# AGNAS: Attention-Guided Micro- and Macro-Architecture Search

**Zihao Sun** [1 2 3] **Yu Hu** [1 2 3] **Shun Lu** [1 2 3] **Longxing Yang** [1 2 3] **Jilin Mei** [1 2 3] **Yinhe Han** [1 2 3] **Xiaowei Li** [2 3]

## Abstract

Micro- and macro-architecture search have emerged as two popular NAS paradigms recently. Existing methods leverage different search strategies for searching micro- and macro- architectures. When using architecture parameters to search for micro-structure such as normal cell and reduction cell, the architecture parameters can not fully reflect the corresponding operation importance. When searching for the macro-structure chained by pre-defined blocks, many sub-networks need to be sampled for evaluation, which is very time-consuming. To address the two issues, we propose a new search paradigm, that is, leverage the attention mechanism to guide the micro- and macro-architecture search, namely AGNAS. Specifically, we introduce an attention module and plug it behind each candidate operation or each candidate block. We utilize the attention weights to represent the importance of the relevant operations for the micro search or the importance of the relevant blocks for the macro search. Experimental results show that AGNAS can achieve 2.46% test error on CIFAR-10 in the DARTS search space, and 23.4% test error when directly searching on ImageNet in the ProxylessNAS search space. AGNAS also achieves optimal performance on NAS-Bench-201, outperforming state-of-the-art approaches. The source code can be available at https://github.com/Sunzh1996/AGNAS.

[1]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China [2]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China [3]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China. Correspondence to: Yu Hu <huyu@ict.ac.cn>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. 

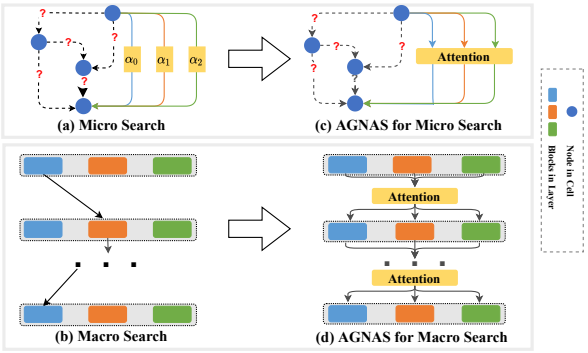



*Figure 1.* Comparison of Micro search or Macro search with our AGNAS. Micro search assigns architecture parameters $\alpha$ for candidate operations to search for the cell, and Macro search samples numerous single choice paths for evaluating. Instead, we insert the attention module to the output of each edge in a cell or to the output of each layer, with the aim that the attention weights can represent the importance of associated operations or choice blocks.

## 1. Introduction

Neural Architecture Search (NAS) has attracted lots of attention in recent years, because it can automatically find optimal neural networks in the pre-defined search space for target tasks. More importantly, the searched architecture can perform better than hand-crafted neural networks in many computer vision tasks, such as image classification (Zoph & Le, 2016; Zoph et al., 2018; Guo et al., 2020), object detection (Chen et al., 2019; Ghiasi et al., 2019), semantic segmentation (Chen et al., 2018; Liu et al., 2019). Early NAS methods adopted reinforcement learning (RL) (Baker et al., 2016; Bello et al., 2017; Zoph et al., 2018) or evolutionary algorithms (EA) (Real et al., 2017; Liu et al., 2018b; Real et al., 2019), requiring to train many sub-networks from scratch, which takes thousands or even tens of thousands of GPU-hours. To alleviate this issue, ENAS (Pham et al., 2018) proposed weight sharing among child architectures in a pre-defined search space, which dramatically reduced the search cost to a few GPU-days.

Search space plays a vital role in NAS, and can be broadly categorized into micro search space and macro search space. Specifically, as shown in Figure 1 (a), the micro search, which is also referred to as operation search, aims to determine the operation associated with each pair of nodes in a

cell and then manually stack a series of identical cells to build a target neural network. DARTS (Liu et al., 2018c) relaxed the discrete search space to be continuous, and used a bi-level optimization to alternately optimize the architecture parameters and network weights, thus achieving an efficient end-to-end training. Thereafter, recent works (Xie et al., 2018; Dong & Yang, 2019; Li et al., 2020; Chu et al., 2020; Chen & Hsieh, 2020; Liang et al., 2019; Zhou et al., 2020) pointed out that the bi-level optimization of DARTS (Liu et al., 2018c) suffers from performance collapse issues. Moreover, recent works (Wang et al., 2021; Xie et al., 2021) also demonstrated that the operation associated with the largest magnitude of architecture parameters does not necessarily result in the highest validation accuracy after discretization. In light of the above-mentioned observations, we propose an alternative paradigm, that is, to leverage the attention mechanism to indicate the importance of candidate operations rather than using architecture parameters.

On the other hand, the macro architecture of neural network is determined by macro search. For example, borrowing from the design of MobileNet (Howard et al., 2017), the macro architecture can be chain-style and candidate blocks of each layer are choosable, as shown in Figure 1 (b). Though prior works (Guo et al., 2020; Chu et al., 2021; You et al., 2020) have proposed to search for superb architectures with a heuristic algorithm based on the pre-trained supernet, such paradigm requires to sample and evaluate numerous sub-networks, which is still time-consuming. However, by using the attention mechanism, we can obtain the searched architecture as the search process converges, without the need for post-sampling.

In summary, we propose a novel paradigm that aims to search micro and macro architectures in one framework based on the attention mechanism to solve the aforementioned issues. Similar to how the human brain selectively focuses on certain parts of the input (Briggs et al., 2013), we demonstrate that the attention mechanism can be used to emphasize useful parts of the network while ignoring the trivial ones. Therefore, we propose to utilize attention weights to indicate the importance of candidate operations. As previous works have pointed out that channels with smaller attention weights can be pruned with neglected influence on network performance (Luo & Wu, 2020; Wang et al., 2019; Yamamoto & Maeno, 2018), we propose to accumulate the attention weights of output channels from different candidate operations to conduct the micro search, and the attention weights from different choice blocks to conduct the macro search.

Unlike prior methods (Nakai et al., 2020; Jiang et al., 2021; Jing et al., 2020; Wang et al., 2020) that included the attention module in the search space to have more candidate operations, we exploit the attention mechanism to guide the neural architecture search. As illustrated in Figure 1 (c), we insert the attention module to the output feature map of candidate operations, and then use the attention weights to identify the optimal operation with a clear advantage. Similarly, the candidate blocks (e.g. MBConv blocks in the ProxylessNAS search space) for each layer can also be optimized by the attention module for the macro search, which is shown in Figure 1 (d). As soon as the training of the supernet converges, the importance of operations or blocks is then specified by the attention weights, which are optimized together with network parameters by an efficient gradient descent algorithm such as SGD. Taking the DARTS search space as an example, the operations with the highest attention weight on each edge is kept to build the searched cell for each layer.

Our contributions can be summarized as follows:

- We first propose a novel search paradigm by leveraging the attention mechanism to efficiently search for the micro and macro neural architectures in one framework.
- We use the accumulated channel attention weights to indicate the importance of candidate operations or choice blocks by plugging the attention module behind the output of each edge in a cell or the output of each layer.
- Benefited from the attention mechanism, the truly operation strength in micro search can be more outstanding and the macro architecture can be easily obtained in an end-to-end manner.
- We conduct extensive experiments on three popular search spaces, achieving convincing results that outperform previous state-of-the-arts, clearly demonstrating the effectiveness of the proposed framework.

## 2. Related Works

**Micro Search.** Micro search aims to determine the operation associated with each edge in a cell. The micro search space proposed by NASNet (Zoph et al., 2018) is widely used in later works. For example, many works (Liu et al., 2018a; Real et al., 2019; Pham et al., 2018) proposed their search strategies in the NASNet search space, whereas their search process is discrete and usually takes thousands or even tens of thousands of GPU-hours. Later on, DARTS (Liu et al., 2018c) proposed relaxing the discrete search space to be continuous so that the search cost can be dramatically reduced by gradient decent technique. DARTS formulates a bi-level problem that simultaneously optimizes the architecture parameters and super-network weights. Unfortunately, the optimization process suffers from the generalizability (Xie et al., 2018; Dong & Yang, 2019; Chang et al., 2019; Li et al., 2020; Chu et al., 2020) and stability

(Zela et al., 2020; Chen & Hsieh, 2020; Wang et al., 2021) issues. In order to mitigate the performance gap between the super-network and discrete architectures, SNAS (Xie et al., 2018), GDAS (Dong & Yang, 2019), and DATA (Chang et al., 2019) adopted the differentiable Gumbel-Softmax to approximate one-hot encoding. SGAS (Li et al., 2020) gradually pruned redundant operations during the search process, hence the search space progressively squeezes and approximates to the final target architecture. On the other hand, RobustDARTS (Zela et al., 2020) leveraged eigenvalues of architecture parameters to monitor the search process and applied stronger regularizations to eliminate the instability. SDARTS (Chen & Hsieh, 2020) proposed a perturbation-based regularization to smooth the loss landscape to overcome the unstable optimization issue. Moreover, recent work (Wang et al., 2021) pointed out that the values of architecture parameters do not necessarily reflect operation strength and proposed a perturbation-based architecture selection strategy, which, however, requires fine-tuning after discretizing each edge, leading to substantial computation costs. Therefore, considering that the optimization collapse and the uncertainty of selecting architecture process caused by architecture parameters, we propose a novel paradigm and introduce the attention mechanism to guide the neural architecture search instead of using architecture parameters.

**Macro Search.** Macro search methods firstly train a chain-style super-network consisting of a series of choice blocks, and then derive the final optimal sub-network based on the validation accuracy. Because the super-network can be used as a basic performance estimator for different architectures, many works focus on solving the problem of super-network training fairly and efficiently in a huge search space. SPOS (Guo et al., 2020) trains the super-network through uniform path sampling. FairNAS (Chu et al., 2021) enforces fairness constraints to alleviate the super-network bias and boost the evaluation capacity. GreedyNAS (You et al., 2020) proposed to ease the training burden by encouraging to focus more on those potentially good paths instead of all paths. However, these methods require sampling many sub-networks based on the evolutionary algorithm and evaluating their performance, so the search phase is time-consuming. Instead, we introduce the attention mechanism to the super-network training, so that the search phase can be naturally integrated into the training process.

**Attention Mechanism.** The visual attention mechanism is inspired by the neuronal structure of the early primate visual system (Itti et al., 1998), and it began to attract lots of attention when (Mnih et al., 2014) applied this mechanism to the RNN model for image classification. Afterwards, lots of works (Xu et al., 2015; Wang et al., 2017; Hu et al., 2018; Woo et al., 2018; Li et al., 2019) designed different attention structures with the aim of aggregating space or channels information to improve the performance. Meanwhile, some works (Luo & Wu, 2020; Wang et al., 2019; Yamamoto & Maeno, 2018) applied the attention mechanism to prune channels with smaller attention weights. Inspired by the fact that attention weights can indicate the channel importance, we introduce it to reflect the influence of candidate operations on the super-network with the aim of guiding the neural architecture search.

## 3. Preliminary and Limitations

In the **micro** search space, the cell is defined as a directed acyclic graph (DAG) with $N$ nodes, and each edge $(i, j)$ between every node is associated with mixed operation $\bar{o}^{(i,j)}$ that is parameterized as architecture parameters $\alpha^{(i,j)}$ by using softmax relaxation. The differentiable architecture search can be formulated as a bi-level optimization problem as follows:

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

where $\alpha$ (the architecture parameters) and $w$ (the network weights) are alternately optimized on the validation and training datasets respectively.

***Limitation.*** Whereas, the architecture parameter $\alpha$ may not accurately indicate how much the operation contributes to the super-network's performance as illustrated in (Wang et al., 2021; Xie et al., 2021). So, we argue that the architecture parameter may not be necessary and propose another alternative paradigm based on the attention mechanism.

In the **macro** search space, the super-network is chained by a sequence of layers that contains many candidate choice blocks. Earlier methods divide the macro search procedure into super-network training that can be expressed as:

$$w^*(a) = \arg\min_w \mathbb{E}_{a\sim\mathcal{A}} \mathcal{L}_{train}(w, a) \tag{2}$$

After the super-network trained to convergence, then perform the sub-network searching as:

$$a^* = \arg\max_{a\sim\mathcal{A}} ACC_{val}(a, w^*(a)) \tag{3}$$

***Limitation.*** Whereas, hundreds of sub-networks need to be evaluated with the evolutionary algorithm in order to discover the optimal sub-network. Instead, we do not need to sample the sub-networks because once the super-network converges, the optimal sub-network will be indicated by the attention weights.

## 4. Methodology

### 4.1. Micro Search with Attention

The goal of the micro search is to determine each operation associated with each edge in a cell. For each searched cell,

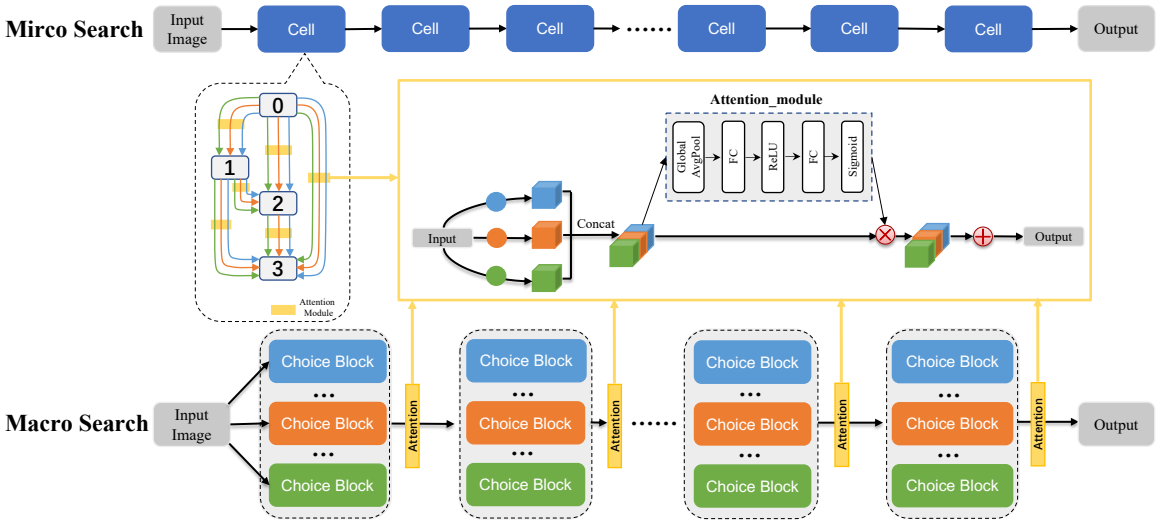



*Figure 2.* The main framework of the proposed AGNAS. Specifically, the goal of the micro search is to determine the optimal operation associated with each edge in a cell based on the attention module added on each edge. The macro search aims to discover the optimal choice block by the plug-in attention module on each layer.

the edge $(i, j)$ is associated with all $m$ candidate operations, that are applied to the predecessor input node $i$, resulting in the $m$ output feature maps $F_k^o(k = 1, 2, ..., m)$. Assume that each output feature map is $F_k^o \in \mathbb{R}^{b\times c\times h\times w}$. The DARTS approach integrates all $m$ output feature maps by element-wise addition, so the feature map of the output node $j$ is also $F^{out} \in \mathbb{R}^{b\times c\times h\times w}$.

Instead, we concatenate all $m$ output feature maps in channel dimension before delivering to the output node $j$, yielding the intermediate concatenated feature map $F^{con} \in \mathbb{R}^{b\times(m*c)\times h\times w}$. To distinguish the importance of each candidate operation, we apply the attention module to the concatenated feature map $F^{con}$. Specifically, as illustrated in Figure 2, the concatenated feature map $F^{con}$ will first go through a global average pooling to get the feature map $F_{gap}^{con} \in \mathbb{R}^{b\times(m*c)\times 1\times 1}$,

$$F_{gap}^{con} = \frac{1}{h \times w} \sum_{p=1}^{h} \sum_{q=1}^{w} F^{con}(p, q) \tag{4}$$

and then the global average pooling feature map $F_{gap}^{con}$ is applied to two fully-connected layers followed by a Sigmoid layer. To reduce the complexity of the model and improve generalisation, we use a bottleneck structure with two fully-connected layers. The first FC layer is responsible for dimensionality reduction and the second FC layer for restoring the original dimensionality,

$$Atten = \sigma(W_2 Relu(W_1 F_{gap}^{con})) \tag{5}$$

where $W_1 \in \mathbb{R}^{\frac{c}{r}\times c}$ and $W_2 \in \mathbb{R}^{c\times\frac{c}{r}}$ are the fully-connection weights, $r$ is the hyper-parameter coefficient of dimensionality reduction, $\sigma(\cdot)$ is the Sigmoid function whose output, $Atten \in \mathbb{R}^{b\times(m*c)\times 1\times 1}$, i.e., the attention weights, can be regarded as the magnitude of importance of each channel.

The attention weights here are the activation values of all channels. Therefore, the importance of the first candidate operation $A_1$ can be determined as the summation of the activation values of the first $c$ channels. Similarly, the importance of each operation is expressed as the sum of the activation values of the corresponding number of channels,

$$\begin{cases} A_1 &= \sum_{i=1}^{c} Atten(i) \\ A_2 &= \sum_{i=c}^{2c} Atten(i) \\ &\vdots \\ A_m &= \sum_{i=(m-1)c}^{mc} Atten(i) \end{cases} \tag{6}$$

Next, in order to back propagate gradient and update the weights of the attention module, the learned activation values of each channel are multiplied by the original concatenated feature map $F^{con}$ to obtain the output feature map with each re-weighted channel $\tilde{F}^{con} \in \mathbb{R}^{b\times(m*c)\times h\times w}$,

$$\tilde{F}^{con} = F^{con} \otimes Atten \tag{7}$$

Finally, the weighted feature map will be also applied the element-wise addition in the channel dimension to get the feature map $F^{out}$ of the output node $j$,

$$F^{out} = \tilde{F}_{[1:c]}^{con} \oplus \tilde{F}_{[c:2c]}^{con} \oplus \cdots \oplus \tilde{F}_{[(m-1)c:mc]}^{con} \tag{8}$$

At the end of the micro search, we forward propagate all images of the validation datasets to obtain the attention

weights associated with all candidate operations on each edge in a cell. Finally, the optimal operation on each edge is selected according to the largest aggregated attention weights,

$$A^* = argmax(A_1, A_2, \cdots, A_m) \tag{9}$$

### 4.2. Macro Search with Attention

For the macro search, we leverage the attention mechanism to search for the specific choice block at each layer of the target network. As shown in Figure 2, the super-network is chained by pre-defined $n$ choice blocks at each layer, and the attention module is plugged in the end of each layer.

The output of the previous layer will be applied to all $n$ choice blocks of the current layer to generate $n$ different feature maps $F_k^o(k = 1, 2, ..., n)$, each of which is represented as $F_k^o \in \mathbb{R}^{b\times c\times h\times w}$. Similar to the micro search, we firstly concatenate the $n$ output feature maps in the channel dimension. Then the concatenated feature map $F^{con} \in \mathbb{R}^{b\times(n*c)\times h\times w}$ is applied to the attention module to compute the channel attention weights by Eq.(4) and Eq.(5).

Afterwards, the channel activation weights are multiplied by the concatenated feature map $F^{con}$ to get the re-weighted output $\tilde{F}^{con} \in \mathbb{R}^{b\times(n*c)\times h\times w}$, and the element-wise addition in channel dimension is applied to the $\tilde{F}^{con}$ to obtain the output $F^{out}$ of this layer as in Eq.(7) and (8).

At the end of the macro search, the importance of each block at each layer can be represented by the summation of the attention weights of the corresponding number of channels on validation datasets,

$$\begin{cases} A_1 &= \sum_{i=1}^{c} Atten(i) \\ A_2 &= \sum_{i=c}^{2c} Atten(i) \\ &\vdots \\ A_n &= \sum_{i=(n-1)c}^{nc} Atten(i) \end{cases} \tag{10}$$

where $A_j(j = 1, 2, ..., n)$ represents the importance of the $j$-th choice block. Finally, the optimal choice block at each layer is selected by the largest attention weights,

$$A^* = argmax(A_1, A_2, \cdots, A_n) \tag{11}$$

### 4.3. Theoretical Analysis

The reason why attention weights can produce more meaningful and reliable results may be explained by the fact that the approach is data-dependent, so that the attention weights can vary more and better fit the data as they are different for each sample.

Moreover, we also analyze it from the perspective of the frequency principle. Notice that we use the global average pooling (GAP) as the pre-processing before computing attention weights. Whereas, GAP is actually a special case of Two-Dimensional Discrete Cosine Transform (2D-DCT) as follows,

$$\begin{aligned} f_{h,w}^{2d} &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} x_{i,j}^{2d} cos(\frac{h\pi}{H}(i+\frac{1}{2}))cos(\frac{w\pi}{W}(j+\frac{1}{2})) \\ f_{0,0}^{2d} &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} x_{i,j}^{2d} cos(\frac{0}{H}(i+\frac{1}{2}))cos(\frac{0}{W}(j+\frac{1}{2})) \\ &= \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} x_{i,j}^{2d} = GAP(x^{2d})HW \\ s.t.\ \ & h \in \{0, 1, \ldots, H-1\}, w \in \{0, 1, \ldots, W-1\} \end{aligned} \tag{12}$$

Therefore, $f_{0,0}^{2d}$ represents the lowest frequency component of 2D-DCT, and it is proportional to GAP. According to F-Principle (Xu et al., 2019b), low-frequency components help to improve the generalization of networks. That is, the operation, i.e., architecture derived by attention weights can achieve better performance than that derived by architectural parameters due to the GAP pre-processing.

**Algorithm 1** AGNAS Search Algorithm

**Input:** Supernet $\mathcal{N}$ with Attention moudule $\mathcal{M}$, Training data $\mathcal{D}_{train}$, Validation data $\mathcal{D}_{val}$.
**Output:** Attention weights $\mathcal{A}$, Searched model.

**Micro Search & Macro Search**

1: Build supernet $\mathcal{N}$ with pre-defined cells or blocks as well as the attention module $\mathcal{M}$
2: **while** not converaged **do**
3: Update network weights $w$ on training data $\mathcal{D}_{train}$ by descending $\nabla_w \mathcal{L}_{train}(w)$
4: **end while**
5: Compute attention weights $\mathcal{A}$ on validation data $\mathcal{D}_{val}$ by Eq.(6) for micro search or Eq.(10) for macro search
6: **return** derived micro cells based on the attention weights $\mathcal{A}^*$ by Eq.(9) or derived macro network based on the attention weights $\mathcal{A}^*$ by Eq.(11)

## 5. Experiments

### 5.1. Search Space

**DARTS Search Space.** Following DARTS (Liu et al., 2018c), a cell is defined as a directed acyclic graph (DAG) consisting of an ordered sequence of $N$ nodes. The input node is represented by the outputs from the previous two cells, each intermediate node aggregates information flows from all of its predecessors, and the output node is defined as a concatenation of a fixed number of its predecessors.

| Methods | Test Err.(%) | Params(M) | Search Cost (GPU-days) | Search Algorithm |
|---|---|---|---|---|
| NASNet-A (Zoph et al., 2018) | 2.65 | 3.3 | 1800 | RL |
| AmoebaNet-A (Real et al., 2019) | 3.34±0.06 | 3.2 | 3150 | EA |
| AmoebaNet-B (Real et al., 2019) | 2.55±0.05 | 2.8 | 3150 | EA |
| PNAS (Liu et al., 2018a) | 3.41±0.09 | 3.2 | 225 | SMBO |
| ENAS (Pham et al., 2018) | 2.89 | 4.6 | 0.5 | RL |
| DARTS (1st order) (Liu et al., 2018c) | 3.00±0.14 | 3.3 | 1.5 | Gradient |
| DARTS (2nd order) (Liu et al., 2018c) | 2.76±0.09 | 3.3 | 4 | Gradient |
| SNAS (Xie et al., 2018) | 2.85±0.02 | 2.8 | 1.5 | Gradient |
| GDAS (Dong & Yang, 2019) | 2.93 | 3.4 | 0.21 | Gradient |
| BayesNAS (Zhou et al., 2019) | 2.81±0.04 | 3.4 | 0.2 | Gradient |
| Robust-DARTS (Zela et al., 2020) | 2.95±0.21 | N/A | 1.6 | Gradient |
| PC-DARTS (Xu et al., 2019a) | 2.57±0.07 | 3.6 | 0.1 | Gradient |
| DATA (Chang et al., 2019) | 2.59 | 3.4 | 1 | Gradient |
| SGAS(Cri.1 avg.) (Li et al., 2020) | 2.66±0.24 | 3.7 | 0.25 | Gradient |
| SDARTS-ADV (Chen & Hsieh, 2020) | 2.61±0.02 | 3.3 | 1.3 | Gradient |
| DARTS+PT (Wang et al., 2021) | 2.61±0.08 | 3.0 | 0.8 | Gradient |
| AGNAS (avg.) | 2.53±0.003 | 3.6 | 0.4 | Gradient |
| AGNAS (best) | **2.46** | 3.6 | 0.4 | Gradient |

*Table 1.* Search results on CIFAR-10 and comparison with other state-of-the-art methods. Unlike other methods that only perform the micro search for identical cells, AGNAS firstly conducts the micro search to obtain the different cells and then builds the target network by stages. We report the average results for three independent runs with different initial random seeds.

**ProxylessNAS Search Space.** Following ProxylessNAS (Cai et al., 2018), we adopted the MobileNet-like search space in our experiments. The super-network consists of 21 choice blocks, each of which has 6 candidate choices with different kernel size {3,5,7} and expansion ratio {3, 6}, as well as an alternate candidate choice of skip-connect.

**NAS-Bench-201 Search Space.** NAS-Bench-201 (Dong & Yang, 2020) is a benchmark for almost up-to-date NAS algorithms. Specifically, the super-network is stacked by cells, but it only requires searching for normal cells and maintaining the reduction cells as residual blocks with a stride of two. The normal cell is built by six edges and five candidate operations associated with each edge, resulting in a total of 15,625 neural architectures in the search space. The diagnostic information about accuracy, loss, and parameters is accessible on three datasets including CIFAR-10, CIFAR-100, and ImageNet-16-120.

### 5.2. Results on CIFAR-10

We conduct the micro search by constructing a super-network with eight cells in the DARTS search space. Reduction cells are located at the 1/3 and 2/3 of the network, whereas the rest are normal cells. The super-network plugged by the attention module is trained on half of the CIFAR-10 (Krizhevsky et al., 2009) training datasets. In particular, we use the SGD optimizer to update network weights with initial learning rate of 0.025, momentum 0.9, and weight decay $3 \times 10^{-4}$. We search for 50 epochs with the batch size of 64. The micro search process elapses nine hours on 1080Ti GPU with only 10 GB GPU memory.

At the end of the micro search process, we can obtain eight different cells according to the attention weights by forwarding propagation on the validation datasets. We divide the super-network into three stages based on the location of the reduction cell. And one of the normal cells is then selected at each stage to build the target network.

Finally, in the evaluating phase, the target network which consists of 20 layers with initial channel size of 36 is trained on the whole CIFAR-10 training datasets. We use an SGD optimizer with a weight decay of $3 \times 10^{-4}$ and a momentum of 0.9. The initial learning rate starts from 0.025 and follows the cosine annealing strategy to a minimum of 0. The network is trained from scratch for 600 epochs with batch size of 96.

As shown in Table 1, the results show that our method achieves state-of-the-art performance with the accuracy of 97.54% compared with other methods in the DARTS search space. Differently, all the previous methods stack the same searched cell to evaluate the performance. However, we build the target network using the different normal cells at different stages, and can obtain the average accuracy of 97.47%, indicating that our method can search for a stable architecture even with different seed initialization.

| Methods | Test Err. (%) | | Params | Search Cost | Search |
|---|---|---|---|---|---|
| | Top-1 | Top-5 | (M) | (GPU-days) | Algorithm |
| MnasNet (Tan et al., 2019) | 26 | 8.2 | 4.2 | 2000 | RL |
| NASNet (Zoph et al., 2018) | 26.0 | 8.4 | 5.3 | 1800 | RL |
| AmoebaNet (Real et al., 2019). | 24.3 | 7.6 | 6.4 | 3150 | EA |
| PNAS (Liu et al., 2018a) | 25.8 | 8.1 | 5.1 | 225 | SMBO |
| FBNet-C (Wu et al., 2019) | 25.1 | 7.9 | 5.5 | 9 | Gradient |
| ProxylessNAS(GPU) (Cai et al., 2018) | 24.9 | 7.5 | 7.1 | 8.3 | Gradient |
| SPOS (Guo et al., 2020) | 26.0 | 8.4 | 5.3 | 11‡ | Evolution |
| FairNAS-A (Chu et al., 2021) | 24.66 | 7.8 | 4.6 | 16‡ | Evolution |
| GreedyNAS-C (You et al., 2020) | 23.8 | 7.5 | 4.7 | 8‡ | Evolution |
| RLNAS (Zhang et al., 2021) | 24.4 | 7.4 | 5.3 | N/A | Evolution |
| AGNAS | **23.4** | **6.8** | 6.7 | 3.3 | Gradient |

*Table 2.* Search results on ImageNet and comparison with other state-of-the-art methods. ‡ denotes the search cost includes the additional subnet searching with the evolutionary algorithm.

| Methods | CIFAR-10 | | CIFAR-100 | | ImageNet-16-120 | |
|---|---|---|---|---|---|---|
| | validation | test | validation | test | validation | test |
| Optimal | 91.61 | 94.37 | 73.49 | 73.51 | 46.77 | 47.31 |
| RSPS | 80.42±3.58 | 84.07±3.61 | 52.12±5.55 | 52.31±5.77 | 27.22±3.24 | 26.28±3.09 |
| DARTS | 39.77±0.00 | 54.30±0.00 | 15.03±0.00 | 15.61±0.00 | 16.43±0.00 | 16.32±0.00 |
| GDAS | 89.89±0.08 | 93.61±0.09 | 71.34±0.04 | 70.70±0.30 | 41.59±1.33 | 41.71±0.98 |
| SETN | 84.04±0.28 | 87.64±0.00 | 58.86±0.06 | 59.05±0.24 | 33.06±0.02 | 32.52±0.21 |
| ENAS | 37.51±3.19 | 53.89±0.58 | 13.37±2.35 | 13.96±2.33 | 15.06±1.95 | 14.84±2.10 |
| AGNAS | **91.25±0.019** | **94.05±0.059** | **72.4±0.382** | **72.41±0.061** | **45.5±0.003** | **45.98±0.457** |

*Table 3.* Search results on NAS-bench-201. We report the average performance for three independent runs of searching. "Optimal" indicates the highest accuracy for each dataset on NAS-Bench-201.

### 5.3. Results on ImageNet

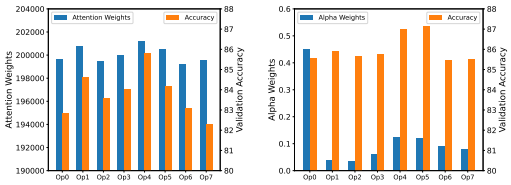
We conduct the macro search directly on ImageNet datasets (Krizhevsky et al., 2017) in the ProxylessNAS search space. Specifically, the super-network is chained by a series of blocks, each of which contains many candidate choice operations. The super-network with the added attention module is trained on 8 NVIDIA V100 GPUs on 10% of the training datasets for 50 epochs with a batch size of 64 per GPU. After that, the optimal choice operation for each layer is determined according to the channel attention weights that are computed on the validation datasets.

The final derived network is trained from scratch on the entire ImageNet training datasets. We use the SGD optimizer with an initial learning rate of 0.5, weight decay of $4 \times 10^{-5}$, and momentum of 0.9. The network is trained for 240 epochs with the batch size of 1024 on 8 NVIDIA V100 GPUs.

From Table 2, we can see that by directly searching on ImageNet within 3.3 GPU-days, AGNAS achieves Top-1 test error of 23.4% and Top-5 test error of 6.8%. The results show that the proposed method is very efficient and significantly outperforms the other methods.

### 5.4. Results on NAS-Bench-201

In NAS-Bench-201 search space, we add the attention module to each edge in a cell. It is worth noting that the benchmark can only retrieve the super-network stacked by the same cell. Therefore, we average the attention weights of the edges in the same position of all cells for selecting the corresponding operations in order to obtain the final single cell. We search for 50 epochs on CIFAR-10 and then index the accuracy of the searched architecture on the three datasets CIFAR-10, CIFAR-100, and ImageNet-16-120, respectively. We keep the hyperparameters the same as DARTS, and repeat the experiment three times with different initial random seeds. The results are shown in Table 3. Compared to other methods, AGNAS achieves the best performance on three datasets. Furthermore, the best accuracies are all close to the global optimal, demonstrating the effectiveness of the neural architecture with the attention mechanism.




(a) AGNAS

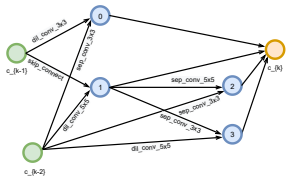



(b) DARTS

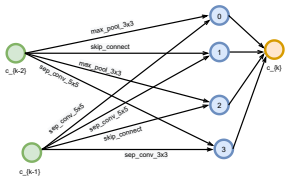
| Discretization Accuracy | Attention Weights | Operation | Alpha Weights | Discretization Accuracy |
|---|---|---|---|---|
| 85.79 | 199663 | none | 0.452 | 87.14 |
| 84.62 | 200717 | max_pool_3x3 | 0.038 | 86.97 |
| 84.16 | 199431 | avg_pool_3x3 | 0.034 | 85.89 |
| 84.00 | 199984 | skip_connect | 0.061 | 85.75 |
| 83.58 | 201223 | sep_conv_3x3 | 0.124 | 85.63 |
| 83.07 | 200489 | sep_conv_5x5 | 0.120 | 85.55 |
| 82.81 | 199193 | dil_conv_3x3 | 0.091 | 85.53 |
| 82.28 | 199547 | dil_conv_5x5 | 0.080 | 85.45 |
| AGNAS Kendall = 0.71 | | | DARTS Kendall = 0 | |

(c) The ranking of attention weights or alpha weights against discretization accuracy.

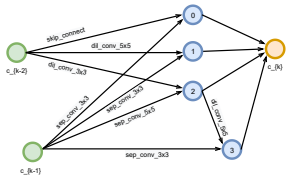
*Figure 3.* Comparison of attention weights and architecture parameters on measuring the importance of candidate operations.



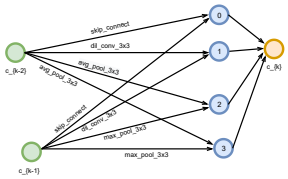
(a) Cell_0



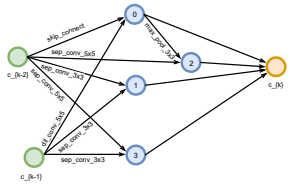
(b) Cell_2



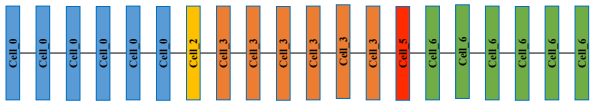
(c) Cell_3



(d) Cell_5



(e) Cell_6




(f) Target network built on CIFAR-10

*Figure 4.* Micro cell searched on CIFAR-10 and the evaluated target network.

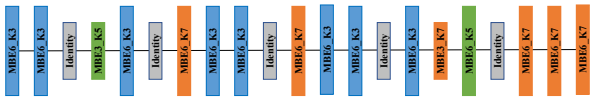



*Figure 5.* Macro architecture searched on ImageNet.

## 6. Discussion

### 6.1. Attention Weight vs. Architecture Parameter

Here we investigate the effect of the attention weights vs. architecture parameters on the performance of the super-network. Specifically, we experiment AGNAS and DARTS on CIFAR-10 dataset, and randomly discretize an edge in a cell at the end of the search phase. The super-network is expected to get the highest *discretization accuracy* if we keep the optimal operation measured by the attention weight or alpha weights on the edge and prune the other operations. Moreover, we use the Kendall $\tau$ metric (Kendall, 1938) to measure the correlation between two rankings of alpha weights or attention weights and discretization accuracy. The closer this metric is to 1, the higher the correlation between the two sequences; otherwise, the opposite is true.

With the instance of both DARTS and AGNAS discretizing the second edge in the first cell, we compare the discretization accuracy of the super-network when only one of the operations is retained on this edge. The Op_4 has the largest attention weight, which also bring the highest discretization accuracy, as demonstrated in Figure 3 (a). Nonetheless, as shown in Figure 3 (b), the largest alpha weight on Op_4 does not reach the highest accuracy after discretizing the edge, showing that the alpha weight cannot reflect the true strength of the operation. Furthermore, our approach achieves higher Kendall $\tau$ than DARTS as shown in Figure 3 (c), which means that attention weights better reflect operational importance than architectural parameters. More details are provided in the *Appendix C*.

### 6.2. Visualization

Here we visualize the searched micro cells and target network on CIFAR-10. We obtain a total of eight different cells in the micro search stage, five of which are shown in Figure 4 (a) (b) (c) (d) (e), where the (b) and (d) are reduction cells, and the remaining normal cells are built the final target network at different stages, which is shown in Figure 4 (f). In addition, as shown in Figure 5, we visualize the macro architecture directly searched in the ProxylessNAS search space on ImageNet.

### 6.3. Limitations

Although AGNAS achieves good performance, we must acknowledge that the resource consumption is a little higher

than architecture parameter-based methods (e.g., AGNAS: 9.7 GB GPU-Memory vs. DARTS: 9.4 GB GPU-Memory). We are also exploring whether there are more appropriate mechanisms to determine the true importance of candidate operations.

## 7. Conclusion

In this paper, we propose to leverage the attention mechanism to search for micro and macro architectures in one framework. We use the attention weights to measure the importance of candidate operations and choice blocks. Extensive experiments validate that the proposed AGNAS is effective and significantly outperforms state-of-the-art approaches. In the future, we will simultaneously search for micro- and macro- architecture in the AGNAS framework and apply AGNAS to other computer vision tasks such as object detection and semantic segmentation.

## Acknowledgement

This work is supported in part by the National Key R&D Program of China under Grant No. 2018AAA0102701, and in part by the National Natural Science Foundation of China under Grant No. 62176250.

## References

Baker, B., Gupta, O., Naik, N., and Raskar, R. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. Neural optimizer search with reinforcement learning. In *ICML*, 2017.

Briggs, F., Mangun, G. R., and Usrey, W. M. Attention enhances synaptic efficacy and the signal-to-noise ratio in neural circuits. *Nature*, 499(7459):476–480, 2013.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2018.

Chang, J., Zhang, X., Guo, Y., Meng, G., Xiang, S., and Pan, C. Data: Differentiable architecture approximation. In *NeurIPS*, 2019.

Chen, L.-C., Collins, M. D., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., and Shlens, J. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, 2018.

Chen, X. and Hsieh, C.-J. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020.

Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., and Sun, J. Detnas: Backbone search for object detection. In *NeurIPS*, 2019.

Chu, X., Zhou, T., Zhang, B., and Li, J. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020.

Chu, X., Zhang, B., and Xu, R. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, 2021.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019.

Dong, X. and Yang, Y. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *CVPR*, 2018.

Itti, L., Koch, C., and Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20 (11):1254–1259, 1998.

Jiang, H., Shen, F., Gao, F., and Han, W. Learning efficient, explainable and discriminative representations for pulmonary nodules classification. *Pattern Recognition*, 113: 107825, 2021.

Jing, K., Xu, J., and Zugeng, H. X. Nasabn: A neural architecture search framework for attention-based networks. In *IJCNN*, 2020.

Kendall, M. G. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Li, G., Qian, G., Delgadillo, I. C., Muller, M., Thabet, A., and Ghanem, B. Sgas: Sequential greedy architecture search. In *CVPR*, 2020.

Li, X., Wang, W., Hu, X., and Yang, J. Selective kernel networks. In *CVPR*, 2019.

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

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *ECCV*, 2018a.

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., and Fei-Fei, L. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. In *ICLR*, 2018b.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *ICLR*, 2018c.

Luo, J.-H. and Wu, J. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107:107461, 2020.

Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. In *NeurIPS*, 2014.

Nakai, K., Matsubara, T., and Uehara, K. Att-darts: Differentiable neural architecture search for attention. In *IJCNN*, 2020.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *ICML*, 2018.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. Large-scale evolution of image classifiers. In *ICML*, 2017.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. Residual attention network for image classification. In *CVPR*, 2017.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021.

Wang, X., Xiong, X., Neumann, M., Piergiovanni, A., Ryoo, M. S., Angelova, A., Kitani, K. M., and Hua, W. Attentionnas: Spatiotemporal attention cell search for video classification. In *ECCV*, 2020.

Wang, X.-J., Yao, W., and Fu, H. A convolutional neural network pruning method based on attention mechanism. In *SEKE*, 2019.

Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. Cbam: Convolutional block attention module. In *ECCV*, 2018.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019.

Xie, S., Zheng, H., Liu, C., and Lin, L. Snas: stochastic neural architecture search. In *ICLR*, 2018.

Xie, X., Zhou, Y., and Kung, S.-Y. Exploiting operation importance for differentiable neural architecture search. In *TNNLS*, 2021.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2019a.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y., and Ma, Z. Frequency principle: Fourier analysis sheds light on deep neural networks. In *ICML*, 2019b.

Yamamoto, K. and Maeno, K. Pcas: Pruning channels with attention statistics for deep network compression. In *BMVC*, 2018.

You, S., Huang, T., Yang, M., Wang, F., Qian, C., and Zhang, C. Greedynas: Towards fast one-shot nas with greedy supernet. In *CVPR*, 2020.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

Zhang, X., Hou, P., Zhang, X., and Sun, J. Neural architecture search with random labels. In *CVPR*, 2021.

Zhou, H., Yang, M., Wang, J., and Pan, W. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, 2019.

Zhou, P., Xiong, C., Socher, R., and Hoi, S. C. Theory-inspired path-regularized differential network architecture search. In *NeurIPS*, 2020.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.

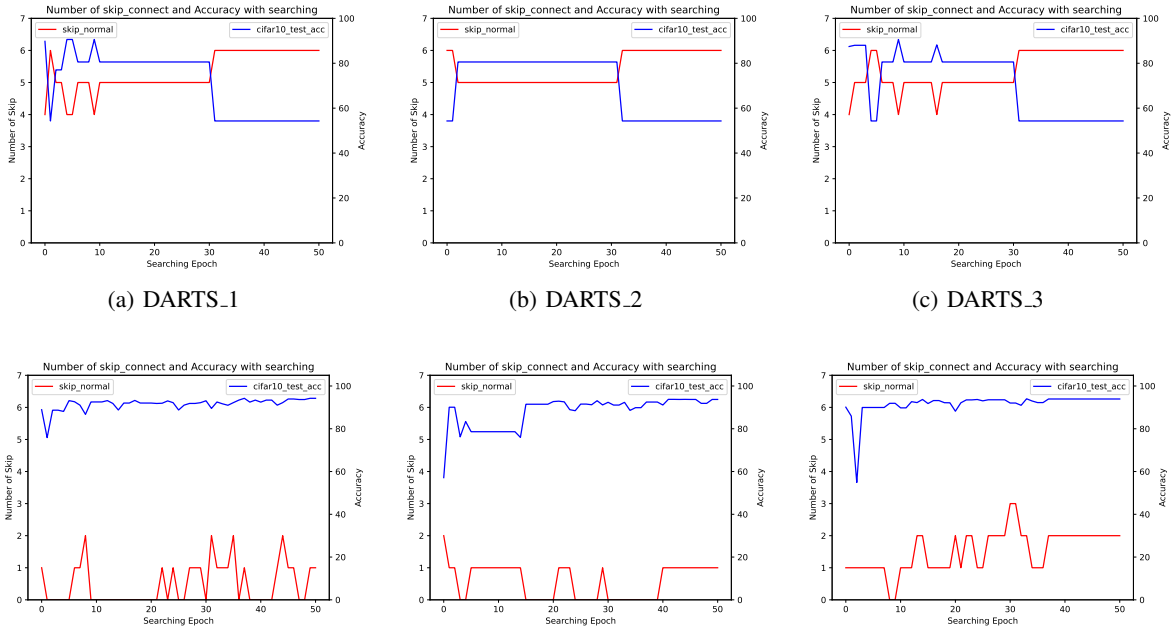
## A. Datasets

We conduct the micro search and macro search on two popular image classification datasets including CIFAR-10 and ImageNet. CIFAR-10 datasets have 50K training RGB images and 10K testing RGB images with a fixed spatial resolution of 32×32. In the search phase, we split the training datasets in half to train the super-network weights and the attention module weights, and the other half as validation datasets to select the optimal operation in micro search by forwarding propagation of all images to get the corresponding attention weights. The ILSVRC2012 ImageNet dataset contains 1.28M training and 50K validation images with 1000 object categories. In the macro search phase, we sample 10% of the training datasets to update the super-network and the attention module weights, and 2.5% training datasets as validation datasets are used to select the final choice block at each layer based on the attention weights.

## B. Analysis of Collapse

DARTS introduces the architecture parameters and leverages a bi-level optimization to alternately optimize the architecture parameters and network weights. However, the optimization may prefer non-parametric operations, especially skip-connections, thus resulting in the performance collapse issue. Therefore, we investigate how the number of skip-connection for DARTS and AGNAS changes througout the search process in the NAS-Bench-201 search space.

We repeat the experiments three times with different initialized seeds, and the results are shown in Figure 6. We plot the number of skip-connection per epoch together with the test accuracy in each experiment. As shown in Figure 6 (a) (b) (c), the searched cells by DARTS are all consisted of skip-connections, and the test accuracy also deteriorates to roughly 53% at the last search epoch, thus indicating the performance collapse induced by architecture parameters. However, as demonstrated in Figure 6 (d) (e) (f), AGNAS achieves state-of-the-art performance with the proposed attention mechanism and the number of skip-connection is no more than three during the search process. The results indicate that the effectiveness of utilizing attention weights to quantify the importance of candidate operations to derive the final cell.



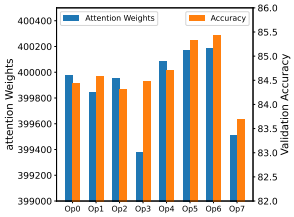

(a) DARTS_1 (b) DARTS_2 (c) DARTS_3

(d) AGNAS_1 (e) AGNAS_2 (f) AGNAS_3

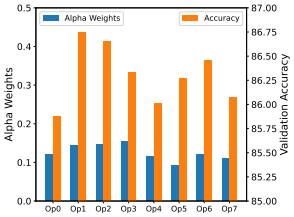
*Figure 6.* The number of skip-connection and test accuracy on CIFAR-10 as the search process in the NAS-Bench-201. We repeat each experiment three times with different initial seeds.

## C. Attention Weight vs. Architecture Parameter

We have discretized additional edges in different cells to illustrate that the attention weights are more consistent than architectural parameters in measuring the importance of candidate operations. As shown in Figure 7-9, when AGNAS discretizes the corresponding edges in different cells, the operations with the largest attention weight all achieve the highest discretization accuracy. Whereas, DARTS with the largest alpha weight does not reach the highest accuracy after discretizing the same edge, and the ranking consistency, i.e., the Kendall $\tau$ is also inferior to AGNAS, indicating that the attention weights can better reflect the operational importance than architecture parameters.




(a) AGNAS

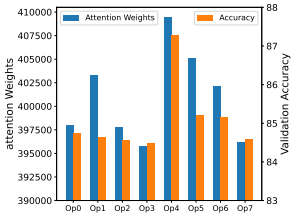



(b) DARTS

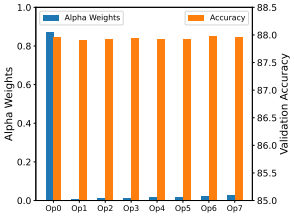
| Discretization Accuracy | Attention Weights | Operation | Alpha Weights | Discretization Accuracy |
|---|---|---|---|---|
| 85.43 | 399506 | none | 0.121 | 86.74 |
| 85.32 | 400182 | max_pool_3x3 | 0.143 | 86.65 |
| 84.70 | 400170 | avg_pool_3x3 | 0.147 | 86.46 |
| 84.58 | 400084 | skip_connect | 0.153 | 86.34 |
| 84.48 | 399979 | sep_conv_3x3 | 0.114 | 86.26 |
| 84.43 | 399844 | sep_conv_5x5 | 0.093 | 86.07 |
| 84.30 | 399949 | dil_conv_3x3 | 0.120 | 86.01 |
| 83.69 | 399378 | dil_conv_5x5 | 0.109 | 85.87 |
| AGNAS Kendall = 0.64 | | | | DARTS Kendall = 0.21 |

(c) The ranking of attention weights or alpha weights against discretization accuracy.

*Figure 7.* Comparison of attention weights and architecture parameters on measuring the importance of candidate operations both on the 1st edge of the 2nd cell.




(a) AGNAS

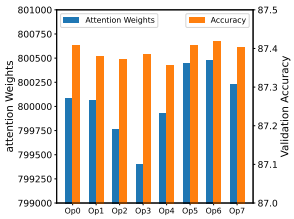



(b) DARTS

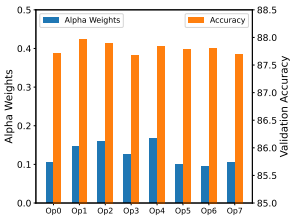
| Discretization Accuracy | Attention Weights | Operation | Alpha Weights | Discretization Accuracy |
|---|---|---|---|---|
| 87.28 | 396245 | none | 0.871 | 87.97 |
| 85.20 | 402170 | max_pool_3x3 | 0.010 | 87.95 |
| 85.17 | 405135 | avg_pool_3x3 | 0.011 | 87.95 |
| 84.74 | 409454 | skip_connect | 0.015 | 87.93 |
| 84.64 | 398013 | sep_conv_3x3 | 0.020 | 87.92 |
| 84.58 | 203271 | sep_conv_5x5 | 0.019 | 87.92 |
| 84.56 | 397763 | dil_conv_3x3 | 0.025 | 87.91 |
| 84.49 | 395805 | dil_conv_5x5 | 0.029 | 87.90 |
| AGNAS Kendall = 0.79 | | | | DARTS Kendall = 0.57 |

(c) The ranking of attention weights or alpha weights against discretization accuracy.

*Figure 8.* Comparison of attention weights and architecture parameters on measuring the importance of candidate operations both on the 13th edge of the 3rd cell.




(a) AGNAS




(b) DARTS

| Discretization Accuracy | Attention Weights | Operation | Alpha Weights | Discretization Accuracy |
|---|---|---|---|---|
| 87.42 | 800229 | none | 0.104 | 87.98 |
| 87.41 | 800471 | max_pool_3x3 | 0.147 | 87.90 |
| 87.41 | 800448 | avg_pool_3x3 | 0.158 | 87.84 |
| 87.40 | 799924 | skip_connect | 0.126 | 87.81 |
| 87.38 | 800078 | sep_conv_3x3 | 0.166 | 87.79 |
| 87.38 | 800057 | sep_conv_5x5 | 0.099 | 87.72 |
| 87.37 | 799757 | dil_conv_3x3 | 0.094 | 87.69 |
| 87.36 | 799397 | dil_conv_5x5 | 0.105 | 87.68 |
| AGNAS Kendall = 0.62 | | | | DARTS Kendall = 0.07 |

(c) The ranking of attention weights or alpha weights against discretization accuracy.

*Figure 9.* Comparison of attention weights and architecture parameters on measuring the importance of candidate operations both on the 10th edge of the 5th cell.