the accuracy on the test dataset of the 10 discovered architectures; we also report the accuracy of the best performing model of the big and the small networks.

Evaluation Results and Analysis. We compare the performance of our discovered architectures with SOTA hand-crafted methods and architectures obtained by Random Search for 3D point clouds classification on ModelNet40. Table 3 shows that SGAS (Cri.2 best), the best architecture discovered by our SGAS with Criterion 2, outperforms all the other models. The smaller network SGAS (Cri.2 small best) discovered by SGAS with Criterion 2 also outperforms all the hand-crafted architectures. Owing to a well-designed search space, Random Search is a strong baseline. The performance of SGAS surpasses the hand-crafted architectures and Random Search, demonstrating the effectiveness of SGAS for GCN architecture search. The best architecture for this task can be found in Figure 4 (a).

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Test Err. (%)</th>
<th>Params (M)</th>
<th>Search Cost (GPU-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception-v1 [46]</td>
<td>30.2</td>
<td>10.1</td>
<td>6.6</td>
</tr>
<tr>
<td>MobileNet [19]</td>
<td>29.4</td>
<td>10.5</td>
<td>4.2</td>
</tr>
<tr>
<td>ShuffleNet 2x (v1) [59]</td>
<td>26.4</td>
<td>10.2</td>
<td>~5</td>
</tr>
<tr>
<td>ShuffleNet 2x (v2) [36]</td>
<td>25.1</td>
<td>-</td>
<td>~5</td>
</tr>
<tr>
<td>NASNet-A [63]</td>
<td>26</td>
<td>8.4</td>
<td>5.3</td>
</tr>
<tr>
<td>NASNet-B [63]</td>
<td>27.2</td>
<td>8.7</td>
<td>5.3</td>
</tr>
<tr>
<td>NASNet-C [63]</td>
<td>27.5</td>
<td>9</td>
<td>4.9</td>
</tr>
<tr>
<td>AmoebaNet-A [40]</td>
<td>25.5</td>
<td>8</td>
<td>5.1</td>
</tr>
<tr>
<td>AmoebaNet-B [40]</td>
<td>26</td>
<td>8.5</td>
<td>5.3</td>
</tr>
<tr>
<td>AmoebaNet-C [40]</td>
<td>24.3</td>
<td>7.6</td>
<td>6.4</td>
</tr>
<tr>
<td>FairNAS-A [10]</td>
<td>24.7</td>
<td>7.6</td>
<td>4.6</td>
</tr>
<tr>
<td>PNAS [31]</td>
<td>25.8</td>
<td>8.1</td>
<td>5.1</td>
</tr>
<tr>
<td>MnasNet-92 [47]</td>
<td>25.2</td>
<td>8</td>
<td>4.4</td>
</tr>
<tr>
<td>DARTS (2nd order) [33]</td>
<td>26.7</td>
<td>8.7</td>
<td>4.7</td>
</tr>
<tr>
<td>SNAS (mild) [55]</td>
<td>27.3</td>
<td>9.2</td>
<td>4.3</td>
</tr>
<tr>
<td>ProxylessNAS [8]</td>
<td>24.9</td>
<td>7.5</td>
<td>7.1</td>
</tr>
<tr>
<td>P-DARTS [9]</td>
<td>24.4</td>
<td>7.4</td>
<td>4.9</td>
</tr>
<tr>
<td>BayesNAS [60]</td>
<td>26.5</td>
<td>8.9</td>
<td>3.9</td>
</tr>
<tr>
<td>PC-DARTS [58]</td>
<td>25.1</td>
<td>7.8</td>
<td>5.3</td>
</tr>
<tr>
<td>SGAS (Cri.1 avg.)</td>
<td>24.41±0.16</td>
<td>7.29±0.09</td>
<td>5.3</td>
</tr>
<tr>
<td>SGAS (Cri.1 best)</td>
<td>24.2</td>
<td>7.2</td>
<td>5.3</td>
</tr>
<tr>
<td>SGAS (Cri.2 avg.)</td>
<td>24.38±0.22</td>
<td>7.39±0.07</td>
<td>5.4</td>
</tr>
<tr>
<td>SGAS (Cri.2 best)</td>
<td>24.1</td>
<td>7.3</td>
<td>5.4</td>
</tr>
</tbody>
</table>

Table 2. Comparison with state-of-the-art classifiers on ImageNet. We transfer the top 3 performing architectures on CIFAR-10 to ImageNet in the mobile setting. × + denote multiply-add operations. The average and best performance of SGAS are reported.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>OA (%)</th>
<th>Params (M)</th>
<th>Search Cost (GPU-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DmFV-Net [4]</td>
<td>91.6</td>
<td>45.77</td>
<td>manual</td>
</tr>
<tr>
<td>SpecGCN [51]</td>
<td>91.5</td>
<td>2.05</td>
<td>manual</td>
</tr>
<tr>
<td>PointNet++ [39]</td>
<td>90.7</td>
<td>1.48</td>
<td>manual</td>
</tr>
<tr>
<td>PCNN [3]</td>
<td>92.3</td>
<td>8.2</td>
<td>manual</td>
</tr>
<tr>
<td>PointCNN [29]</td>
<td>92.2</td>
<td>0.6</td>
<td>manual</td>
</tr>
<tr>
<td>DGCNN [52]</td>
<td>92.2</td>
<td>1.84</td>
<td>manual</td>
</tr>
<tr>
<td>KPConv [48]</td>
<td>92.9</td>
<td>14.3</td>
<td>manual</td>
</tr>
<tr>
<td>Random Search</td>
<td>92.65±0.33</td>
<td>8.77</td>
<td>random</td>
</tr>
<tr>
<td>SGAS (Cri.1 avg.)</td>
<td>92.69±0.20</td>
<td>8.78</td>
<td>0.19</td>
</tr>
<tr>
<td>SGAS (Cri.1 best)</td>
<td>92.87</td>
<td>8.63</td>
<td>0.19</td>
</tr>
<tr>
<td>SGAS (Cri.2 avg.)</td>
<td>92.93±0.19</td>
<td>8.87</td>
<td>0.19</td>
</tr>
<tr>
<td>SGAS (Cri.2 best)</td>
<td><strong>93.23</strong></td>
<td>8.49</td>
<td>0.19</td>
</tr>
<tr>
<td>SGAS (Cri.2 small best)</td>
<td>93.07</td>
<td>3.86</td>
<td>0.19</td>
</tr>
</tbody>
</table>

Table 3. Comparison with state-of-the-art architectures for 3D object classification on ModelNet40. 10 architectures are derived for both SGAS and Random Search within the same search space.

batch size increases by 4 after each decision. The search takes around 0.19 GPU day on one NVIDIA GTX 1080Ti.
4.2.3 Architecture Search on PPI

PPI is a popular biological graph dataset in the data mining domain. We search for GCN architectures on the PPI dataset for the task of node classification.

Training Settings. We use 1 cell with 32 channels. We train and search the architectures for 50 epochs with a batch size of 6 on PPI. We do not increase the batch size after making decisions since PPI is small and only contains 20 batches. The other parameters are the same as when searching on ModelNet10. The search takes around 0.003 day (4 minutes) on a Nvidia Tesla V100 GPU (16GB).

4.2.4 Architecture Evaluation on PPI

We evaluate architectures on the PPI test set. We report the mean, standard derivation and the best accuracy and compare them with the SOTA methods and Random Search. We also conduct an ablation study on number of cells and channel size which we include in the supplementary material.

Training Settings. We stack the discovered cell 5 times with channel size 512. Adam is used to optimize the model weights with initial learning rate 0.002. We use a cosine annealing to schedule the learning rate. Our architectures are trained for 2000 epochs with batch size of 1 as suggested in DeepGCNs [27]. We find the best model on the validation dataset and obtain the micro-F1 score on the test dataset.

Evaluation Results and Analysis. We compare the average and best performance of SGAS to other state-of-the-arts methods and Random Search on node classification on the PPI dataset. Table 4 shows the best architecture discovered by our SGAS outperforms the state-of-the-art DenseMRGCN-14 [27] by ~0.03% with ~30.24 M less parameters. The average performance of SGAS also surpasses the Random Search baseline consistently. In addition, SGAS (Cri.2 avg.) outperforms SGAS (Cri.1 avg.) in terms of both mean and standard deviation. This indicates that Criterion 2 provides more stable results. We visualize the architecture with the best performance in Figure 4 (b).

5. Conclusion

In this work, we propose the Sequential Greedy Architectural Search (SGAS) algorithm to design architectures automatically for CNNs and GCNs. The bi-level optimization problem in NAS is solved in a greedy fashion using heuristic criteria which take the edge importance, the selection certainty and the selection stability into consideration. Such an approach alleviates the effect of the degenerate search-evaluation correlation problem and reflects the true ranking of architectures. As a result, architectures discovered by SGAS achieve state-of-the-art performance on CIFAR-10, ImageNet, ModelNet and PPI datasets.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) through VCC funding. The authors thank the KAUST IBEX team for helping to optimize the training workloads on ImageNet.
References


[10] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search, 2019. 7


Inter-
