EXPLICIT LEARNING TOPOLOGY FOR DIFFERENTIABLE NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable neural architecture search (NAS) has gained much success in discovering more flexible and diverse cell types. Current methods couple the operations and topology during search, and simply derive optimal topology by a hand-craft rule. However, topology also matters for neural architectures since it controls the interactions between features of operations. In this paper, we highlight the topology learning in differentiable NAS, and propose an explicit topology modeling method, named TopoNAS, to directly decouple the operation selection and topology during search. Concretely, we introduce a set of topological variables and a combinatorial probabilistic distribution to explicitly indicate the target topology. Besides, we also leverage a passive-aggressive regularization to suppress invalid topology within supernet. Our introduced topological variables can be jointly learned with operation variables and supernet weights, and apply to various DARTS variants. Extensive experiments on CIFAR-10 and ImageNet validate the effectiveness of our proposed TopoNAS. The results show that TopoNAS does enable to search cells with more diverse and complex topology, and boost the performance significantly. For example, TopoNAS can improve DARTS by 0.16% accuracy on CIFAR-10 dataset with 40% parameters reduced or 0.35% with similar parameters.

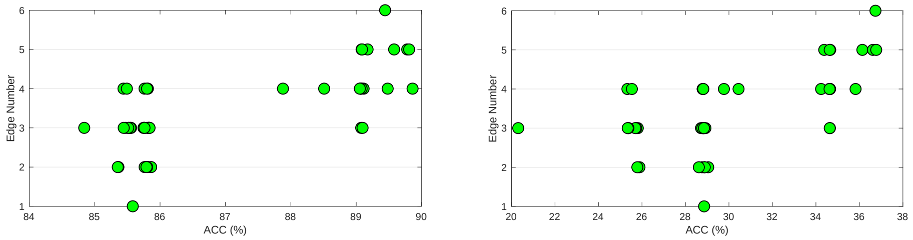
1 INTRODUCTION

Targeting at slipping the leash of human empirical limitations and liberating the manual efforts in designing networks, neural architecture search (NAS) emerges as a burgeoning tool to automatically seek promising network architectures in a data-driven manner. To accomplish the architecture search, early literatures mainly adopt sheer reinforcement learning (RL) (Baker et al., 2017; Zoph & Le, 2017) or evolutionary algorithms (Real et al., 2019). Nevertheless, it often involves hundreds of GPUs for computation and takes a large volume of GPU hours to finish the searching.

For sake of searching efficiency, pioneer work NASNet (Zoph et al., 2018) proposed to search on a cell level, where the searched cells can be stacked to develop task-specific networks. (Pham et al., 2018; Bender et al., 2018) leverage weight-sharing scheme, and amortize the cost of training for each candidate architecture. Recently, DARTS (Liu et al.) makes the most of both sides, and proposes a differentiable NAS variant. In DARTS, a one-shot over-parameterized supernet is regarded as a full graph, from which all candidate architectures are derived as its sub-graphs. Besides, a set of operation variables are introduced to indicate the importance of different operations, and the optimal architecture corresponds to that with largest importance.

Due to the simplicity and searching efficiency, many follow-up works have been devoted to further boosting its performance in various aspects, such as MiLeNAS (He et al., 2020) in optimization, ProxylessNAS (Cai et al.) and PC-DARTS (Xu et al.) in memory consumption, FBNet (Wu et al., 2019) and SNAS (Xie et al., 2019b) in stochastic modification, and P-DARTS (Chen et al., 2019) and Robust-DARTS (Zela et al.) in reducing the searching gap.

We notice that, to search for an architecture (sub-graph) from the supernet (full-graph), both graph topology and edge types (*i.e.*, operations) matter. However, current differentiable methods mainly focus on the operation selection, and overlook the learning of topology during searching. Although some works (Liu et al.; Xu et al.; He et al., 2020) introduce a special `zero` operation for cutting edges to take account of topology to some extent, the operation selection and topology are still



(a) CIFAR-10

(b) ImageNet-16

Figure 1: Average accuracies of each topology in NAS-bench-201 (Dong & Yang) among different number of edges on CIFAR-10 (left) and ImageNet (right). Details can be found in Appendix A.4.

severely *coupled*, *i.e.*, modeling of topology and operation selection are involved in the introduced operation variables simultaneously. And the topology is usually determined by a hand-crafted rule that keeps two edges for each node with the highest operation importance. Nevertheless, even given fixed operations for each egde, the optimal topology does not necessarily correspond to this naive and heuristic practice. As shown in Figure 1, average accuracies of each topology in NAS-Bench-201 (Dong & Yang) for different number of edges scatter in a wide range. So the ground-truth performance of different topological architectures can be fairly diverse, implying sub-optimal results are usually expected by current methods. These inspires us that we should highlight the learning of topology in differentiable NAS.

Recent works (Xie et al., 2019a; Wortsman et al., 2019) also get down to investigating the importance of graph topology in neural networks. RandWire (Xie et al., 2019a) indicates that randomly wired neural networks generated by random graph algorithms can achieve competitive performance to the manually designed architectures; (Wortsman et al., 2019) proposes a method of Discovering Neural Wirings (DNW) to joint train network and its fine-grained wiring of channels. However, it merely investigates on the channel dimension with fixed network architecture, and does not apply to differentiable NAS methods.

In this paper, we propose explicit learning topology (TopoNAS) for differentiable NAS. Concretely, we decouple the modeling of operation selection and topology during search, and introduce a set of topological variables to indicate the topology learning within the supernet. Instead of modeling each edge individually, we use the topological variables to model a combinatorial probabilistic distribution of all kinds of edge pairs, then the optimal topology corresponds to the edge pair with the largest topology score. By dint of merging the combinatorial probabilities as a factor, almost no additional memory cost will be involved. Besides, our TopoNAS is capable of modeling sub-graphs with either fixed or arbitrary number of edges, which promotes the learned topology to be more diverse.

Our topological variables can be applied to various differentiable NAS methods, and optimized jointly with operation variables and their weights as bi-level DARTS (Liu et al.) or mixed-level MiLeNAS (He et al., 2020). In addition, to eliminate invalid topology during search, we propose a *passive-aggressive* regularization on the topological variables. Extensive experiments on the benchmark CIFAR-10 and ImageNet datasets validate the effectiveness of our proposed TopoNAS. And results show that it does enable to search architectures with more diverse and complex topology, and greatly improve the performance for both DARTS and MiLeNAS. For example, our TopoNAS can achieve 97.40% accuracy on CIFAR-10 dataset but has only 2.0M parameters compared to 97.24% accuracy with 3.3M parameters of DARTS. Meanwhile, with the similar amount of parameters, our TopoNAS obtains 97.59% accuracy with 3.5M parameters while the state-of-the-art MiLeNAS merely has 97.49% accuracy but with 3.9M parameters.

2 Revisiting Differentiable NAS

We first review the baseline differentiable NAS method DARTS (Liu et al.), which searchs for a computation cell as the building block of the final architecture. Mathematically, a cell can be considered as a directed acyclic graph (DAG) consisting of an ordered sequence of N nodes. Each node $\boldsymbol{x}_i$ is represented as a feature map, and each directed edge (i, j) between nodes indicates the candidate operations $o \in \mathbb{O}$, such as `max pooling`, `convolution` and `identity mapping`. Then the goal is to determine one operation o from $\mathbb{O}$ to connect each pair of nodes. DARTS relaxes

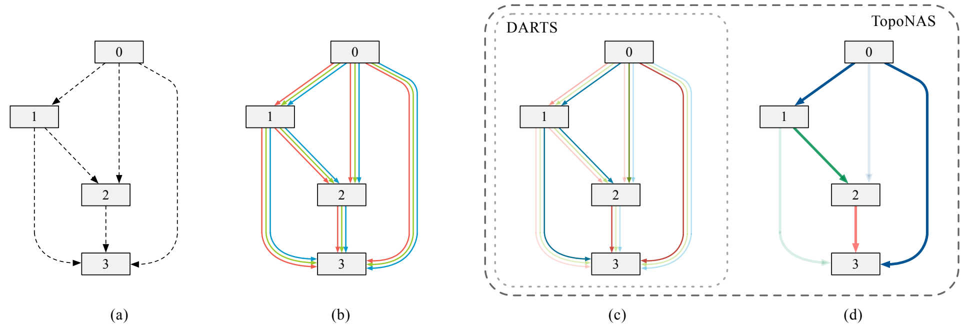


Figure 2: An overview of TopoNAS: (a) a cell represented by directed acyclic graph. The edges between nodes denote the operations to be learned. (b) Following DARTS (Liu et al.), the operation on each edge is replaced by a mixture of all candidate operations parameterized by operation variables $\boldsymbol{\alpha}$. (c) DARTS selects operation with the largest $\boldsymbol{\alpha}$ for each edge. (d) TopoNAS introduces additional topological variables $\boldsymbol{\beta}$ to explicitly learn topologies, which decouples operation selection (c) and topology learning (d).

this categorical operation selection into a soft and continuous selection using softmax probabilities with a set of variables $\boldsymbol{\alpha}_{i,j} \in \mathbb{R}^{|\mathbb{O}|}$ to indicate the operation importance,

$$o^{(i,j)}(\boldsymbol{x}_i) = \sum_{o \in \mathbb{O}} \frac{\exp(\alpha_{i,j}^{o})}{\sum_{o' \in \mathbb{O}} \exp(\alpha_{i,j}^{o'})} o(\boldsymbol{x}_i), \tag{1}$$

where $|\mathbb{O}|$ is the number of all candidate operations, $o(\boldsymbol{x}_i)$ is the result of applying operation o on $\boldsymbol{x}_i$, and $o^{(i,j)}(\boldsymbol{x}_i)$ means the summed feature maps from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$. Then the output of a node $\boldsymbol{x}_j$ is the sum of all feature maps from all its precedent nodes, with associated edges $\{(1,j), ..., (j-1,j)\}$, *i.e.*,

$$\boldsymbol{x}_j = \sum_{i<j} o^{i,j}(\boldsymbol{x}_i). \tag{2}$$

Moreover, each cell has two input nodes $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, which are the outputs of previous two cells, and the final output of an entire cell is formed by concatenating all intermediate nodes inside, *i.e.*, $\{\boldsymbol{x}_3, ..., \boldsymbol{x}_N\}$. The operation variables $\boldsymbol{\alpha}$ can be jointly trained with supernet weights by gradient-based optimizers (Liu et al.; He et al., 2020). After training, the optimal operation corresponds to the one with the maximum operation importance.

As for the derivation of topology, a hand-crafted rule is usually followed. Concretely, manually specify that only two input edges are active for each node $\boldsymbol{x}_j$ by selecting the edges with the top-2 largest operation importance in $\{o^{(1,j)}, ..., o^{(j-1,j)}\}$. However, this heuristic rule fails to obtain more diverse topology, and thus does not ensure the optimal one. We will elaborate our TopoNAS in the sequel, which aims to explicitly learn the topology of a cell.

3 TopoNAS: Towards Explicit Learning of Topology

In this section, we formally elaborate our explicit topology learning method TopoNAS, which enables to model topologies with either fixed or arbitrary edges, and applies to various DARTS variants.

3.1 Decoupling topology and operations

As previously illustrated, DARTS and its variants couple the operation selection and topology in their architecture modeling, but resort to a hand-crafted rule for deriving the topology. However, this practice usually induces sub-optimal results. Besides, observations on NAS-bench-201 (Dong & Yang) dataset in Figure 1 show that accuracies of architectures with different topologies can be fairly different. Both sides motivate us that we should highlight the learning of topology in differentiable NAS.

Note that DARTS stipulates that just two input edges are allocated to each node. Now we first investigate this *fixed-topology* case; nevertheless, instead of using the hand-crafted rule, we propose to learn automatically which two edges should be connected to each node. Based on the definition of DAG, the topology space $\mathcal{T}$ can be decomposed by each node and represented by their input edges, *i.e.*, $\mathcal{T} = \bigotimes_{j=1}^{N} \tau_j$, where $\bigotimes$ is the Cartesian product of all τ_j's and τ_j is the set of all input edges pairs for node $\boldsymbol{x}_j$, *i.e.*,

$$\tau_j = \{(1,2), ..., (1, j-1), ..., (2,3), ..., (j-2, j-1)\}, \tag{3}$$

and $|\tau_j| = \mathbb{C}_{j-1}^2$ since only two input edges are specified for each node. Besides, for the fixed input from the previous two cells, we denote them as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively, and we have $\tau_1 = \tau_2 = \emptyset$.

Since we have a clear and complete modeling over all possible topology $\mathcal{T}$ as Eq.(3), to determine the optimal topology, it is natural to introduce another set of topological variables $\boldsymbol{\beta} = \{\boldsymbol{\beta}_j\}_{j=1}^{N}$ with $|\boldsymbol{\beta}_j| = |\tau_j|$. $\boldsymbol{\beta}_j$ explicitly represents the soft importance of each topology (*i.e.*, input edge pairs). Then each node is rewritten as

$$\boldsymbol{x}_j = \sum_{(m,n)\in\tau_j} p_j^{(m,n)} (o^{m,j}(\boldsymbol{x}_m) + o^{n,j}(\boldsymbol{x}_n)), \tag{4}$$

with the combinatorial probability $p_j^{(m,n)}$ for selecting edges (m, j) and (n, j) as $\boldsymbol{x}_j$'s inputs, *i.e.*,

$$p_j^{(m,n)} = \frac{\exp(\beta_j^{(m,n)})}{\sum_{(m',n')\in\tau_j} \exp(\beta_j^{(m',n')})}, \tag{5}$$

where $\beta_j^{(m,n)}$ is the corresponding variable for $p_j^{(m,n)}$. Different from Eq.(2), Eq.(4) considers a combinatorial probabilistic distribution over all possible valid topologies, and the optimal topology simply refers to the one with the largest topological importance. Note that since the topology and operations are already decoupled, there is no need to involve a `zero` operation.

However, directly computing Eq.(4) will increase the memory consumption of feature maps since it needs to compute the summation of two feature maps $o^{m,j}(\boldsymbol{x}_m) + o^{n,j}(\boldsymbol{x}_n)$. Fortunately, this can be well addressed by merging the combinatorial probabilities associated with the same edges, and accordingly, Eq.(4) can be simplified as

$$\boldsymbol{x}_j = \sum_{i<j} s(i,j) \cdot o^{i,j}(\boldsymbol{x}_i), \quad \text{with} \quad s(i,j) = \sum_{k<i} p_j^{(k,i)} + \sum_{i<k<j} p_j^{(i,k)}, \tag{6}$$

where $s(i, j)$ is the merged combinatorial probability w.r.t. node $\boldsymbol{x}_i$. In this way, the memory cost of Eq.(6) is almost the same as DARTS, with only a learnable $s(i, j)$ added before edge accumulation.

3.1.1 Switching topology space with output edges

However, according to the experiment results in A.3.4, we empirically find the method mentioned above Eq.(4) and Eq.(6) performs poorly. This might result from that we examine the topology space from the perspective of input edges. In other words, we investigate which input edge pair should be selected to connect each node. However, as the *skip-dominant issue* (Xu et al.; Liang et al., 2019; Bi et al., 2019) in operation selection, precedent features of a node will dominate its current features during the training of supernet. Similarly, edges from the nodes that are more precedent will also tend to be dominant since they can be regarded as edge-level skip layers and benefits more from optimization. Note that a DAG can be both described by the input edges and output edges; selecting the output edges yet will alleviate the dominant issue on topology since features from the same node are compared. Then we propose to switch the topology space to the perspective of output edges.

Concretely, we examine the topology by selecting which edge pair should be the output for each node. By doing so, all output edges use the same nodes as input, and can be compared more fairly. In this way, the output nodes of each node $\boldsymbol{x}_i$ are from its posterior nodes $\boldsymbol{x}_{i+1}, ..., \boldsymbol{x}_N$, and we switch the topology space Eq.(3) to

$$\tilde{\tau}_i = \{(i+1, i+2), ..., (i+1, N), ..., (i+2, i+3), ..., (N-1, N)\}, \tag{7}$$

which is the set of output edge pairs of node $\boldsymbol{x}_i$ from all $N - i$ edges. Similar to the previous discussions, we also introduce a set of topological variables $\tilde{\boldsymbol{\beta}} = \{\tilde{\boldsymbol{\beta}}_i\}_{i=1}^{N}$ to model the soft importance for each edge pair, and $\tilde{p}_i^{(m,n)}$ denotes the combinatorial probability of choosing (i,m) and (i,n) as output edges of node $\boldsymbol{x}_i$ with the corresponding variable $\tilde{\beta}_i^{(m,n)}$ and $|\tilde{\boldsymbol{\beta}}_i| = |\tilde{\tau}_i| = \mathbb{C}_{N-i}^2$,

$$\tilde{p}_i^{(m,n)} = \frac{\exp(\tilde{\beta}_i^{(m,n)})}{\sum_{(m',n')\in\tilde{\tau}_i}\exp(\tilde{\beta}_i^{(m',n')})}. \tag{8}$$

Then a node is also represented as Eq.(6), but with different merged probabilities, *i.e.*,

$$\boldsymbol{x}_j = \sum_{i<j}\tilde{s}(i,j)\cdot o^{i,j}(\boldsymbol{x}_i), \quad \text{with} \quad \tilde{s}(i,j) = \sum_{i<k<j}\tilde{p}_i^{(k,j)} + \sum_{k>j}\tilde{p}_i^{(j,k)}. \tag{9}$$

Since the number of posterior nodes for each node is not equal (*e.g.*, node $\boldsymbol{x}_2$ has 4 posterior nodes with $N = 6$, but node $\boldsymbol{x}_3$ only has 3) , the merged probabilities $\tilde{s}(i,j)$ associated with each edge (i,j) have different magnitude of value. We scale it for more stable optimization, *i.e.*,

$$\tilde{s}(i,j) = \frac{\mathbb{C}_{N-i}^2}{\mathbb{C}_{N-i-1}^1}\Big(\sum_{i<k<j}\tilde{p}_i^{(k,j)} + \sum_{k>j}\tilde{p}_i^{(j,k)}\Big). \tag{10}$$

3.2 Generalizing to arbitrary topology space

Previous formulation considers a fixed topology space, since we only allocate two output edges for each node and model a combinatorial probability distribution over all possible output edge pairs. In fact, our modeling method TopoNAS can be naturally generalized to an arbitrary topology space, if we simply do not restrict the number of output edges for each node, and allow each node to connect its posterior nodes freely.

As a result, for arbitrary topology modeling, each node $\boldsymbol{x}_i$ can be connected with any number (at least 1) of its posterior nodes $\boldsymbol{x}_{i+1}, ..., \boldsymbol{x}_N$, and thus the amount of all possible combinatorial pairs $\hat{\tau}_i$ becomes

$$|\hat{\tau}_i| = \sum_{n=1}^{N-i}\mathbb{C}_{N-i}^n = 2^{N-i} - 1, \tag{11}$$

and each combinatorial pair can be uniquely defined by a binary code vector, *i.e.*, $\boldsymbol{b}_i = (b_i^{i+1}, b_i^{i+2}, ..., b_i^N)$ with $b_i^k \in \{0,1\}$, where $b_i^k = 1$ if edge (i,k) exists and $b_i^k = 0$ otherwise. Let $\boldsymbol{B}_i = \{\boldsymbol{b}_i^{(1)}, ..., \boldsymbol{b}_i^{(|\hat{\tau}_i|)}\}$ denotes the set of all valid binary code vectors for $\boldsymbol{x}_i$, then we also impose a combinatorial probability distribution to indicate the importance for each topology (*i.e.*, binary node vector), *i.e.*,

$$\hat{p}(\boldsymbol{b}_i) = \frac{\exp(\beta_i^{\boldsymbol{b}_i})}{\sum_{\boldsymbol{b}'_i\in\boldsymbol{B}_i}\exp(\beta_i^{\boldsymbol{b}'_i})}, \tag{12}$$

where we denote $\beta_i^{\boldsymbol{b}_i} \in \mathbb{R}$ as the introduced topology variable corresponding to the binary code $\boldsymbol{b}_i$. Then Eq.(9) evolves similarly, but also with a different merged probabilities for each node, *i.e.*,

$$\hat{s}(i,j) = \frac{2^{N-i} - 1}{2^{N-i-1}}\sum_{\boldsymbol{b}_i\in\boldsymbol{B}_i, b_i^j=1}\hat{p}(\boldsymbol{b}_i). \tag{13}$$

Remark. Note that PC-DARTS (Xu et al.) proposes an edge normalization technique, which also adopts a learnable variable for each edge. However, these variables are parameterized individually for each edge, and mainly for stabilizing the optimization. It still resorts to a hand-crafted rule for deriving topology, and does not support an explicit learning. As a comparison, our combinatorial probability can learn the edge number for each node, which is beyond the ability of the simple relaxation method in PC-DARTS.

3.3 Regularizing invalid topologies during search

Tough the topology can be explicitly modeled in our TopoNAS, there might be invalid topologies during search if a node is not connected with any of its precedent nodes. To suppress this trivial case, we introduce a *passive-aggressive* regularization during search, *i.e.*,

$$r(\boldsymbol{\beta}) = \sum_{j<N}\prod_{i<j}\Big(\big(\max_{\boldsymbol{b}_i\in\boldsymbol{B}_i}\hat{p}(\boldsymbol{b}_i) - \max_{\boldsymbol{b}_i\in\boldsymbol{B}_i, b_i^j=1}\hat{p}(\boldsymbol{b}_i)\big)/\max_{\boldsymbol{b}_i\in\boldsymbol{B}_i}\hat{p}(\boldsymbol{b}_i)\Big), \tag{14}$$

where $\max_{\boldsymbol{b}_i \in \boldsymbol{B}_i} \hat{p}(\boldsymbol{b}_i)$ denotes the max probability among all the output edges of node $\boldsymbol{x}_i$, and $\max_{\boldsymbol{b}_i \in \boldsymbol{B}_i, \boldsymbol{b}_i^j = 1} \hat{p}(\boldsymbol{b}_i)$ denotes the max probability associated to edge (i, j) .

Note that $r(\boldsymbol{\beta})$ only *aggressively* punishes the topology variables which predict invalid topology, while it *passively* does no harm to the optimization when the topology is valid. If edge (i, j) is chosen to be kept in the final architecture, it will hold that $\max_{\boldsymbol{b}_i \in \boldsymbol{B}_i} \hat{p}(\boldsymbol{b}_i) - \max_{\boldsymbol{b}_i \in \boldsymbol{B}_i, \boldsymbol{b}_i^j = 1} \hat{p}(\boldsymbol{b}_i) = 0$. The goal is to minimize $r(\boldsymbol{\beta})$, and $r(\boldsymbol{\beta}) = 0$ if the architecture is valid. This regularization can be integrated into the loss w.r.t. $\boldsymbol{\beta}$, *i.e.*,

$$\mathcal{L}_{val_{\boldsymbol{\beta}}}(\boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{L}_{task}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \cdot r(\boldsymbol{\beta}), \tag{15}$$

where $\mathcal{L}_{task}$ is task-specific loss, and we set λ as 10 in our experiments.

3.4 Case study: integrating TopoNAS in DARTS variants

Now we illustrate how our proposed topology modeling TopoNAS can be applied to DARTS variants for further boosting their performance. Details can be found in Appendix A.2.

DARTS. The original DARTS (Liu et al.) formulates the NAS into a bi-level optimization problem (Anandalingam & Friesz, 1992; Colson et al., 2007):

$$\min_{\boldsymbol{\alpha}} \ \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \ \text{ s.t. } \ \boldsymbol{w}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}), \tag{16}$$

where the operation variables $\boldsymbol{\alpha}$ and supernet weights $\boldsymbol{w}$ can be jointly optimized. By introducing additional topology variables $\boldsymbol{\beta}$, Eq.(16) then evolves into

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \ \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}), \boldsymbol{\alpha}, \boldsymbol{\beta}), \quad \text{ s.t. } \boldsymbol{w}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}). \tag{17}$$

Since TopoNAS involves extra topology optimization, for efficiency consideration, we adopt the first-order approximation to solve Eq.(17), which we find empirically suffices.

MiLeNAS. MiLeNAS (He et al., 2020) proposes a mixed-level reformulation of DARTS, which aims to optimize NAS more efficiently and reliably by mixing the training and validation loss together for architecture optimization. With the introduced topology variables $\boldsymbol{\beta}$, the objective is different from Eq.(17) and becomes as,

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \ \mathcal{L}_{tr}(\boldsymbol{w}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}), \boldsymbol{\alpha}, \boldsymbol{\beta}) + \lambda' \cdot \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}), \boldsymbol{\alpha}, \boldsymbol{\beta}), \tag{18}$$

which can also be solved efficiently using the first-order approximation as (He et al., 2020).

4 Experimental Results

4.1 Datasets and implementation details

We perform experiments on two benchmark datasets CIFAR-10 (Krizhevsky et al., 2014) and ImageNet (Deng et al., 2009). We search and evaluate convolution cells with fixed-edges topology or arbitrary-edges topology on CIFAR-10, and then transfer the searched cells to the ImageNet dataset.

For fair comparison, the operation space $\mathbb{O}$ in TopoNAS is similar to DARTS, which contains 3×3 max pooling, 3×3 average pooling, 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions and identity. However, we do not involve zero operation cause our method can explicitly learn topologies.

Fixed edge selection. For comparison with DARTS and other DARTS-based methods, we first conduct architecture search with fixed edges, which keeps the same edge number 8 as DARTS. Our cell consists of $N = 7$ nodes, the first and second nodes are input nodes, which are equal to the outputs of previous two cells, and the output node $\boldsymbol{x}_7$ is the concatenation of all 4 hidden nodes $\boldsymbol{x}_3$, $\boldsymbol{x}_4$, $\boldsymbol{x}_5$, $\boldsymbol{x}_6$. In order to fix the edge number, each node in $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ is set to have 2 output edges, besides, the nodes $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ have 1 output node.

Arbitrary edge selection. To further investigate the potential of topology learning, we perform architecture search with arbitrary edges, which has $N = 7$ or more nodes in each cell, and every nodes can choose to connect to any (at least 1) of its posterior nodes.

Detailed experimental setup can be found in Appendix.

Table 1: Search results on CIFAR-10 and comparison with state-of-the-art methods. Search cost is tested on a single NVIDIA GTX 1080 Ti GPU.

Methods	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	-	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	1800	RL
AmoebaNet-B +cutout (Real et al., 2019)	2.55±0.05	2.8	3150	evolution
ProxylessNAS+cutout (Cai et al.)	**2.08**	5.7	4.0	gradient-based
PC-DARTS + cutout (Xu et al.)	2.57±0.07	3.6	0.1	gradient-based
DARTS (1st order) + cutout (Liu et al.)	3.00±0.14	3.3	0.4	gradient-based
DARTS (2nd order) + cutout (Liu et al.)	2.76±0.09	3.3	4.0	gradient-based
MiLeNAS + cutout (He et al., 2020)	2.51±0.11	3.87	0.3	gradient-based
MiLeNAS + cutout (He et al., 2020)	2.76	2.09	0.3	gradient-based
TopoNAS-fixed-DARTS + cutout	2.72±0.12	**1.8**	0.6	gradient-based
TopoNAS-fixed-MiLe + cutout	2.68±0.09	**1.8**	0.6	gradient-based
TopoNAS-arbitrary-DARTS + cutout	2.67±0.14	1.9	0.6	gradient-based
TopoNAS-arbitrary-MiLe + cutout	2.60±0.06	2.0	0.6	gradient-based
TopoNAS-fixed-DARTS + cutout(C=48)	2.48±0.09	3.2	0.6	gradient-based
TopoNAS-fixed-MiLe + cutout(C=48)	2.54±0.07	3.2	0.6	gradient-based
TopoNAS-arbitrary-DARTS + cutout(C=48)	2.44±0.11	3.2	0.6	gradient-based
TopoNAS-arbitrary-MiLe + cutout(C=48)	2.41±0.14	3.5	0.6	gradient-based

4.2 Results on CIFAR-10 dataset

As previously discussed, our method can perform both fixed edges search and arbitrary edges search. We adopt fixed edges search and arbitrary edges search with node number $N = 7$ using first-order approximation in DARTS and MiLeNAS. Four obtained models are named *TopoNAS-fixed-DARTS*, *TopoNAS-arbitrary-DARTS*, *TopoNAS-fixed-MiLe* and *TopoNAS-arbitrary-MiLe*. The searched cells are visualized in A.6.

In the search stage, the supernet is built by stacking 8 cells with 2 reduction cells and 6 normal cells, the initial number of channels is set to 16. The training dataset is split into three sets $\mathcal{D}_{tr}$, $\mathcal{D}_{val_\alpha}$ and $\mathcal{D}_{val_\beta}$ with equal size. We simply choose the latest optimized networks after training 50 epochs with batch size 64 for deriving architectures.

Different from DARTS using 20 stacked cells to build evaluation networks, all our searched networks use the layer number of 12 for better performance, which will be detailed discussed in Sec. 4.4.1 . Our evaluation results on CIFAR-10 dataset compared with recent approaches are summarized in Table 1. TopoNAS can obtain competitive results but with much less parameters, for example, *TopoNAS-fixed-DARTS* can achieve 2.72% test error with only 1.8M parameters, which reduces $\sim 45\%$ parameters with higher accuracy compared to the original DARTS. That might because our method explicitly learns more suitable topologies for the operations and networks. By removing redundant edges, the obtained edges are more effective for the networks, thus we can decrease the layer number for less parameters and FLOPs. Besides, for fair comparision with other methods, we enlarge the initial channel number C of network from 32 to 48, thus the architectures searched by TopoNAS could have similar parameters to other competitive methods. The results show that, with similar amount of parameters, our TopoNAS can achieve lower test error, significantly outperforms our baseline methods DARTS and MiLeNAS.

4.3 Results on ImageNet dataset

We transfer our searched cells *TopoNAS-fixed-DARTS*, *TopoNAS-arbitrary-DARTS*, *TopoNAS-fixed-MiLe* and *TopoNAS-arbitrary-MiLe* into ImageNet dataset, the stacked layer number 14 in evaluation network is the same as DARTS. Table 2 presents the evaluation results on ImageNet and shows our cells searched on CIFAR-10 can be successfully transferred to ImageNet. Our method achieves competitive performances to other methods. Retraining hyperparameter settings are presented in A.1.

Table 2: Search results on ImageNet and comparison with state-of-the-art methods. Search cost is tested on a single NVIDIA GTX 1080 Ti GPU.

Methods	Test Err. (%)		Params	Flops	Search Cost	Search Method
	top-1	top-5	(M)	(M)	(GPU days)	
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	-	manual
ShuffleNetV2 2× (Ma et al., 2018)	25.1	-	∼5	591	-	manual
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	570	3150	evolution
DARTS (2nd order) (Liu et al.)	26.7	8.7	4.7	574	4.0	gradient-based
ProxylessNAS (ImageNet) (Cai et al.)	24.9	7.5	7.1	465	8.3	gradient-based
PC-DARTS (CIFAR-10) (Xu et al.)	25.1	7.8	5.3	586	0.1	gradient-based
PC-DARTS (ImageNet) (Xu et al.)	**24.2**	7.3	5.3	597	3.8	gradient-based
MiLeNAS (He et al., 2020)	24.7	7.6	5.3	584	0.3	gradient-based
TopoNAS-fixed-DARTS	25.1	7.8	4.9	562	0.6	gradient-based
TopoNAS-fixed-MiLe	25.4	7.9	4.8	548	0.6	gradient-based
TopoNAS-arbitrary-DARTS	25.3	8.1	4.8	537	0.6	gradient-based
TopoNAS-arbitrary-MiLe	24.6	7.5	5.3	598	0.6	gradient-based

Table 3: Evaluation results on CIFAR-10 with different layer numbers.

Methods	Layer Number							
	10		12		16		20	
	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)
DARTS (2nd order)	3.02	1.6	3.01	1.7	2.84	2.7	**2.74**	3.3
TopoNAS-fixed-DARTS	3.01	1.7	**2.72**	1.8	2.93	2.8	3.17	3.5
TopoNAS-fixed-MiLe	3.12	1.7	**2.68**	1.8	3.01	2.7	3.11	3.4
TopoNAS-arbitrary-DARTS	3.09	1.7	**2.67**	1.9	2.89	2.7	2.81	3.3
TopoNAS-arbitrary-MiLe	2.78	1.9	**2.60**	2.0	2.84	3.0	3.15	3.7

4.4 Ablation Studies

4.4.1 Effect of layer numbers of CIFAR-10 retraining

Based on the reported results (Liu et al.), DARTS tend to search a cell that simply use the two input nodes $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as inputs for most of nodes. In contrast, our method empirically encourages a "deeper" cell architecture, in which nodes near to output usually take the nearest precursor nodes as input. This difference in the cell depth implies that during retraining, the network cell number 20 in evaluation for DARTS may not still be optimal for our method. In this way, we investigate the performance of DARTS cells and our searched cells at different number of layers.

The results are summarized in Table 3. Our searched models usually get higher accuracies at a lower layer number, however, the accuracy decreases as the layer number reduces on original DARTS cells. By decreasing the layer numbers, our models can still obtain competitive performances with significant lower parameter numbers, which indicates that our cells are more efficient with removal of redundant node connections. Based on the results, we set the layer number 12 for all of our searched models in CIFAR-10 experiments. More ablation studies can be found in A.3 .

5 Conclusion

Besides operations, topology also matters for neural architectures since it controls the interactions between features of operations. In this paper, we highlight the topology learning in differentiable NAS, and propose an explicit topology modeling method named TopoNAS, which directly decouples operation selection and topology learning. Concretely, we introduce a combinatorial probabilistic distribution to explicitly indicate the target topology, and leverage a passive-aggressive regularization to suppress invalid topology during search. We apply our TopoNAS into two typical algorithms DARTS and MiLeNAS, and experimental results show that our TopoNAS can significantly boost the performance, with better classification accuracy but much less parameters. As for future work, we will dig more about the relationships among depth of supernet and target net, number of nodes in a cell, and our topology learning.

References

G Anandalingam and Terry L Friesz. Hierarchical optimization: An introduction. *Annals of Operations Research*, 34(1):1–11, 1992.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 550–559, 2018.

Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Stabilizing darts with amended gradient estimation on architectural parameters, 2019.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1294–1303, 2019.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131, 2018.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104, 2018.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. In *Advances in Neural Information Processing Systems*, pp. 2684–2694, 2019.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1284–1293, 2019a.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019b.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A Appendix

A.1 Details of experimental settings

A.1.1 Searching on CIFAR-10

At the search stage, following DARTS (Liu et al.), the supernet is built by stacking 8 cells with 6 normal cells and 2 reduction cells located at 2th and 5th cell. And the initial channel number is set to 16. The CIFAR-10 (Krizhevsky et al., 2014) training set is split to 3 equal-size sets $\mathcal{D}_{tr}$, $\mathcal{D}_{val_\alpha}$ and $\mathcal{D}_{val_\beta}$ for optimizing $\boldsymbol{w}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, respectively. We train the network for 50 epochs using Algorithm 1 with batch size 64. The network weights $\boldsymbol{w}$ are optimized using SGD with momentum 0.9 and 3×10^{-4} weight decay. The associated initial learning rate is set to 0.025 with cosine decay strategy. For optimizing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we use Adam optimizer with a fixed learning rate 3×10^{-4} and 10^{-3} weight decay. For deriving architectures, we simply choose the latest optimized networks at 50th epoch. Besides, in each experiment, we run the search 4 times with different random seeds and choose the architecture with highest evaluation accuracy as final result.

A.1.2 CIFAR-10 retraining

For CIFAR-10 retraining, we train the network for 600 epochs with batch size 96, and a cosine decayed learning rate scheduler is adopted with initial value 0.025. The cells are stacked 20 layers for DARTS-searched architectures and 12 layers for our architectures, where cells allocated at $1/3$ and $2/3$ of the total depth of network are reduction cells. Following DARTS (Liu et al.), additional enhancements include cutout (DeVries & Taylor, 2017), path dropout of probability 0.2 and auxiliary towers with weight 0.4. For each architecture, we train 10 times and report its mean error on validation set with standard deviation.

A.1.3 ImageNet retraining

For retraining on ImageNet dataset, we use 224×224 as input image size. The network is trained for 250 epochs with batch size 1024 and weight decay 3×10^{-5} using SGD. A linear learning rate scheduler is adopted with initial value 0.5. Other hyperparameters follow DARTS (Liu et al.).

Algorithm 1 Differentiable operation and topology searching for TopoNAS.

```
Initialize network with weights $\boldsymbol{w}$ parametrized by architecture variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.
while not converged do
    if use MiLeNAS then
        Update weights $\boldsymbol{w}$ by descending $\nabla_w \mathcal{L}_{tr}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
        Update operation architecture $\boldsymbol{\alpha}$ by descending $\nabla_{\boldsymbol{\alpha}}(\mathcal{L}_{tr}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) + \eta_\alpha \lambda \mathcal{L}_{val_\alpha}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}))$
        Update topology architecture $\boldsymbol{\beta}$ by descending $\nabla_{\boldsymbol{\beta}}(\mathcal{L}_{tr}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) + \eta_\beta \lambda \mathcal{L}_{val_\beta}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}))$
    else
        Update operation architecture $\boldsymbol{\alpha}$ by descending $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val_\alpha}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ ;
        Update topology architecture $\boldsymbol{\beta}$ by descending $\nabla_{\boldsymbol{\beta}} \mathcal{L}_{val_\beta}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ ;
        Update weights $\boldsymbol{w}$ by descending $\nabla_{\boldsymbol{\beta}} \mathcal{L}_{tr}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ ;
    end if
end while
Derive final architecture using architecture variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ .
```

A.2 Details of applying our TopoNAS in DARTS and MiLeNAS

A.2.1 DARTS

We first introduce the original DARTS method, which considers architecture optimization and weights optimization as a bi-level optimization problem (Anandalingam & Friesz, 1992; Colson et al., 2007):

$$\min_{\boldsymbol{\alpha}} \ \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \ \text{ s.t. } \ \boldsymbol{w}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}). \tag{19}$$

In contrast, we introduce topological variables $\boldsymbol{\beta}$ as an additional architecture variables, and now this optimization becomes

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \ \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha},\boldsymbol{\beta}),\boldsymbol{\alpha},\boldsymbol{\beta}) \tag{20}$$

$$\text{s.t.} \ \boldsymbol{w}^*(\boldsymbol{\alpha},\boldsymbol{\beta}) = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) \tag{21}$$

To solve the optimization problem, DARTS uses both first-order and second-order for the optimization. Nevertheless, our method involves a topology architecture optimization step, which may be more computation consuming than DARTS. For efficiency consideration, all of our experiments use the first-order approximation.

A.2.2 MILENAS

Compared to the first-order approximation in DARTS, MiLeNAS (He et al., 2020) mixes the training loss and validation loss together for architecture optimization:

$$\begin{aligned}\boldsymbol{w} &= \boldsymbol{w} - \eta_{\boldsymbol{w}}\nabla_{\boldsymbol{w}}\mathcal{L}_{tr}(\boldsymbol{w},\boldsymbol{\alpha})\\ \boldsymbol{\alpha} &= \boldsymbol{\alpha} - \nabla_{\boldsymbol{\alpha}}(\mathcal{L}_{tr}(\boldsymbol{w},\boldsymbol{\alpha}) + \eta_{\alpha}\lambda\mathcal{L}_{val}(\boldsymbol{w},\boldsymbol{\alpha})),\end{aligned} \tag{22}$$

where η_w and η_α denote the step size in a gradient descending step. Our proposed explicit learning for topology can be applied to MiLeNAS by parameterizing topologies using additional topological variables $\boldsymbol{\beta}$. Adapting to MiLeNAS, $\boldsymbol{\beta}$ can also be efficiently optimized by mixing training loss and validation loss:

$$\begin{aligned}\boldsymbol{w} &= \boldsymbol{w} - \eta_{\boldsymbol{w}}\nabla_{w}\mathcal{L}_{tr}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta})\\ \boldsymbol{\alpha} &= \boldsymbol{\alpha} - \nabla_{\boldsymbol{\alpha}}(\mathcal{L}_{tr}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) + \eta_{\alpha}\lambda\mathcal{L}_{val_\alpha}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}))\\ \boldsymbol{\beta} &= \boldsymbol{\beta} - \nabla_{\boldsymbol{\beta}}(\mathcal{L}_{tr}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) + \eta_{\beta}\lambda\mathcal{L}_{val_\beta}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta})),\end{aligned} \tag{23}$$

Our iterative procedure is summarized as Algorithm 1.

A.3 ABLATION STUDIES

A.3.1 EFFECT OF LAYER NUMBERS OF IMAGENET RETRAINING

As in the Ablation studies of Sec. 4.4.1 on CIFAR-10 dataset, we also examine the effect of layer numbers during retraining on ImageNet dataset. The results are summarized in Table 4. For fair comparison, all networks are trained using the same strategy. From the results, we can see that with the increasement of number of layers, all methods tend to have better classification performance. However, comparing to the baseline method DARTS, our TopoNAS can achieve better performance over all number of layers. Note that for ImageNet dataset, we only implement medium number of layers (10∼20) due to the consideration of training cost. Thus at this level of layer number, increasing layers contributes to the improvement of classification performance. Note that our searched cell tends to have ”deeper structure” than other DARTS variants. Then we can expect the performance superiority of our TopoNAS to DARTS when dealing with the same number of layers, which also in a way implies the advantages of TopoNAS in discovering more diverse cell types.

Table 4: Evaluation results on ImageNet with different layer numbers.

Methods	Layer Number									
	10		12		14		16		20	
	Error (%)	Params (M)	Error (%)	Params (M)	Error (%)	Params (M)	Error (%)	Params (M)	Error (%)	Params (M)
DARTS (2nd order)	27.9	3.6	26.8	3.8	25.8	4.7	25.4	5.5	**25.2**	6.7
TopoNAS-fixed-DARTS	27.0	3.7	26.2	3.9	25.1	4.9	24.9	5.7	**24.4**	6.8
TopoNAS-fixed-MiLe	27.5	3.7	26.9	3.9	25.4	4.8	25.3	5.5	**24.7**	6.6
TopoNAS-arbitrary-DARTS	26.6	3.8	26.4	4.0	25.3	4.8	24.9	5.5	**24.2**	6.5
TopoNAS-arbitrary-MiLe	25.9	4.0	25.4	4.3	24.6	5.3	**24.4**	6.1	**24.4**	7.3

Table 5: Results on CIFAR-10 with different node numbers.

Node Number	Test Error (%)	Params (M)
7	2.60	2.0
8	2.56	2.5
9	2.53	2.3
11	2.77	3.6

A.3.2 Effect of more nodes in a cell

To further investigate the effects of topology learning, we increase the node number N to 8, 9 and 11, which can search for even more complex topologies. As shown in Table 5 , all the cells are searched and stacked 12 layers for evaluation on CIFAR-10. Based on the results, the increase of node number can improve the performance, however, the performance drops when $N = 11$, which might because the increasing node number makes architectures hard to optimize, and needs more careful selection on hyperparameters (*e.g.*, number of layers). Besides, as the visualization of cells searched with $N = 11$ in Figure 8, we find that `identity` operation appears to increase for the case of larger node numbers, which might result from that identify tends to ease the optimization to some extent.

Table 6: Evaluation results on CIFAR-10 with different fixed node numbers.

Methods	Fixed Node Number							
	1		2		3		arbitrary	
	Error (%)	Params (M)	Error (%)	Params (M)	Error (%)	Params (M)	Error (%)	Params (M)
DARTS	3.06	3.5	2.75	3.3	2.94	3.3	-	-
TopoNAS-DARTS	2.63	3.5	2.48	3.2	2.58	3.5	2.44	3.2

A.3.3 Comparision between different fixed node numbers

To investigate the importance of arbitrary connections, we futher implement experiments on different fixed node numbers. The results are shown in table 6, for fair comparison, we change the initial channel number C of each fixed node number to keep similar parameters. From the results, we can infer that, at the same level of parameters, the increase of node number does not always ensure the performance improvement. This might because more nodes will increase the diversity of network; however, since we control the parameter amount to be similar, more nodes in a cell also imply parameters for each operation will reduce accordingly, limiting their modeling capacity. We can see fixed node number 2 achieves higher performances since it reaches a better tradeoff between cell diversity and modeling capacity per operation. Meanwhile, the arbitrary edge selection performs better than all the fixed edge selections, because the network can learn the topology more adaptively instead of being specified by a manual rule.

A.3.4 Comparison between different topology modeling manners

As previously discussed, we empirically find that the topology modeling of selecting input edges performs poorly and propose to switch topology space with output edges. In this section, we implement experiments on CIFAR-10 dataset and compare these two manners. As shown in Table 7, modeling topological probabilities with output nodes performs better than using input nodes, and the input manner also performs worse than the original DARTS, it indicates that a bad topology does harm to the performance of neural networks.

Keeping two output nodes in DARTS. DARTS chooses the top-2 input edges for each node, however, TopoNAS switches the input edge selection to output edge selection, which causes a slightly different search space. In this section, we investigate the influence of switching input edges to output edges in DARTS. The results are summarized in Table 7. We can infer that, DARTS obtains similar performances on input edge selection and output edge selection since these two modeling methods have the same edge numbers (*i.e.*, operation number). Besides, when DARTS and TopoNAS-fixed

Table 7: Test errors on CIFAR-10 with different topology modeling manners.

Method	Input Edges (%)	Output Edges (%)
DARTS (2nd order)	2.76	2.82
TopoNAS-fixed-DARTS	3.08	2.72

Table 8: Test errors on CIFAR-10 with joint optimization or alternating optimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Method	joint optimization (%)	alternating optimization (%)
TopoNAS-fixed-DARTS	3.01	2.72
TopoNAS-arbitrary-DARTS	3.08	2.67

have the same search space, our method can still outperform DARTS, it indicates the effectiveness of our explicit topology learning.

A.3.5 COMPARISION BETWEEN ALTERNATING OPTIMIZATION AND JOINT OPTIMIZATION OF α AND β

TopoNAS introduces additional topological parameters, and thus slows down the training speed. Nevertheless, if we optimize $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ jointly using the same data (*i.e.*, same mini-batch), the search cost will decrease to the same as DARTS. In this section, we conduct experiments to investigate the difference between alternating optimization and joint optimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. From the results summarized in Table 8, we find that the joint optimization performs worse than alternating optimization. This might because the joint optimization worsens the coupled modeling of operation selection and topology learning, thus leads to overfit on training data. It also indicates us that we should consider operation selection and topology learning independently, instead of coupling them together using a manually-designed rule.

A.3.6 COMPARISION BETWEEN DIFFERENT APPROXIMATION METHODS

As previously discussed before, for efficiency consideration, our TopoNAS uses first-order approximation in DARTS; however, the second-order approximation might perform better since it involves higher-order derivative. So we further perform experiments to compare the performances of these two approximation methods on TopoNAS. The evaluation results on CIFAR-10 are summarized in Table 9. The performance on the second-order approximation is slightly better than the first-order approximation; however, it takes much more time on searching (7.8 GPU days vs. 0.6 GPU day), so we use first-order approximation on all of our experiments.

A.3.7 INVESTIGATING THE STABILITY DURING SEARCH

We also investigate the stability in the search stage by deriving and evaluating the cells at different epochs. The results are presented in Table 10. Comparing the performance at different epochs for each run, the accuracies and parameters for DARTS change rapidly during search, and the best architectures often appear before 50th epoch. However, our TopoNAS acts more steadily.Note that the best performance in different runs for DARTS also changes more dramatically than TopoNAS. In this way, the performance in Table 10 indicates that our method can obtain more stable performance during search. This might result from that our TopoNAS decouples the topology learning and operation selection. So we simply use the latest optimized supernet at 50th epoch for architecture derivation.

A.4 DETAILS OF AVERAGE ACCURACY COMPUTATION FOR EACH TOPOLOGY IN NAS-BENCH-201

In Figure 1, we calculate the average accuracies of topologies in NAS-bench-201 (Dong & Yang). To compare the performance of different architectures, for each topology, we conduct the average accuracies of all its possible architectures (*i.e.*, all the operation combinations) as its accuracy, *i.e.*,

$$\text{Avg-ACC}(\tau) = \text{Mean}_{\boldsymbol{o}\in\mathcal{O}_\tau}(\text{ACC}(\tau, \boldsymbol{o})), \tag{24}$$

Table 9: Test errors on CIFAR-10 with different approximation methods.

Method	first order (%)	second order (%)
DARTS	3.00	2.76
TopoNAS-fixed-DARTS	2.72	2.68

Table 10: Evaluation results on CIFAR-10 at different epochs during search. We set the number of layers as 20 and 12 for DARTS (2nd order) and TopoNAS-fixed-DARTS, respectively, and each method runs 3 times independently.

Methods	Epochs							
	20		30		40		50	
	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)	Test Err. (%)	Params (M)
DARTS (2nd order)	2.93	3.6	**2.88**	3.2	3.02	2.5	2.90	2.3
DARTS (2nd order)	3.05	3.1	3.16	2.3	**3.00**	2.3	3.41	2.1
DARTS (2nd order)	2.83	4.1	**2.82**	3.0	2.87	2.6	2.87	2.6
TopoNAS-fixed	2.89	2.0	2.94	1.9	2.81	1.8	**2.71**	1.8
TopoNAS-fixed	2.91	2.0	2.89	1.8	2.83	1.8	**2.69**	1.9
TopoNAS-fixed	2.85	2.1	2.81	1.9	**2.73**	1.8	2.75	1.9

where $\mathcal{O}_\tau$ denotes the operation space of topology τ; specifying τ and $\boldsymbol{o}$, the network architecture can be uniquely determined.

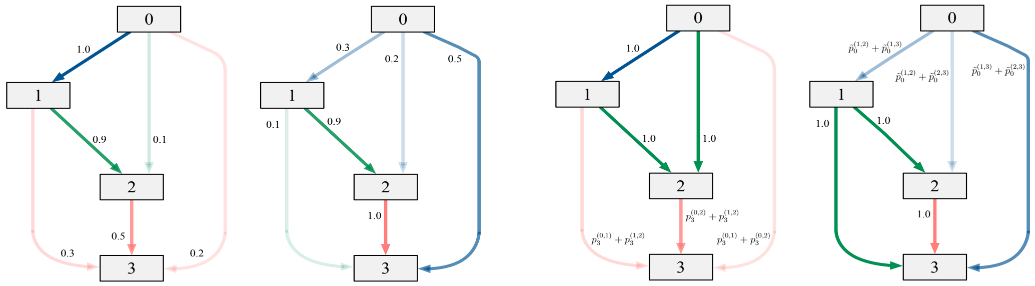
From Figure 1, we can infer that, even at the same edge number(*i.e.*, operation number), accuracies of different topologies lie in a large range. Besides, the more edge number may not always get the higher performance, *e.g.*, the average accuracy of edge number 6 is lower than the best average accuracy of edge number 5 and 4 in CIFAR-10, which indicates that we should highlight the topology learning in NAS, and also proves the performance boost of our arbitrary topology learning.

A.5 Diagram of topological parameterizing

The diagram of our topological modeling idea is illustrated in Figure 3. The left and right diagrams in subgraph (a) and (b) denote input edge selection method and output edge selection method, respectively. On subgraph (a), we show the simplified diagram which only selecting one input (output) node. The numbers on each edge denote the selection importance (probabilistic factors) of it. Meanwhile, we illustrate more complex diagrams of choosing two input (output) nodes on subgraph (b), the probabilistic factor on each edge is accumulated by all the combinatorial probabilities of this edge.

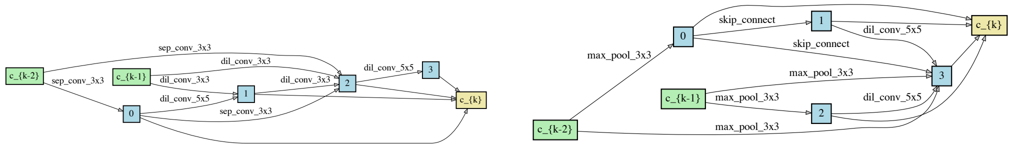
A.6 Visualization of searched cells

Our searched cells are visualized below. The visualizations show that, the normal cells tend to choose more convolutional operations, while reduction cells prefer pooling operations. Moreover, on topology, compared to DARTS, cells searched by TopoNAS are more “deeper” and “slimmer”, which indicates that our models perform better on a smaller layer number and are more parameter-efficient.


0
1
2
3
1.0
0.9
0.1
0.5
0.3
0.2
0
1
2
3
0.3
0.2
0.5
0.1
0.9
1.0
0
1
2
3
1.0
1.0
1.0
$p_3^{(0,2)} + p_3^{(1,2)}$
$p_3^{(0,1)} + p_3^{(1,2)}$
$p_3^{(0,1)} + p_3^{(0,2)}$
0
1
2
3
$\tilde{p}_0^{(1,2)} + \tilde{p}_0^{(1,3)}$
$\tilde{p}_0^{(1,2)} + \tilde{p}_0^{(2,3)}$
$\tilde{p}_0^{(1,3)} + \tilde{p}_0^{(2,3)}$
1.0
1.0
1.0

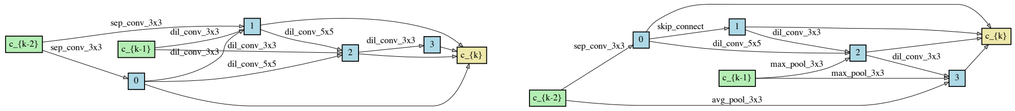
(a) (b)

Figure 3: The diagrams of input edge selection and output edge selection. (a) The simplified diagrams with one selecting input(output) node. (b) The combinatorial probabilities with selecting node number as 2. Left: input edge selection; right: output edge selection.



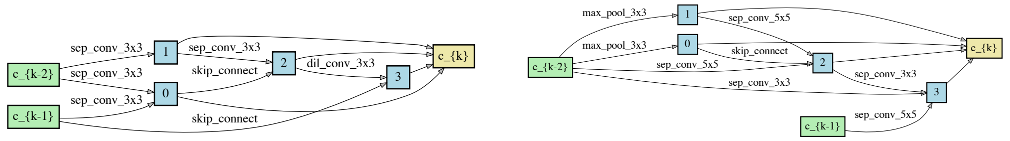
(a) Normal cell (b) Reduction cell

Figure 4: Cells for TopoNAS-fixed-DARTS with 2.72% testing error and 1.8M parameters on CIFAR-10.



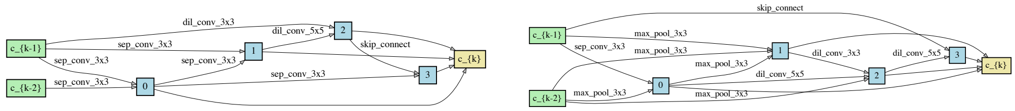
(a) Normal cell (b) Reduction cell

Figure 5: Cells for TopoNAS-fixed-MiLe with 2.68% testing error and 1.8M parameters on CIFAR-10.



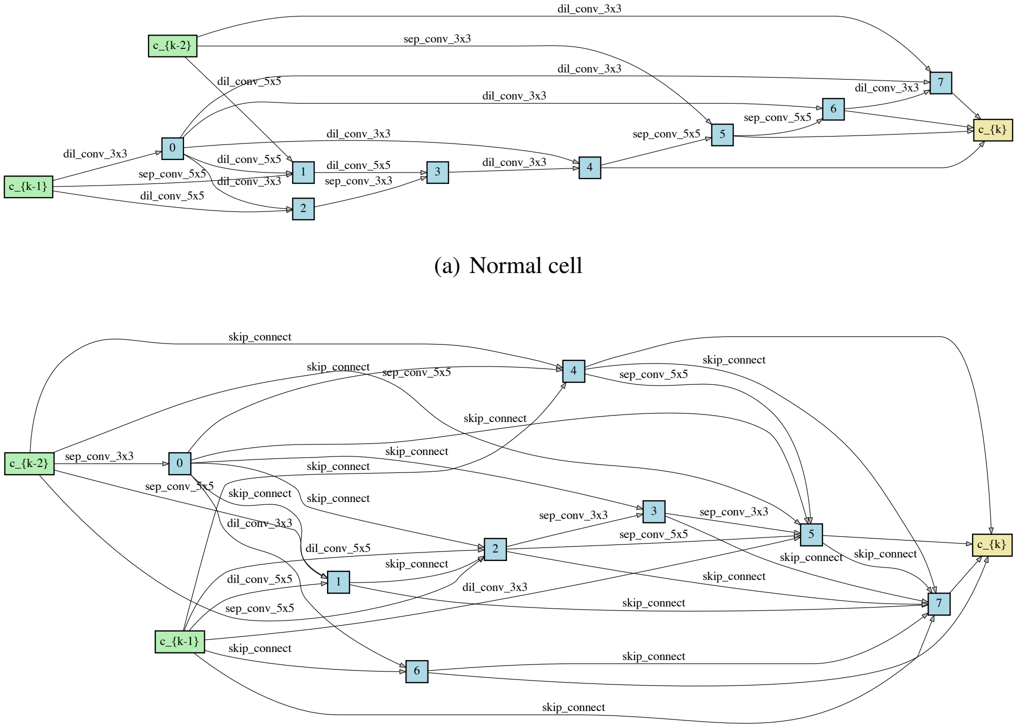
(a) Normal cell (b) Reduction cell

Figure 6: Cells for TopoNAS-arbitrary-DARTS with 2.67% testing error and 1.9M parameters on CIFAR-10.



(a) Normal cell (b) Reduction cell

Figure 7: Cells for TopoNAS-arbitrary-MiLe with 2.60% testing error and 2.0M parameters on CIFAR-10.



(a) Normal cell

(b) Reduction cell

Figure 8: Cells for TopoNAS-arbitrary-11 with 2.77% testing error and 3.6M parameters on CIFAR-10.