# EXPLORING COMPLICATED SEARCH SPACES WITH INTERLEAVING-FREE SAMPLING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The existing neural architecture search algorithms are mostly working on search spaces with short-distance connections. We argue that such designs, though safe and stable, obstacles the search algorithms from exploring more complicated scenarios. In this paper, we build the search algorithm upon a complicated search space with long-distance connections, and show that existing weight-sharing search algorithms mostly fail due to the existence of **interleaved connections**. Based on the observation, we present a simple yet effective algorithm named **IF-NAS**, where we perform a periodic sampling strategy to construct different sub-networks during the search procedure, avoiding the interleaved connections to emerge in any of them. In the proposed search space, IF-NAS outperform both random sampling and previous weight-sharing search algorithms by a significant margin. IF-NAS also generalizes to the micro cell-based spaces which are much easier. Our research emphasizes the importance of macro structure and we look forward to further efforts along this direction.

## 1 INTRODUCTION

Neural architecture search (NAS) is a research field that aims to automatically design deep neural networks (Zoph & Le, 2017; Real et al., 2019; Baker et al., 2017). There are two important factors that define a NAS algorithm, namely, the search space that determines what kinds of architectures can appear, and the search strategy that explores the search space efficiently. Despite the rapid development of search algorithms which have become faster and more effective, the search space design is still in a preliminary status. In particular, for the most popular search spaces used in the community, either MobileNet-v3 (Howard et al., 2019) or DARTS (Liu et al., 2019b), the macro structure (*i.e.*, how the network blocks are connected) is not allowed to change. Such a conservative strategy is good for search stability (*e.g.*, one can guarantee to achieve good performance even with methods that are slightly above random search), but it reduces the flexibility of NAS, impeding the exploration of more complicated (and possibly more effective) neural architectures.

The goal of this paper is to break through the limitation of existing search spaces. For this purpose, we first note that the MobileNet-v3 and DARTS allow a cell to be connected to 1 and 2 precursors, respectively, resulting in rel-

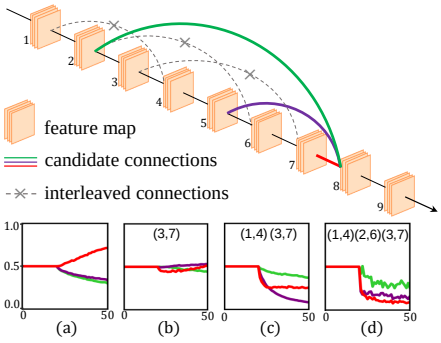


Figure 1: Illustration of how interleaved connections contaminate the choice of neural network connections. The red, purple, and green links denote three candidate connections, among which the red one is the best choice (according to the individual evaluation). However, when interleaved connections (the dashed links) are added for the search sampling, the judgment of the candidates gradually becomes ridiculous. The bottom figures indicate the weight change of the three candidates throughout the search process (warm-up for 20 epochs), while (a) is under the interleaving-free setting and 1–3 interleaved connections are added for (b)–(d). Specifically, the interleaved connection (3,7) is added for (b), the connections (1,4), (3,7) added for (c), and (1,4), (2,6), (3,7) added for (d).

atively simple macro structures. In opposite, we propose a variant that each cell is connected to $L$ precursors ($L$ is 4, 6, or 8), and each connection can be either present or absent. We evaluate three differentiable NAS algorithms, namely DARTS (Liu et al., 2019b), PC-DARTS (Xu et al., 2020), and GOLD-NAS (Bi et al., 2020) in the designed $L$-chain search space, and all of them run into degraded results. We perform diagnosis in the failure cases and the devil turns out to be the so-called **interleaved connections**, which refers to a pair of connections $(a, b)$ and $(c, d)$ that satisfies $a < c < b < d$. Figure 1 shows an example that how interleaved connections affects the search results. With the increasing extent of interleaving, the search results gradually deteriorate, reflecting in the reduced accuracy and the weak operator gaining heavier weights. More examples are provided in the appendix.

The above observation motivates us to maximally eliminate the emerge of interleaved connections during the search procedure. This is easily done by performing interleaving-free sampling, where we group all candidate connections into $L$ groups and there exist no interleaved connections in every single group, based on which we periodically choose one group and perform regular NAS algorithms. This schedule can optimize the weight of every connection without suffering the issue of interleaved connections. Discretization and pruning are performed afterwards to derive the final architecture. The entire algorithm is named interleaving-free NAS, or **IF-NAS** for short.

We conduct experiments on ImageNet, a popular benchmark of NAS. In the newly proposed $L$-chain space, IF-NAS significantly outperforms three differentiable search baselines, DARTS, PC-DARTS and GOLD-NAS, and the advantage becomes more evident as the number of possible input blocks grows larger, *i.e.*, heavier interleaving presents. Moreover, we evaluate IF-NAS in the existing search spaces of DARTS and GOLD-NAS, and show that it generalizes well to these easier search spaces.

In summary, the contributions of this paper are two-fold. **First**, we advocate for investigating the macro structure and put forward a novel search space for this purpose. The existing NAS methods cannot guarantee satisfying performance in this space. **Second**, we show that the major difficulty lies in dealing with the interleaved connections and hence propose an effective solution named IF-NAS. We hope that our efforts can inspire the NAS community to study the challenging new problem.

## 2 Related Work

**Search Strategy** Early NAS methods generally rely on individually evaluating the sampled sub-architectures under heuristic strategies, including reinforcement learning (Zoph & Le, 2017) and the evolutionary algorithm (Real et al., 2019), which are computationally expensive. To accelerate the search, one-shot methods (Bender et al., 2018; Brock et al., 2018; Guo et al., 2020b) propose to represent the search space with a super-network, where the weights of all the candidate architectures are shared. Individual architecture training from scratch is avoided and the search cost is reduced by large magnitudes. Recently differentiable NAS (DNAS) has aroused great popularity in this field, which maps the discrete search space into a parameterized super network so that the search process can be executed by gradient descent. DARTS (Liu et al., 2019b), as a pioneer differentiable framework, relaxes the search space by introducing updatable architecture parameters. The bi-level optimization is performed to update super-network weights and architectural parameters alternately. The target architecture is derived according to the distribution of architectural parameters. Due to its high efficiency, many works extend DNAS to more applications, including semantic segmentation (Liu et al., 2019a; Zhang et al., 2019), object detection (Fang et al., 2020b; Guo et al., 2020a), *etc*. Some DNAS works (Cai et al., 2019; Wu et al., 2019; Fang et al., 2020a) propose to integrate co-optimization with both accuracy and hardware properties into the search phase. In this paper, we target at promoting DNAS in terms of both flexibility and robustness. With the two factors improved, the final performance of DNAS can reach a higher level.

**Eliminating Collapse in DNAS** Though DNAS has achieved great success due to its high efficiency, many inherent problems exist with it and may cause collapse during search. A series of methods propose to improve DNAS from various perspectives. P-DARTS (Chen et al., 2019) bridges the gap between the super and searched network by gradually increasing the depths the networks. FairDARTS (Chu et al., 2020) improves the sampling strategy and breaks the two indispensable factors of unfair advantages and exclusive competition. (Tian et al., 2021; Bi et al., 2020) alleviate the discretization error by pushing the weights to sharp distributions. (Liang et al., 2019; Li et al., 2019;

Zela et al., 2020) robustify the search process by introducing regularization techniques, including early stopping and observing eigenvalues of the validation loss Hessian, *etc*. We reveal the degradation phenomenon in complicated search spaces when using the conventional DNAS method, and propose to suppress the interleaved connections during search. The proposed IF-NAS eliminates collapse most DNAS methods may encounter and shows superior performance.

**Search Space** Most existing NAS methods only explore in the micro search space, while the limited flexibility hinders further development of NAS. The cell-based space is firstly proposed in NASNet (Zoph & Le, 2017), which is widely adopted by the following works (Pham et al., 2018; Real et al., 2019; Zhou et al., 2019; Liu et al., 2019b). This type of search space takes several nodes into one cell structure, which though eases the search procedure, suppresses many possible architectures with stronger feature extraction ability. (Liu et al., 2018) proposes to search architectures under a hierarchical space which introduces flexible topology structures. Auto-DeepLab (Liu et al., 2019a) searches in a space with multiple paths and allows feature extraction in diverse resolutions. DenseNAS (Fang et al., 2020a) introduces densely connections in the search space and improves search freedom in terms of operators, depths and widths. GOLD-NAS (Bi et al., 2020) liberates the restriction of the cell-based design and performs search in a global range. We further extends the search space complexity to explore architectures with more possibilities and potential. Though intractable is the search in such a complicated space, our proposed IF-NAS still shows evident effectiveness and advantages over other compared DNAS methods.

## 3 Our Approach

### 3.1 Preliminaries: NAS in a Super-Network

In neural architecture search (NAS), a deep neural network can be formulated into a mathematical function that receives an image $\mathbf{x}$ as the input and produces the desired information (*e.g.*, a class label $y$) as the output. We denote the function to be $y = f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\omega})$ where the form of $f(\cdot)$ is determined by a set of architectural parameters, $\boldsymbol{\alpha}$, and the learnable weights (*e.g.*, the convolutional weights) are denoted by $\boldsymbol{\omega}$. The goal of NAS is to find the optimal architecture, $\boldsymbol{\alpha}^\star$, that leads to the best performance, *i.e.*,

$$\begin{aligned} \boldsymbol{\alpha}^\star &= \arg\min_{\boldsymbol{\alpha}} \mathbb{E}_{(\mathbf{x},y^\star)\in\mathcal{D}_{\text{val}}} |y^\star - f(\mathbf{x})| \\ \text{s.t.} \quad \boldsymbol{\omega}^\star(\boldsymbol{\alpha}) &= \arg\min_{\boldsymbol{\omega}} \mathbb{E}_{(\mathbf{x},y^\star)\in\mathcal{D}_{\text{train}}} |y^\star - f(\mathbf{x})|, \end{aligned} \tag{1}$$

where $y^\star$ denotes the ground truth label. Most often, $\boldsymbol{\alpha}$ takes discrete values, implying that solving Eqn equation 1 requires enumerating a large number of sampled architectures and performing individual evaluation. To accelerate, researchers propose to slack $\boldsymbol{\alpha}$ into a continuous form so that solving Eqn equation 1 involves optimizing a super-network, after which $\boldsymbol{\alpha}^\star$ is discretized into the optimal architecture for other applications.

In particular, this paper is built upon the differentiable search algorithms, in which the super-network is solved by computing the gradient with respect to $\boldsymbol{\alpha}$. We will introduce the details of optimization in the experimental part.

### 3.2 Exploring a Complicated Search Space

We design a search space shown in Figure 2. We define a fixed integer, $L$, indicating that each layer can be connected to $L$ precursors. For convenience, we name this space $L$**-chain space**. When $L = 1$, it degrades to the chain-styled network (the backbone of MobileNet-v3). On the contrary, we study the cases of $L = 4$, $L = 6$, and even $L = 8$, making the topology of the space

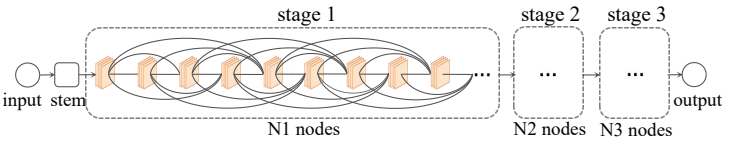


Figure 2: The studied search space in this paper. Each layer (also called node) can be connected to $L$ precursors. 3 stages are to be searched with N1, N2 and N3 nodes respectively. For better visualization, we show the example of $L = 4$, yet we also study $L = 6$ and $L = 8$ which are even more complicated.

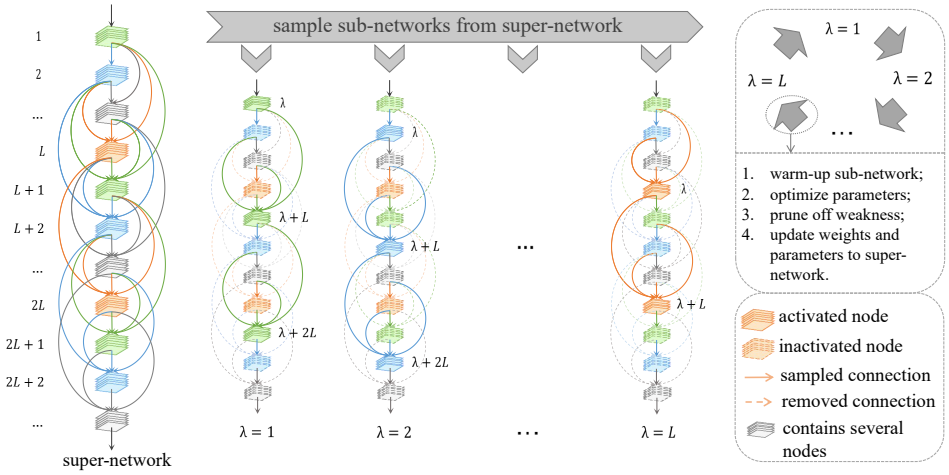


Figure 3: The flowchart of IF-NAS. The key is to avoid interleaved connections in *any* time. For this purpose, we repeat a loop with a length of $L$, and each time part of the connections remain active and all others are removed. The layers of the same color are always activated and inactivated together. This figure is best viewed in color.

much more complicated. We follow R-DARTS (Zela et al., 2020) to allow two candidate operations, *i.e.* the $3 \times 3$ separable convolution and skip-connection. Throughout the remaining part of this paper, we use $\mathbb{S}^{(L)}$ to indicate the proposed space with $L$ precursor connections. For convenience, the connection between the $n$-th and $(n-l)$-th layers (nodes) is named the pre-$l$ connection of the $n$-th layer.

Before entering the algorithm part, we emphasize that we do not use additional rules to assist the architecture search, *e.g.*, forcing all nodes to survive by preserving at least one connection. This raises new challenges to the search algorithm, because the depth of the searched architecture becomes quite indeterminate. Although some prior work has explored the option of optimizing the macro and micro architectures jointly (Liu et al., 2019a) or adding mid-range connections to the backbone (Fang et al., 2020a), they still rely on the cell/block unit to perform partial search. Instead, we break the limitations of the unit, and allow search in a wider and macro space.

Table 1: Comparisons of the search space complexity.

| **DARTS** | **GOLD-NAS** | 4-**chain** | 6-**chain** | 8-**chain** |
|---|---|---|---|---|
| $1.1 \times 10^{18}$ | $3.1 \times 10^{117}$ | $1.8 \times 10^{116}$ | $7.5 \times 10^{163}$ | $1.6 \times 10^{204}$ |

In each edge, there are two candidate operators in our $L$-chain space. Each node receives at most $2L$ input feature maps. For the purpose of validity, each node has at least one input to be preserved. However, not every node may be used as an input node, and these nodes will be deleted, which means that the architectures are allowed to be very shallow. For a node $x_k (k > L)$, there are $2^{2L} - 1$ input possibilities because each of the $2L$ operators can be on or off. As a result, there are $(2^2 - 1) \times (2^4 - 1) \times \cdots \times (2^{2(L-1)} - 1) \times (2^{2L} - 1)^{N-L}$ combinations if there are $N$ nodes in one stage. There are 3 stages to be searched in our space, Fig. 2, with the node number of 18, 20 and 18 respectively. Therefore, if $L$ is 4, 6 and 8, there are about $1.8 \times 10^{116}$, $7.5 \times 10^{163}$ and $1.6 \times 10^{204}$ possible architectures respectively. The complexity comparison of popular spaces are shown in Tab. 1.

### 3.3 Failure Cases and Interleaved Connections

We first evaluate DARTS (Liu et al., 2019b) and GOLD-NAS (Bi et al., 2020) in the search space of $\mathbb{S}^{(4)}$. On the ImageNet-1k dataset, trained for 250 epochs for each network, DARTS and GOLD-NAS report top-1 accuracy of $74.7\%$ and $75.3\%$, respectively, and the FLOPs of both networks are close to $600\text{M}$ (*i.e.*, the mobile setting), Tab. 2. In comparison, the simple chain-styled architecture (each node is only connected to its direct precursor) achieves $74.9\%$ with merely $520\text{M}$ FLOPs.

More interestingly, if we only preserve one input for each node, DARTS and GOLD-NAS report completely failed results of $70.2\%$ and $71.2\%$ (trained for 100 epochs), which are even much worse than preserve one input randomly, Tab. 3.

We investigate the searched architectures, and find that both DARTS and GOLD-NAS tend to find long-distance connections. This decreases the depth of the searched architectures, however, empirically, deeper networks often lead to better performance. This drives us to rethink the reason that the algorithm gets confused. For this purpose, we randomly choose an intermediate layer from the super-network and the algorithm needs to determine the preference among its precursors.

We start from the simplest situation that all network connections are frozen (the architectural weights are fixed) besides the candidate connections. We show the trend of three (pre-1, pre-3, pre-6) connections in Figure 1. One can observe that the pre-1 connection overwhelms other two connections, aligning with our expectation that a deeper network performs better. However, when we insert one connection that lies between these candidates, we observe that the advantage of the pre-1 connection largely shrinks, and a long training procedure is required for it to take the lead. The situation continues deteriorating if we insert more connections into this region. When two or more connections are added, the algorithm cannot guarantee to promote the pre-1 connection and, sometimes, the ranking of the three connections is totally reversed.

The above results inspire us that two connections are easily interfered by each other if the covering regions (*i.e.*, the interval between both ends) overlap. Hereafter, we name such pair of connections **interleaved connections**. Mathematically, denote a connection that links the a-th and b-th nodes as $(a, b)$, where $b-a \geq 2$. Overlook the candidate connections of the activated nodes, two connections, $(a, b)$ and $(a', b')$, interleave if and only if there exists at least one integer $d$ that satisfies $a < d + 1/2 < b$ and $a' < d + 1/2 < b'$. Intuitively, for a real number $x$, if $\{x|a < x < b\} \cap \{x|a' < x < b'\} \neq \emptyset$ is satisfied, the interleaved connection occurs.

As a side comment, the DARTS space is also impacted by the interleaved connections, which partly cause the degradation problem observed in prior work (Liang et al., 2019; Chen et al., 2019; Shu et al., 2020). However, since the search space is cell-based, the degradation does not cause dramatic accuracy drop. In the experiments, we show that our solution, elaborated in the next part, generalizes well to the DARTS space.

### 3.4 Interleaving-Free Sampling

Following the above analysis, the key to design the search algorithm is to avoid interleaved connections, meanwhile ensuring that all connections can be considered. For this purpose, we propose IF-NAS, a sampling-based approach that each time optimizes a interleaving-free sub-super-network from the super-network.

This is implemented by partitioning the layers into $L$ groups, $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L$, according to their indices modulo $L$. In other words, the distance between any consecutive layers that belong to the same group is exactly $L$. When any group $\mathcal{G}_\lambda$ is chosen, we obtain an interleaving-free sub-super-network by preserving (i) the main backbone (all pre-1 connections) and (ii) all connections that end at any layer in $\mathcal{G}_\lambda$. We denote the sub-super-network by $\mathbb{S}_\lambda^{(L)}$ for $\lambda = 1, 2, \ldots, L$. Correspondingly, the optimization goal can be written as:

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\omega}} \mathbb{E}_{(\mathbf{x}, y^\star) \in \mathcal{D}} \left| y^\star - f\left(\mathbf{x} \mid \mathbb{S}_\lambda^{(L)}\right) \right|. \tag{2}$$

A straightforward search procedure repeats the loop of $L$ sub-super-networks, as shown in Figure 3. Training each sub-super-network is an approximation of optimizing the entire super-network, $\mathbb{S}^{(L)}$, but, since the sub-super-network are fixed, the co-occurring connections may gain unexpected advantages over other ones. To avoid it, we add a warm-up step to the beginning of each loop, in which the entire super-network is trained (all connections are activated, but the architectural parameters are not updated). Throughout the remainder of the loop, we train the $L$ sub-super-networks orderly. We set the length of each step to be $100$ iterations (*i.e.*, mini-batches). Note that this number shall not be too large, otherwise the gap between the sampled and non-sampled connections will increase and harm the search performance.

### 3.5 DISCRETIZATION AND PRUNING

The last step is to determine the final architecture. In a complicated search space, this is not simple as it seems, because many connections may have moderate weights (*i.e.*, not close to 0 or 1). For such a connection, either pruning it or promoting it can bring significant perturbation on the super-network. Therefore, we follow the idea of prior works (Chu et al., 2020; Bi et al., 2020; Tian et al., 2021) to first perform a discretization procedure to push the weights towards either 0 or 1, after which pruning is much safer. We introduce two kinds of architectural parameters, the connectivity of edges are determined by $\boldsymbol{\beta}$ and operators are determined by $\boldsymbol{\alpha}$.

The key of discretization is to add a regularization term to Eqn. (2). The term penalizes the moderate weights of edges and operators, represented by $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ respectively. Once these weights are less than $0.01$, the corresponding edges or operators will be pruned off. The regularization term is computed by:

$$\begin{aligned}\mathcal{R}(\alpha,\beta) = \mu_1 \cdot \sum_{j\leq N} \sum_{0\leq j-L\leq i<j} \ln(1+g(\beta_{i,j})/\overline{g(\beta_{i,j})}) \\ + \mu_2 \cdot \sum_{j\leq N} \sum_{0\leq j-L\leq i<j} \sum_{o\in\mathcal{O}} \ln(1+g(\alpha^o_{i,j}|\beta_{i,j})/\overline{g(\alpha^o_{i,j}|\beta_{i,j})}),\end{aligned} \tag{3}$$

where $N$ is the number of layers, and $g(\cdot)$ is the activate function $sigmoid$. $g(\alpha|\beta)$ means to only include operators on $\boldsymbol{\beta}$ that have not been pruned off. $\overline{g(\cdot)}$ is the average of $g(\cdot)$. $\mathcal{R}(\alpha,\beta)$ is multiplied by a factor of $\mu$ and added to the cross-entropy loss, so that optimizing the overall loss will not only improve the recognition accuracy of the super-network, but also push the weights of all connections towards 0 or 1.

When the weight of a connection is sufficiently small, we prune it permanently from the super-network. This merely impacts the super-network itself, but the computation of $\overline{g(\cdot)}$ changes and will push the next weak operator to 0. This process iterates until the complexity (*e.g.*, FLOPs) of the super-network achieves the lower-bound.

## 4 EXPERIMENTS

In this section, we first introduce experimental details and implementation details. Next, we introduce the results and analysis of IF-NAS and other advanced methods in our $L$-chain space under different settings. Finally, we compare $L$-chain spaces with 2 cell-based micro spaces using IF-NAS and other 2 search strategies.

### 4.1 IMPLEMENTATION DETAILS

**The Dataset.** We use the large-scale ImageNet dataset (ILSVRC2012) to evaluate the models. ImageNet contains 1,000 object categories, which consists of 1.3M training images and 50K validation images. The images are almost equally distributed over all classes. Unless specified, we apply the mobile setting, in which the input image is set to $224 \times 224$ and the number of multiply-add operations (MAdds) does not exceed 600M. We randomly sample 100 classes from the original ImageNet dataset to perform studying and analysis experiments, ImageNet-100 for short.

**The Search Settings.** Before searching, IF-NAS warm-ups the super-network for 20 epochs, with only the super-network weights updated and the architectural parameters frozen. Then we start to update $\beta$ firstly, and after 10 epochs, start to update $\alpha$. Super-network weights and architectural parameters are both optimized by the SGD optimizer. We gradually prune off weak edges and operators with threshold of $0.01$ until the MAdds of the retained architecture meets the mobile setting, *i.e.*, 600M.

**The Training Settings.** We evaluate the searched architectures following the setting of PC-DARTS (Xu et al., 2020). Each searched architecture is trained from scratch with a batch size of 1024 on 8 Tesla V100 GPUs. A SGD optimizer is used with an initial learning rate of 0.5, weight decay ratio of $3 \times 10^{-5}$, and a momentum of 0.9. Other common techniques including label smoothing, auxiliary loss, and learning rate warm-up for the first 5 epochs are also applied.

### 4.2 RESULTS ON $L$-CHAIN SEARCH SPACE

We study our method IF-NAS on the proposed macro $L$-chain search space, and compare it with three other popular DNAS frameworks, including DARTS (Liu et al., 2019b), PC-DARTS (Xu et al., 2020) and GOLD-NAS (Bi et al., 2020). For a fair comparison, the hyper-parameters are kept the same for all the studied methods. Without pruning gradually, DARTS and PC-DARTS derive the final architectures by removing weak edges and operators after searching until they meet the requirements of mobile setting. All the searched architectures are re-trained from scratch on ImageNet-1k for 250 epochs.

Table 2: Results of popular methods on our enlarged complicated space: comparison of classification test error (%) trained on ImageNet-1k for 250 epochs under the mobile setting.

| Setting | Method | Test Err. (%) | | Params | ×+ |
|---|---|---|---|---|---|
| | | top-1 | top-5 | (M) | (M) |
| $L$ = **4** | DARTS (Liu et al., 2019b) | 25.3 | 8.1 | 5.1 | 589 |
| | PC-DARTS (Xu et al., 2020) | 24.7 | 7.5 | 5.3 | 593 |
| | GOLD-NAS (Bi et al., 2020) | 24.7 | 7.5 | 5.5 | 591 |
| | IF-NAS | **24.3** | **7.4** | 5.3 | 592 |
| $L$ = **6** | DARTS (Liu et al., 2019b) | 25.7 | 8.1 | 5.1 | 587 |
| | PC-DARTS (Xu et al., 2020) | 24.9 | 7.7 | 5.2 | 586 |
| | GOLD-NAS (Bi et al., 2020) | 25.0 | 7.8 | 5.4 | 596 |
| | IF-NAS | **24.4** | **7.4** | 5.2 | 598 |
| $L$ = **8** | DARTS (Liu et al., 2019b) | 25.9 | 8.2 | 5.2 | 591 |
| | PC-DARTS (Xu et al., 2020) | 25.1 | 7.9 | 5.2 | 582 |
| | GOLD-NAS (Bi et al., 2020) | 25.1 | 7.9 | 5.3 | 592 |
| | IF-NAS | **24.3** | **7.3** | 5.4 | 594 |

We perform three sets of experiments by setting $L$, which indicates the precursors number of each layer can be connected, to 4, 6 and 8 respectively, and show the results in Tab. 2. When $L$ is 4, DARTS gets the worst performance as expected. Compared with DARTS, PC-DARTS and GOLD-NAS get better results of $24.7\%$ and $24.7\%$ with similar MAdds. Our IF-NAS gets the best result of $24.3\%$ with equivalent Params and MAdds. This proves that IF-NAS can work better in such a enlarged complicated space compared to the methods.

When $L$ is 6, DARTS achieves the result of $25.7\%$. PC-DARTS and GOLD-NAS get results of $24.9\%$ and $25.0\%$ respectively. And our IF-NAS gets the best result of $24.4\%$ with similar Params and MAdds. Note that the results of the compared 3 frameworks get worse when $L$ increases to 6 compared to $L$ as 4. However, IF-NAS nearly maintains the performance. When $L$ increases, more interleaved connections occur, and bring more interference to the search process. The compared 3 methods, which do not suppress interleaved connections during search, are more severely effected and achieve worse results. However, IF-NAS samples interleaving-free sub-networks and updates the architectural parameters more accurately. This further confirms that our IF-NAS has great potential to handle complicated spaces.

When set $L$ to 8, IF-NAS still achieves a promising result of $24.3\%$, which is comparable to the former two settings. The compared ones get similar or weaker results. Due to the increase in complexity, the performances of PC-DARTS and GOLD-NAS continue to decrease. But the MAdds of these architectures are close to 600M, which prevents performance from declining. If we do not derive architectures with MAdds and preserve one input for each node, the results are shown in Tab. 3. Interestingly, without the protection of keeping so many MAdds, DARTS and GOLD-NAS show completely failed results. When $L$=4, the results of DARTS and GOLD-NAS are $29.8\%$ and $28.8\%$, which are even much worse than the randomly generated architectures. When $L$ increases to 8, the performances of DARTS and GOLD-NAS further degrade. The architectures searched without suppressing interleaved connections only preserve very few nodes, which damages the final performance dramatically. On the contrary, IF-NAS still achieves comparable results even without any MAdss restriction.

Table 3: Results of reserving one input for each node. all architectures are trained on ImageNet-1k for 100 epochs.

| Setting | random | DARTS | GOLD-NAS | IF-NAS |
|---|---|---|---|---|
| $L$ = **4** | $27.6 \pm 0.3$ | 29.8 | 28.8 | 26.7 |
| $L$ = **6** | $28.0 \pm 0.3$ | 30.2 | 29.0 | 26.9 |
| $L$ = **8** | $28.9 \pm 0.4$ | 31.6 | 30.8 | 27.3 |

The above experiments imply interleaving connections cause non-negligible impacts to DNAS methods when the search space grows larger and more complicated. IF-NAS is able to handle this challenge by the proposed interleaving-free sampling.

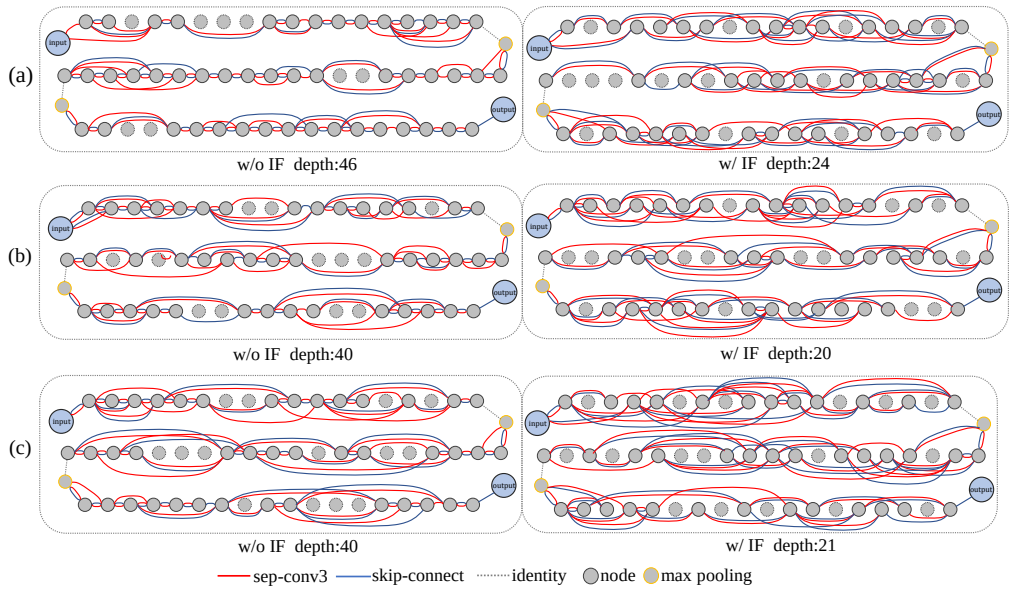


Figure 4: Architectures searched with (left) and without (right) IF. (a): results on 4-chain space. (b): results on 6-chain space. (c): results on 8-chain space. The reserved connections with IF are mainly serial pattern, which guarantees the depths of architectures. The reserved connections without IF are mainly parallel pattern, resulting in a short longest-path at each stage, that is, the depths of architectures are shallow.

### 4.3 IF Sampling Effectiveness on $L$-chain Space

In this part, we design a comparative experiment to verify the effectiveness of IF sampling. We search for 4 times independently for each setting, and the results are shown in Tab. 4. The **Depth** in Tab. 4 refers to the depth summation of 3 stages, which represents the longest path length of the network. And all architectures are train on ImageNet-1k for 100 epochs.

We find the accuracy of the architectures searched without IF ("w/o IF") sampling is much worse than that with IF ("w/ IF") sampling by a margin of $0.7\%$ with similar Params. It is worth noting that the depth of w/o IF is much smaller than w/ IF, because the interleaved connections produce serious interference to the search process. This verifies that our IF-NAS can effectively handle interleaved connections and guarantee the search performance.

Table 4: Results of exploring the effectiveness of IF sampling on the $L$-chain space. All the architectures are trained on ImageNet-1k for 100 epochs.

| **Setting** | **Depth** | **Params (M)** | **Err. (%)** |
|---|---|---|---|
| w/o IF | $22.5 \pm 2.0$ | $5.2 \pm 0.2$ | $26.7 \pm 0.3$ |
| w/ IF | $43 \pm 3$ | $5.3 \pm 0.1$ | $26.0 \pm 0.3$ |

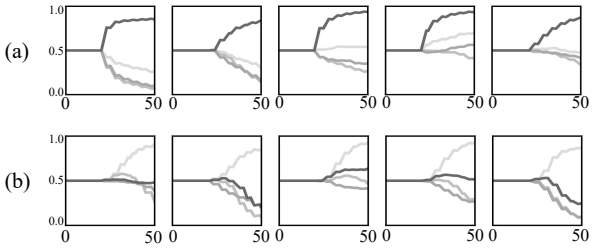


Figure 5: (a): The weight changing curve of candidate inputs with IF. (b): The weight changing curve of candidate inputs without IF. The darker the color, the closer the corresponding candidate inputs are to the activated node.

The visualizations of searched architectures are shown in Fig. 4. We can find that w/ IF preserves many serial pattern connections and w/o IF mainly preserves parallel connections, as a result, the architectures of w/ IF is much deeper than w/o IF. Fig. 5 shows some weight change curves of w/ IF and w/o IF. One can see that w/ IF can select more closer inputs, which derives deeper architectures. w/o IF prefers farther nodes, which usually leads to parallel connections and discards more nodes.

### 4.4 Effects of Interleaved Connections

We allow various numbers (1/2/3) of interleaved connections to be sampled during search to understand the effects they may bring. The searched architectures are trained on ImageNet-100 for 100 epochs and the results are shown in Tab. 5. We find with more interleaved connections added, the performance degrades more. This shows more interleaved connections cause greater impact on the search. It implies that Interleaving-Free is necessary in such a flexible space. Interestingly, the performance drops greatly even through one interleaved connection is sampled during searching, which demonstrates that as long as the interleaved connection is involved, interference is generated.

Table 5: Results of retaining architectures searched with different number of interleaved connections (IC) involved.

| IC Number | Depth | Params (M) | Err. (%) |
|---|---|---|---|
| 0 | $41.5 \pm 2.5$ | $5.0 \pm 0.1$ | $19.1 \pm 0.2$ |
| 1 | $26.0 \pm 3.0$ | $4.9 \pm 0.3$ | $19.7 \pm 0.3$ |
| 2 | $24.5 \pm 2.5$ | $5.0 \pm 0.3$ | $19.9 \pm 0.4$ |
| 3 | $24.0 \pm 3.0$ | $5.0 \pm 0.2$ | $20.0 \pm 0.3$ |

### 4.5 Comparison with Cell-based Micro Spaces

In this section, we test IF-NAS in two micro search spaces, DARTS space referred to as $space_D$ and GOLD-NAS space referred to as $space_G$. The two spaces adopt the same $3 \times 3$ separable convolution and skip-connection operators as ours, thus that the comparison is fair. The results are shown in Fig. 6. The horizontal axis denotes the space complexity (the exponential base of 10) and the vertical axis denotes the accuracy trained on ImageNet-1k for 100 epochs. The bubble sizes imply the complexity of spaces. The results on $space_D$ and $space_G$ are close among 3 strategies (even with random search), which implies the 2 spaces are inflexible and have a protection mechanism (the cell design) to help the search methods. The $L$-chain spaces are more flexible and difficult so that the gaps between IF-NAS and others in $L$-chain spaces are much larger. Moreover, the larger the spaces, the worse the performances of DNAS and random search. However, our IF-NAS provides all best results in the 5 spaces, which shows that IF-NAS are robust and generalize well to micro spaces.

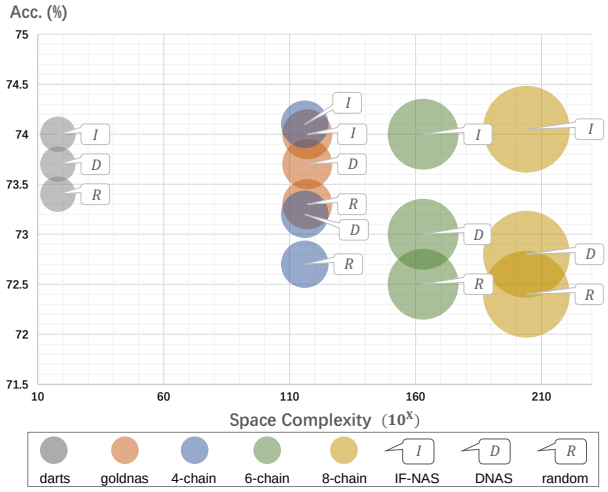


Figure 6: Results on different spaces using 3 search strategies. DNAS indicates the differential-based search without IF. The random indicates random search.

## 5 Conclusions

This work investigates neural architecture search and extends the design of search space towards higher flexibility. This is done by adding a large number of long-range connections to the super-network. The long-range connections raise new challenges to the search algorithm, and we locate the problem to be the interference introduced by interleaved connections during search. We propose a simple yet effective solution named interleaving-free sampling based on this observation, which makes a schedule to sample sub-networks and thus guarantees that interleaved connections do not occur. Experiments in the complicated search space show the effectiveness of our approach, IF-NAS.

Searching in a more complicated space has been of critical importance for the development of NAS. Our research delivers the message that new properties/challenges of NAS emerge when the search space is augmented. We advocate for more efforts in this direction to improve the ability of the search algorithms.

## ETHICS STATEMENT

The proposed methods in this paper is used to design neural architectures automatically. This helps to explore superior macro architectures in the very complicated search spaces, which could brings more potential architectures but also brings bigger difficulties for NAS and activates researchers to explore the challenging problem.

To the best of our knowledge, our research does not raise any concerns in ethics.

## REPRODUCIBILITY STATEMENT

The reproducibility of this paper is guaranteed by two factors. Firstly, we will release the code. Secondly, the proposed method is effective but simple. IF-NAS samples sub-networks iteratively and each sub-network is interleaving-free, which guarantee the interference is reduced. The searched architectures are deeper, which are generally considered better.

## REFERENCES

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018.

Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. GOLD-NAS: gradual, one-level, differentiable. *CoRR*, abs/2007.03331, 2020. URL https://arxiv.org/abs/2007.03331.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *CVPR*, 2020a.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Kangjian Peng, Yuan Li, Wenyu Liu, and Xinggang Wang. Fna++: Fast network adaptation via parameter remapping and architecture search. *TPAMI*, 2020b.

Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. Hit-detector: Hierarchical trinity architecture search for object detection. In *CVPR*, 2020a.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020b.

GAndrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, 2019.

Guilin Li, Xing Zhang, Zitong Wang, Zhenguo Li, and Tong Zhang. Stacnas: Towards stable and consistent optimization for differentiable neural architecture search. *CoRR*, abs/1909.11926, 2019.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. DARTS+: improved differentiable architecture search with early stopping. *CoRR*, abs/1909.06035, 2019. URL http://arxiv.org/abs/1909.06035.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Fei-Fei Li. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019a.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019b.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *ICLR*, 2020.

Yunjie Tian, Chang Liu, Lingxi Xie, Jianbin Jiao, and Qixiang Ye. Discretization-aware architecture search. *Pattern Recognit.*, 120:108186, 2021.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, 2019.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

# A Appendix

## A.1 More Examples of Interleaved Connections

In Section 3.3, we show failure cases from the simplest situation. In this part we perform more experiments to show the influence of interleaved connections and the weight change curves are shown in Figure 7. In Fig. 7 (a), we run 4 times independently under interleaving-free setting and the nearest candidate inputs have a dominant weight at this time. In Fig. 7 (b)-(d), We run 4 times independently and each time we add 1-3 different interference connections. One can see that the more interleaved connections are added, the harder it it the nearest candidate input is picked.

## A.2 Degradation in Micro Space

We run DARTS, a representative DNAS framework, 40 times independently under different settings. The normal cell depth of the searched architectures are organized in Fig. 9. Most architectures in Fig. 9 bias for shallow depth, which has been observed by some former works. Especially, the ratio of depth 1 and depth 2 exceeds $60\%$.

In Fig. 8, we sample 8 operators randomly for 10 times, and each time 4 normal cells with different depths and same parameters can be constructed by only changing the topology. In addition, a reduction cell is randomly sampled for all normal cells. So that 40 architectures are generated. We train these architectures in CIFAR100 and ImageNet-small from scratch for 100 epochs. ImageNet-small dataset is generated by sampling 100 images each class randomly from ImageNet-1k. The results, shown in Fig. 8, show that shallow architectures are nothing to do with their performance. On the contrary, the shallow architectures are not better than the deeper ones, especially for datasets with more categories. This also shows that degradation is detrimental to performance even in micro space.

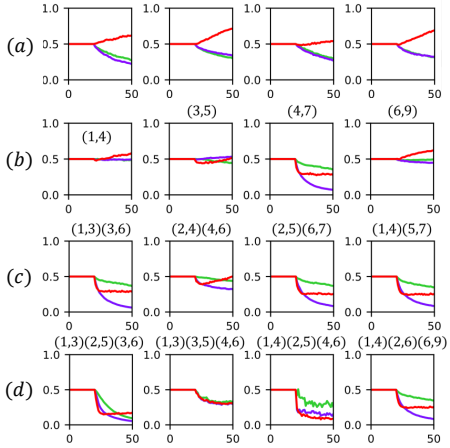


Figure 7: More examples of adding different numbers of interleaved connections to impact the NAS algorithm. (a) is under the interleaving-free setting and 1-3 interleaved connections are added for (b)-(d).

Degradation is a common problem in DNAS methods, which has been observed by some previous works. However, due to the design limitation of micro space, this problem has little impact and has not been further studied. But in the newly proposed space with more flexibility and potential, the impact of interleaved connections becomes large.

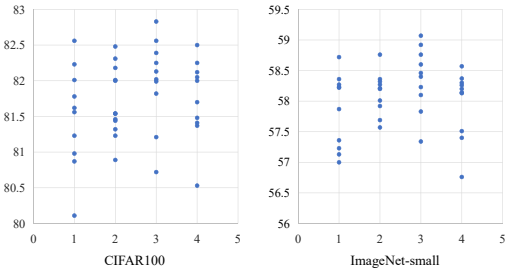


Figure 8: Results of randomly sampled architectures on CIFAR100 and ImageNet-small. The horizontal axis is the depth of the normal cell, and the vertical axis is the accuracy (%).

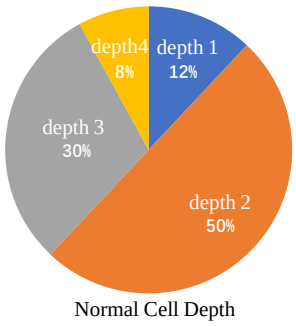


Figure 9: Normal cell depth ratio of DARTS.

### A.3 Analysis on the Search Space

In our $L$-chain search space, 3 stages are to be searched. There are 18, 20 and 18 nodes for these 3 stages respectively and 56 nodes in total, which is the same with DARTS and GOLD-NAS space. If needed, we can also easily increase the number of nodes in each stage, which also means greater flexibility and difficulty. There is no output node in our $L$-chain like the micro space, which integrates the concat features of the 4 intermediate nodes through $1 \times 1$ convolution, so the parameters and MAdds can be keep indistinguishable although with a larger channel number.

In Section 3.2.2, we compare our $L$-chain space with two popular spaces. Since there is no cell design, $L$-chain space contains much more possibilities. The size of our 4-chain space is comparable to GOLD-NAS, and 6-chain and 8-chainspace contain much more possibilities. Even though 4-chain contains comparable even less architectures than GOLD-NAS, 4-chain space is much more difficult than GOLD-NAS because it is more flexible and allows extremely degraded architectures.

### A.4 Visualization of architectures by reserving one input

We show some architectures derived by only reserving one input, Fig. 10 and Fig. 11. In Fig. 10, the searched architectures reserve many near candidate inputs, so architectures are still deep. On the

Table 6: Cell depths and results of IF-NAS in $space_D$ and $space_G$. All architectures are trained on ImageNet-100 for 100 epochs.

| Space | methods | Depth | Params (M) | Err. (%) |
|---|---|---|---|---|
| $space_D$ | PC-DARTS | 2/2/3/3 | $4.8 \pm 0.5$ | $19.5 \pm 0.6$ |
| | IF-NAS | 3/3/3/4 | $4.8 \pm 0.4$ | $19.3 \pm 0.5$ |
| $space_G$ | GOLD-NAS | 2.5/2.7/2.5/2.6 | $5.3 \pm 0.6$ | $19.2 \pm 0.4$ |
| | IF-NAS | 3.1/3.3/2.8/3.1 | $5.2 \pm 0.8$ | $19.0 \pm 0.4$ |

contrary, in Fig. 11 without IF, only a few nodes are kept, resulting in extreme degradation of the architectures.

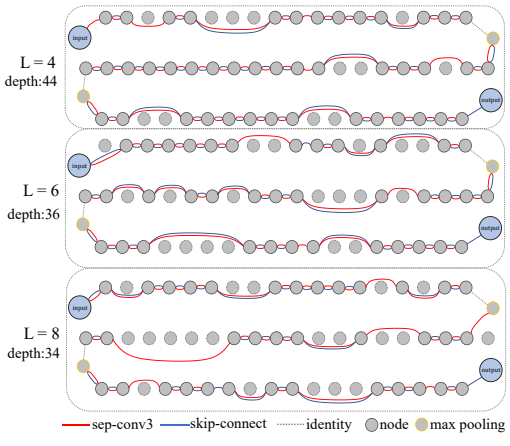


Figure 10: Architectures searched with IF by only reserving one input.

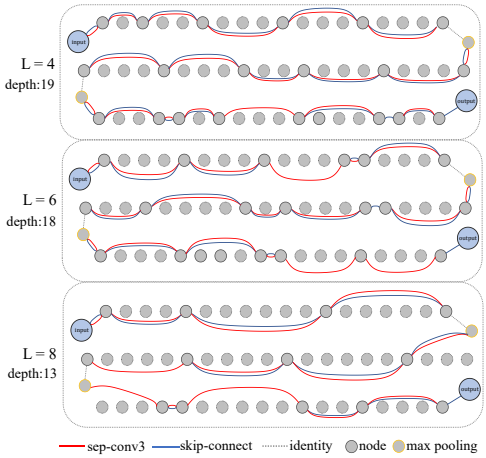


Figure 11: Architectures searched without IF by only reserving one input.

### A.5 Searched Cell Depths in Micro Spaces

In this section, we test our IF-NAS in terms of the searched architecture depths in two micro search spaces from DARTS as $space_D$ and GOLD-NAS as $space_G$. The results are shown in Tab. 6. In $space_D$, we run PC-DARTS and IF-NAS 4 times independently. The depths of the searched normal cells by PC-DARTS are 2/2/3/3 respectively. By training the model for 100 epochs on ImageNet-100, PC-DARTS achieves a test error of $19.5\% \pm 0.6$ with $4.8 \pm 0.5$M Params. The depth of the searched normal cells by IF-NAS are 3/3/3/4 respectively, deeper than PC-DARTS. A better test error of $19.3\% \pm 0.5$ is achieved with similar Params. Similarly in $space_G$, we run GOLD-NAS and IF-NAS 4 times respectively, and count the average depth of all cells of each architecture. The average depth of GOLD-NAS are 2.5/2.7/2.5/2.6 respectively, which are shallower than IF-NAS with average depths of 3.1/3.3/2.8/3.1. Moreover, better results are obtained by IF-NAS with even fewer Params, compared with GOLD-NAS. The above experiments verify that our IF-NAS can still perform well in micro spaces, and the architecture depths are larger with interleaved connections suppressed.