ENTRANNAS: TOWARDS CLOSING THE GAP BETWEEN THE ARCHITECTURES IN SEARCH AND EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most gradient-based neural architecture search methods construct a super-net for search and derive a target-net as its sub-graph for evaluation. There is a significant gap between the architectures in search and evaluation. As a result, current methods suffer from the problem of an inaccurate, inefficient, and inflexible search process. In this paper, we aim to close the gap and solve these problems. We introduce EnTranNAS that is composed of **En**gine-cells and **Tran**sit-cells. The Engine-cell is differentiable for architecture search, while the Transit-cell only transits the current sub-graph by architecture derivation. Consequently, the gap between the architectures in search and evaluation is significantly reduced. Our method also spares much memory and computation cost, which speeds up the search process. A feature sharing strategy is introduced for more efficient parameter training in the search phase. Furthermore, we develop a new architecture derivation method to replace the traditional one that is based on a hand-crafted rule. Our method enables differentiable sparsification, so keeps the derived architecture equivalent to the one in search. Besides, it supports the search for topology where a node can be connected to prior nodes with any number of connections, so that the searched architectures could be more flexible. For experiments on CIFAR-10, our search on the standard space requires only 0.06 GPU-day. We further have an error rate of 2.22% with 0.07 GPU-day for the search on an extended space. We can directly perform our search on ImageNet with topology learnable and achieve a top-1 error rate of 23.2%. Code will be released.

1 INTRODUCTION

Current neural architecture search (NAS) methods include reinforcement learning-based NAS (Baker et al., 2017; Zoph & Le, 2017), evolution-based NAS (Real et al., 2017; Liu et al., 2018b), Bayesian optimization-based NAS (Kandasamy et al., 2018; Zhou et al., 2019), and gradient-based NAS (Luo et al., 2018; Liu et al., 2019b), some of which have successfully been applied to related tasks for better architectures, such as semantic segmentation (Chen et al., 2018; Liu et al., 2019a) and object detection (Peng et al., 2019; Chen et al., 2019b; Ghiasi et al., 2019; Tan et al., 2019b).

Among the NAS methods, gradient-based algorithms gain much attention because of the simplicity. Liu *et al.* first propose the differentiable search framework, DARTS (Liu et al., 2019b), based on continuous relaxation and weight sharing (Pham et al., 2018), and inspire a series of follow-up studies (Xie et al., 2019; Cai et al., 2019; Chang et al., 2019; Xu et al., 2020; Chen et al., 2019a). In DARTS, different architectures share their weights as sub-graphs of a super-net. The super-net is trained for search, after which a target-net is derived for evaluation by only keeping important paths according to their softmax activations with a hand-crafted rule. Despite the simplicity, the architecture for evaluation only covers a small subset of the one for search, which causes a significant gap between super-net and target-net. We point out that the gap causes the following problems:

- *inaccurate*: The super-net trained in the search phase is a summation among all candidate connections with a trainable distribution induced by softmax. It essentially optimizes a feature combination, instead of feature selection, which is the real goal of a search problem. As noted by (Chang et al., 2019; Yao et al., 2020), operations may be highly correlated. Even if the

weight of some connection is small, the corresponding path may be indispensable for the performance. So the target-net derived from a high-performance super-net is not ensured to be a good one (Sciuto et al., 2020). The search process is inaccurate.

- *inefficient*: Because the super-net is a combination among all candidate connections, the whole graph needs to be stored in both forward and backward stages, which requires much memory and computational consumption. As a result, the search can be performed only on a very limited number of candidate operations, and the super-net is slow and inefficient to train.
- *inflexible*: The gap between the architectures in search and evaluation does not allow the search for topology in a differentiable way. In current methods (Liu et al., 2019b; Xie et al., 2019; Cai et al., 2019; Xu et al., 2020; Chen et al., 2019a), the target-net is derived based on a hand-crafted rule where each intermediate node keeps the top-2 strongest connections to prior nodes. However, it limits the diversity of derived architectures in the topological sense. There is no theoretical or experimental evidence that supports this rule. Therefore, the search result is not flexible as we cannot obtain other kinds of topologies.

Some studies adopt the Gumbel Softmax strategy (Jang et al., 2016; Maddison et al., 2016) to sample a target-net that approaches to the one in search so that the gap can be reduced (Xie et al., 2019; Wu et al., 2019; Chang et al., 2019; Dong & Yang, 2019). But still, the demand for computation and memory of the whole graph is not relieved. Chen *et al.* (Chen et al., 2019a) propose a progressive shrinking method to bridge the depth gap between the super-net and target-net. NASP (Yao et al., 2020) and ProxylessNAS (Cai et al., 2019) only propagate the proximal or sampled paths in search, which effectively reduces the computational cost. However, all these methods do not support the search for more flexible topologies in a differentiable way. DenseNAS (Fang et al., 2019) and PC-DARTS (Xu et al., 2020) introduce another set of trainable parameters to model path probabilities, but the target-net is still derived based on a hand-crafted rule.

In this paper, we aim to close the gap between the architectures in search and evaluation, and solve the problems mentioned above. Inspired by the observation that only one cell armed with learnable architecture parameters suffices to enable differentiable search, we introduce EnTranNAS composed of **En**gine-cells and **Tran**sit-cells. The Engine-cell is differentiable for architecture search as an *engine*, while the Transit-cell only *transits* the current derived architecture. So the network in search is close to that in evaluation. We also adopt a feature sharing strategy that improves the training efficiency and reduces the memory cost in search. Given that Engine-cell still has a gap with the architecture in evaluation, we further develop a new architecture derivation method that enables differentiable sparsification. The connections with non-zero weights are active for evaluation, which keeps the derived architecture equivalent to the one in search, and meanwhile supports differentiable search for more flexible topologies.

We list the contributions of this study as follows:

- We introduce a new NAS algorithm, named EnTranNAS, which effectively reduces the gap between the architectures in search and evaluation. A feature sharing strategy is developed for improving training efficiency and reducing memory cost of the super-net during search.
- We propose a new architecture derivation method to replace the hand-crafted rule widely adopted in current studies. The derived target-net has the equivalent architecture to the one in search, which closes the architecture gap between search and evaluation. It also makes topology learnable to explore more flexible search results.
- Extensive experiments verify the validity of our proposed methods. We achieve an error rate of 2.22% on CIFAR-10 with 0.07 GPU-day. Our method is able to efficiently search for flexible architectures of different scales directly on ImageNet with state-of-the-art performances.

2 ENTRANNAS

In this section, we first briefly review the gradient-based search method widely adopted in current studies, and then develop our EnTranNAS. Finally, we show how to close the gap and meanwhile enable the differentiable search for topology in EnTranNAS-DST.

2.1 PRELIMINARIES

In (Liu et al., 2019b; Xie et al., 2019; Cai et al., 2019; Chen et al., 2019a; Xu et al., 2020; Chang et al., 2019), the cell-based search space is represented by a directed acyclic graph (DAG) composed

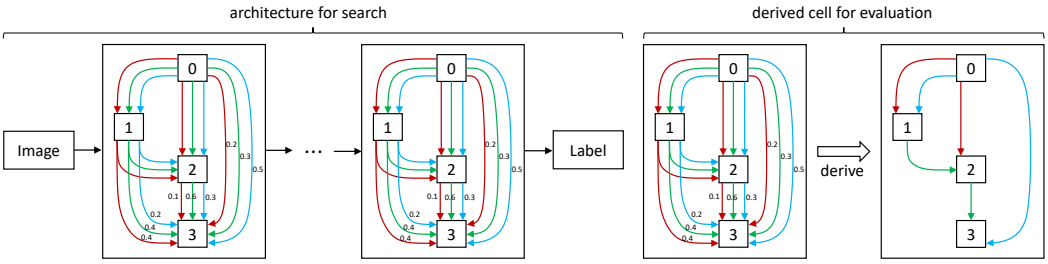

Figure 1: A diagram of DARTS. The target-net is derived by keeping the top-2 strongest connections of each node and has a significant gap with the architecture in search. The connections in different color represent candidate operations. The numbers beside them are exemplar weights.

of n nodes $\{x_1, x_2, \cdots, x_n\}$ and a set of edges $E = \{e^{(i,j)} | 1 \leq i < j \leq n\}$. For each edge $e^{(i,j)}$, there are K connections in accordance with the candidate operations $\mathcal{O} = \{o_1, \cdots, o_K\}$. The forward propagation of the super-net for search is formulated as:

$$x_j = \sum_{i<j} \sum_{k=1}^{K} p_k^{(i,j)} o_k(w_k^{(i,j)}, x_i), \tag{1}$$

where $p_k^{(i,j)} \in \{0, 1\}$ is a binary variable that indicates whether the connection is active, o_k denotes the k-th operation, and $w_k^{(i,j)}$ is its corresponding weight on this connection and becomes none for non-parametric operations, such as max pooling and identity. Since binary variables are not easy to optimize in a differentiable way, continuous relaxation is adopted and $p_k^{(i,j)}$ is relaxed as:

$$p_k^{(i,j)} = \frac{\exp(\alpha_k^{(i,j)})}{\sum_k \exp(\alpha_k^{(i,j)})}, \tag{2}$$

where $\alpha_k^{(i,j)}$ is the introduced architecture parameter jointly optimized with the super-net weights. After search, as shown in Figure 1, a target-net is derived according to a hand-crafted rule based on $p_k^{(i,j)}$ as the importance of connections. We let $\mathbf{P} \in R^{|E| \times K}$ denote the matrix formed by $p_k^{(i,j)}$, and the forward propagation of the target-net for evaluation is formulated as:

$$x_j = \sum_{(i,k) \in S_j} o_k(w_k^{(i,j)}, x_i), \tag{3}$$

$$S_j = \{(i,k) | A_k^{(i,j)} = 1, \forall i < j, 1 \leq k \leq K\}, \quad \mathbf{A} = \texttt{Proj}_\Omega(\mathbf{P}), \tag{4}$$

where $A_k^{(i,j)}$ is the element of $\mathbf{A} \in \{0, 1\}^{|E| \times K}$ and Ω denotes the hand-crafted rule by which only the top-2 strongest connections of each node are projected onto 1 and others are 0.

It is shown that there is a gap between the super-net and target-net in DARTS. As mentioned in Section 1, the gap may cause a sub-optimal target-net, and the super-net is not efficient to train. Besides, the hand-crafted rule limits the derived architecture to be a fixed topology.

2.2 Engine-cell and Transit-cell

Given that only one cell armed with learnable parameters suffices to enable differentiable search, we re-design the DARTS framework. At the super-net level, we introduce EnTranNAS composed of **En**gine-cells and **Tran**sit-cells. As shown in Figure 2 (a), the architecture derivation is not a post-processing step as in DARTS, but is performed at each iteration of search. Engine-cell has the same role as the cell in DARTS and stores the whole DAG. It performs architecture search as an *engine* by optimizing architecture parameters $\alpha_k^{(i,j)}$. As a comparison, Transit-cell only *transits* the currently derived architecture as a sub-graph into later cells. By doing so, EnTranNAS keeps the differentiability for architecture search by Engine-cell, and effectively reduces the gap between the super-net and target-net. At the final layer of super-net, representation is output from a Transit-cell, which has the same architecture as the target-net. Thus, with more confidence, a higher super-net performance indicates a better target-net architecture. Besides, a huge amount of computation and memory overhead is saved, so we can use a larger batchsize to speed up the search process. In

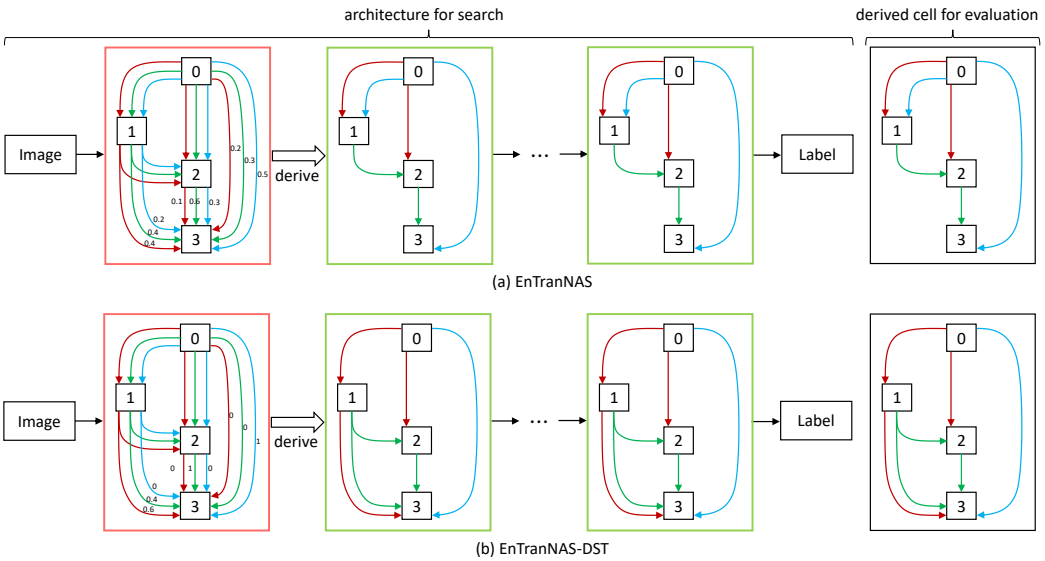

Figure 2: A diagram of our (a) EnTranNAS and (b) EnTranNAS-DST. Engine-cell and Transit-cell are in red and green boxes, respectively. EnTranNAS reduces the gap between the super-net and target-net. EnTranNAS-DST derive the architecture by keeping the connections with non-zero weights, so the valid computation graph for search is equivalent to the one for evaluation, and is not subject to any hand-crafted topology, which means the gap is closed. Zoom in to view better.

implementation, we set the first normal and reduction cells as Engine-cells, and the other cells as Transit-cells. Its illustration is shown in Figure 5 in Appendix A.

By introducing a temperature parameter (Xie et al., 2019; Wu et al., 2019), we calculate $p_k^{(i,j)}$ in Engine-cell as:

$$p_k^{(i,j)} = \frac{\exp(\alpha_k^{(i,j)}/\tau)}{\sum_k \exp(\alpha_k^{(i,j)}/\tau)}, \tag{5}$$

where τ is a temperature parameter. As $\tau \to 0$, $p_k^{(i,j)}$ approaches to a one-hot vector. We do not introduce the Gumble random variables as adopted in (Xie et al., 2019; Wu et al., 2019) because our architecture is not derived by sampling. We anneal τ with epoch so that Engine-cell approximately performs operation selection after convergence and has a smaller gap with Transit-cell.

2.3 Feature Sharing Strategy

Since Transit-cell only conducts the derived sub-graph, only a small portion of super-net weights $w_k^{(i,j)}$ is optimized in Transit-cell at each update. It impedes the training efficiency of super-net and may affect the search result due to the uneven optimization on candidate operations. In order to circumvent this issue, we introduce a feature sharing strategy at the cell level.

We notice that the non-parametric operation from a node to different nodes always produces the same features, which can be stored and computed only once. We extend this effect to parameterized operations, by assuming that the same operation from node i to other nodes $j > i$ always shares the same feature in one cell. The output of node x_j in our EnTranNAS is thus formulated as:

$$x_j = \begin{cases} \sum_{i<j} \sum_{k=1}^{K} p_k^{(i,j)} o_k(w_k^{(i)}, x_i), & \text{for Engine-cell,} \\ \sum_{(i,k)\in S_j} o_k(w_k^{(i)}, x_i), & \text{for Transit-cell,} \end{cases} \tag{6}$$

where $w_k^{(i)}$ is the parameter of operation k for node i, and becomes none for non-parametric operations. In this way, we further reduce the computation and memory overhead of super-net for search. In addition, the number of trainable connections in one cell is reduced from $|E| \times \bar{K}$ to $(n-1) \times \bar{K}$, where $\bar{K}$ denotes the number of parametrized operations and $|E| = C_n^2$. Consequently, the less learnable parameters have a more balanced opportunity to be optimized. The training efficiency of super-net is improved and the memory cost is reduced. Note that the feature sharing strategy harms the representation power of super-net. However, this approximation is valid for the search process as the features for selection are still produced by the same operations on the same nodes.

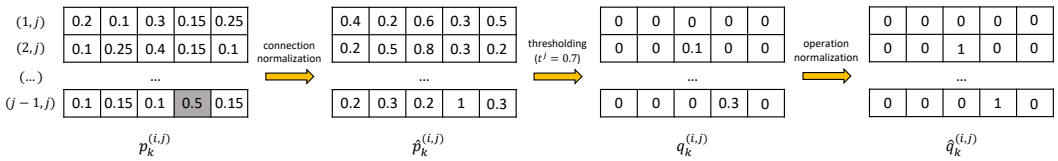

Figure 3: An illustration of the computation procedures of $\hat{q}_k^{(i,j)}$ as an example. The gray bin denotes the maximal element of $p_k^{(i,j)}$ for all $1 \leq k \leq K$ and $i < j$.

2.4 Differentiable Search for Topology

Albeit EnTranNAS reduces the gap between super-net and target-net, the Engine-cell still computes the whole graph and is different from the derived cell for evaluation. To this end, we further reduce the gap by proposing a new architecture derivation method that enables the differentiable search for topology, named EnTranNAS-DST. As shown in Figure 2 (b), in Engine-cell, the non-derived connections always have zero weights, such that the valid propagation of Engine-cell is equivalent to that of derived cell, which closes the gap between the architectures in search and evaluation.

In prior studies (Liu et al., 2019b; Chen et al., 2019a; Xu et al., 2020), connection weights are softmax activations and do not support zero values. A differentiable sparsification method is proposed in (Lee, 2019) for network pruning. We keep the softmax activations and also enable the differentiability for zero weighs. Concretely, we compute $p_k^{(i,j)}$ by Eq. (5), and then perform a connection normalization for each intermediate node $j > 1$ as:

$$\hat{p}_k^{(i,j)} = \frac{p_k^{(i,j)}}{\max\limits_{i<j, 1\leq k\leq K} \{p_k^{(i,j)}\}}, \tag{7}$$

where $\hat{p}_k^{(i,j)}$ is the activation after connection normalization. We introduce another set of trainable parameters $\{\beta^{(j)}\}_{j=2}^n$ and have the threshold of each intermediate node as $t^{(j)} = \text{sigmoid}(\beta^{(j)})$. With the thresholds, we can prune these connections as:

$$q_k^{(i,j)} = \sigma(\hat{p}_k^{(i,j)} - t^{(j)}), \tag{8}$$

where σ denotes the ReLU function. Finally, if there exists a k such that $q_k^{(i,j)} \neq 0$ for edge (i, j), we perform an operation normalization by:

$$\hat{q}_k^{(i,j)} = \frac{q_k^{(i,j)}}{\sum_k q_k^{(i,j)}}, \tag{9}$$

where $\hat{q}_k^{(i,j)}$ is used as the coefficients of connections. It enables sparsification in a differentiable way. Given that $\max_{i<j, 1\leq k\leq K}\{\hat{p}_k^{(i,j)}\} = 1$ and $t^{(j)} < 1$, there is at least one connection left for each intermediate node j by Eq. (8), so the cell structure will not be broken, and will keep valid along the training. An illustration of how do we compute $\hat{q}_k^{(i,j)}$ is shown in Figure 3.

In Engine-cell, we replace the $p_k^{(i,j)}$ in Eq. (6) with $\hat{q}_k^{(i,j)}$ for search. To derive the architecture in Transit-cell or for evaluation, the S_j in Eq. (6) changes from Eq. (4) to the following form:

$$S_j = \{(i, k) | \hat{q}_k^{(i,j)} > 0, \forall i < j, 1 \leq k \leq K\}. \tag{10}$$

In this way, we only keep the connections with non-zero weights as the derived architecture, which closes its gap with super-net, and does not limit the architecture to any hand-crafted topology.

In implementation, we enforce sparsification by adding a regularization. Our optimization is in accordance with the bi-level manner introduced in (Liu et al., 2019b). The loss function of our super-net when optimizing the architecture parameters $\{\alpha_k^{(i,j)}\}$ and $\{\beta^{(j)}\}$ is formulated as:

$$\min_{\alpha,\beta} \quad \mathcal{L}_{val}(\alpha, w^*) + \lambda \frac{1}{n-1} \sum\nolimits_{j=2}^{n} -\log(t^{(j)}), \tag{11}$$

where $\mathcal{L}_{val}(\alpha, w^*)$ is the validation loss with current network parameters w^*, and λ is a hyper-parameter by which we can control the degree of sparsification to obtain more flexible topologies. We visualize the search process of EnTranNAS-DST ($\lambda = 0.1$) in the video attached in the supplementary material. Its corresponding description is shown in Appendix C.

Engine-cell	Super-net Acc. (%)	Child-net Acc. (%)
all (DARTS)	88.29	63.97
one half	87.45	65.51
last	84.02	83.35
first	86.68	86.24

Table 1: Super-net accuracy drop in different settings of Engine-cell.

	Memory (G)	Batchsize (64)	Cost (GPU-day)
DARTS (1st order)	9.0	$\times$1	0.73
+Engine&Transit-cell	4.5	$\times$2	0.22
+feature sharing	2.6	$\times$4	0.09
+bottleneck	1.5	$\times$8	0.06

Table 2: Efficiency improved by each component. The three components are accumulated from top to bottom.

3 Related Work

Reinforcement learning is first adopted to assign the better architecture with a higher reward in (Baker et al., 2017; Zoph & Le, 2017). Follow-up studies focus on reducing the computational cost (Zoph et al., 2018; Zhong et al., 2018; Liu et al., 2018a; Cai et al., 2018a;b; Pham et al., 2018). As another line of NAS methods, evolution-based algorithms search for architectures as an evolving process towards better performance (Xie & Yuille, 2017; Real et al., 2017; Liu et al., 2018b; Real et al., 2019; Elsken et al., 2019; Miikkulainen et al., 2019). However, the search cost of both reinforcement leaning- and evolution-based methods is still demanding for practical applications. A good solution to this problem is one-shot methods that constructs a super-net covering all candidate architectures (Bender et al., 2018; Brock et al., 2018). The super-net is trained only once in search and is then deemed as a performance estimator. Some studies train the super-net by sampling a single path (Guo et al., 2019; Li & Talwalkar, 2019; You et al., 2020) in a chain-based search space (Hu et al., 2020; Cai et al., 2020; Mei et al., 2020; Yu et al., 2020). As a comparison, DARTS-based methods (Liu et al., 2019b; Xie et al., 2019) introduce architecture parameters jointly optimized with the super-net weights and performs the differentiable search in a cell-based space. Our study belongs to this category because it enables to discover more complex connecting patterns.

Despite the simplicity of DARTS, the architecture gap between search and evaluation impedes its validity. Follow-up studies aim to reduce the gap (Xie et al., 2019; Chen et al., 2019a; Chang et al., 2019), improve the search efficiency (Yao et al., 2020), and model path probabilities (Xu et al., 2020). However, all these methods derive the final architecture based on a hand-crafted rule, which constrains the topology and inevitably introduces an architecture gap. Our method differs from these studies in that the super-net of EnTranNAS dynamically changes in the search phase in a differentiable way, and then derives a target-net that has the same architecture as the one in search, and is not subject to any specific topology.

4 Experiments

We first analyze how each of our designs contributes to a more accurate, efficient and flexible search process, and then compare our results on CIFAR-10 and ImageNet with state-of-the-art methods.

4.1 Datasets and Implementation Details

Most of our settings for search and evaluation are consistent with (Liu et al., 2019b; Xu et al., 2020) for fair comparison. We have the detailed description of datasets and our implementations in Appendix B. All searched architectures are visualized in Appendix C.

4.2 Ablation Studies

Accuracy. EnTranNAS reduces the gap between the super-net and target-net. We test the effects of our design with different settings. After search on CIFAR-10, we perform inference only through the paths in the derived architecture as a child-net and compare their validation accuracies. As shown in Table 1, when all cells are set as Engine-cell, the super-net is equivalent to DARTS and has the largest accuracy drop. Setting one half of cells as Engine-cell also causes a large accuracy drop. As a comparison, when only one Engine-cell is used, we have a small accuracy drop, which demonstrates the validity of our method to reduce the gap. We set the first cell as Engine-cell because it has a better super-net accuracy than the last cell setting.

Efficiency. The improved efficiency of search on CIFAR-10 by each component is shown in Table 2. "Memory" shows the memory consumption with a batchsize of 64. "Batchsize" is the largest batch-size that can be used on a single GTX 1080 Ti GPU. "Cost" denotes the corresponding search time using the enlarged batchsize. Both of our Engine&Transit-cell design and feature sharing strategy

λ	Edges (N / R)	Params (M)	Flops (M)
0.2	9 / 8	5.07	580
0.1	11 / 6	5.88	673
0.05	13 / 14	6.99	779

Table 3: EnTranNAS-DST with different λ. "N" and "R" denote normal and reduction cell, respectively.

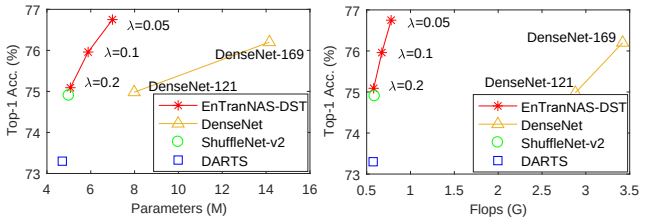

Figure 4: Comparison of top-1 accuracies on ImageNet with parameters (left) and Flops (right). Zoom in to view better.

significantly improve the search efficiency. Similar to (Xu et al., 2020) that reduces the number of channels when performing operations, we adopt a bottleneck before all candidate operations, and then recover the channel number for each intermediate node. Its detailed description is shown in Appendix B.4. When "bottleneck" is added, we can use a batchsize of 512 and reduce the search time to 0.06 GPU-day, which is about ten times as fast as our re-implementation of DARTS.

Flexibility. EnTranNAS-DST enables the differentiable search for topology and does not limit the number of edges in normal or reduction cells. We can obtain architectures with diverse capacities. A larger λ makes $t^{(j)}$ closer to 1, which cuts out more connections by Eq. (8) and leads to a more sparse architecture. Our search results on ImageNet with different λ are shown in Table 3. Their accuracies on ImageNet validation are depicted as a function of parameters and FLOPs in Figure 4. It is shown that we have a better trade-off than the strong baseline of manually designed architecture, DenseNet. Our EnTranNAS-DST (λ=0.05) surpasses DenseNet-169 with about one half of parameters and less than one fourth of FLOPs. Its FLOPs is relatively large in the mobile setting, because our method breaks the topology constraint and enables to look for higher performance outside the limitation of mobile setting. EnTranNAS-DST supports an explicit learning of flexible topologies with diverse capacities, which is beyond the ability of most current NAS methods.

4.3 Results on CIFAR-10

We search on CIFAR-10 from the standard and extended version of candidate operation space. The standard space has 7 operations and is consistent with current studies (Liu et al., 2019b; Xie et al., 2019; Chen et al., 2019a; Xu et al., 2020). The extended version additionally has 5 more operations, which are 1×1 convolution, 3×3 convolution, 1×3 then 3×1 convolution, 1×5 then 5×1 convolution, and 1×7 then 7×1 convolution. The two versions are listed in Table 6 in Appendix B.2. As shown in Table 4, for the standard search space, EnTranNAS achieves a state-of-the-art performance of 2.53% error rate with only 0.06 GPU-day. The accuracy is on par with MiLeNAS (He et al., 2020), whose search cost is 5 times as much as ours. To our best knowledge, 0.06 GPU-day is the top speed on DARTS-based search space. EnTranNAS-DST achieves a better performance with less parameters than EnTranNAS due to its superiority in learnable topology. When we search on the extended search space, a higher-performance architecture is searched with an error rate of 2.22%, which is better than NASP (Yao et al., 2020) that also searches on 12 operations. The search cost still has superiority and is increased by only 0.01 GPU-day than that on the standard version. That is because the extra operations only add the computational cost on Engine-cells, which account for a small portion of the super-net in search. Therefore, the search cost of EnTranNAS increases sub-linearly as the search space is enlarged.

4.4 Results on ImageNet

We use both EnTranNAS and EnTranNAS-DST for experiments on ImageNet with the standard search space. As shown in Table 5, EnTranNAS searched on CIFAR-10 has a top-1 error rate of 24.8%, which is competitive given that its search time is much more friendly than other methods. We also directly search on ImageNet. EnTranNAS achieves a top-1/5 error rates of 24.3%/7.2%, which is on par with PC-DARTS whose search cost is twice as much as ours. Different from other studies, EnTranNAS-DST is the only method that does not limit the topology of searched architecture. When λ in Eq. (11) is 0.05, a large model is searched and has a top-1 error rate of 23.2%, which surpasses EnTranNAS (ImageNet) by 1.1% error rate. The search cost is larger than EnTranNAS because at the beginning of search all connections to any node have non-zero weights and are kept active. As the search proceeds, EnTranNAS-DST dynamically drops connections. We see its search cost is still faster than PC-DARTS. An illustration of how EnTranNAS-DST changes its derived architecture in

Methods	Test Error (%)	Params (M)	Search Cost (GPU-day)	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
ENAS + cutout (Pham et al., 2018)	2.89	4.6	0.5	RL
AmoebaNet-B +cutout (Real et al., 2019)	2.55±0.05	2.8	3150	evolution
Hierarchical Evolution (Liu et al., 2018b)	3.75±0.12	15.7	300	evolution
DARTS (2nd order) + cutout (Liu et al., 2019b)	2.76±0.09	3.3	4.0	gradient
SNAS (moderate) + cutout (Xie et al., 2019)	2.85±0.02	2.8	1.5	gradient
ProxylessNAS+cutout (Cai et al., 2019)	2.08†	5.7	4.0	gradient
PC-DARTS + cutout (Xu et al., 2020)	2.57±0.07	3.6	0.1	gradient
NASP + cutout (Yao et al., 2020)	2.83±0.09	3.3	0.1	gradient
MiLeNAS + cutout (He et al., 2020)	**2.51±0.11**	3.87	0.3	gradient
EnTranNAS + cutout	2.53±0.06	3.45	**0.06**	gradient
EnTranNAS-DST + cutout	**2.48±0.08**	3.20	**0.10**	gradient
NASP (12 operations) + cutout (Yao et al., 2020)	2.44±0.04	7.4	0.2	gradient
EnTranNAS (12 operations) + cutout	**2.22±0.05**	7.68	**0.07**	gradient

Table 4: 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. The best and second best results are shown in blue and black bold. †: ProxylessNAS uses a different macro-architecture from other methods.

Methods	Test Err. (%)		Params	Flops	Search Cost	Method
	top-1	top-5	(M)	(M)	(GPU days)	
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	-	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	-	manual
ShuffleNet 2× (v2) (Ma et al., 2018)	25.1	-	∼5	591	-	manual
MnasNet-92 (Tan et al., 2019a)	25.2	8.0	4.4	388	-	RL
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	570	3150	evolution
DARTS (2nd order) (Liu et al., 2019b)	26.7	8.7	4.7	574	4.0	gradient
SNAS (Xie et al., 2019)	27.3	9.2	4.3	522	1.5	gradient
P-DARTS (Chen et al., 2019a)	24.4	7.4	4.9	557	0.3	gradient
ProxylessNAS (ImageNet) (Cai et al., 2019)	24.9	7.5	7.1	465	8.3	gradient
PC-DARTS (ImageNet) (Xu et al., 2020)	**24.2**	7.3	5.3	597	3.8	gradient
EnTranNAS (CIFAR-10)	24.8	7.6	4.9	562	0.06	gradient
EnTranNAS (ImageNet)	24.3	**7.2**	5.5	637	**1.9**	gradient
EnTranNAS-DST (ImageNet)	**23.2**	**6.9**	7.0	779	**2.2**	gradient

Table 5: Search results on ImageNet and comparison with state-of-the-art methods. Search cost is tested on eight NVIDIA GTX 1080 Ti GPUs. "(ImageNet)" indicates the method is directly searched on ImageNet. Otherwise, it is searched on CIFAR-10, and then transfered to ImageNet.

search is shown in the supplementary video and Appendix C. The parameters and FLOPs are large because EnTranNAS-DST is able to break the topology limitation and look for higher performance. We show in our ablation studies that architectures with flexible topologies of diverse capacities can be searched by controlling the hyper-parameter λ.

5 Conclusion

In this paper, we introduce EnTranNAS that reduces the gap between the architectures in search and evaluation and saves much computational and memory cost. A feature sharing strategy is adopted to improve the training efficiency of search. We further propose EnTranNAS-DST that closes the gap by a differentiable architecture derivation. It supports search for more flexible architectures without topology constraint. Experiments show EnTranNAS improves the search accuracy and efficiency, and EnTranNAS-DST extends the flexibility of searched architectures. We produce state-of-the-art search results on CIFAR-10 or directly on ImageNet with obvious superiority in search cost.

REFERENCES

B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcemen learning. In *ICLR*, 2017.

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

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

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *AAAI*, 2018a.

Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. In *ICML*, volume 80, pp. 678–687. PMLR, 2018b.

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

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020.

Jianlong Chang, Yiwen Guo, GAOFENG MENG, SHIMING XIANG, Chunhong Pan, et al. Data: Differentiable architecture approximation. In *NeurIPS*, pp. 874–884, 2019.

Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS*, pp. 8699–8710, 2018.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, pp. 1294–1303, 2019a.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NeurIPS*, pp. 6638–6648, 2019b.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, pp. 1761–1770, 2019.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *ICLR*, 2019.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. *arXiv preprint arXiv:1906.09607*, 2019.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pp. 7036–7045, 2019.

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. *arXiv preprint arXiv:1904.00420*, 2019.

Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *CVPR*, pp. 11993–12002, 2020.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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.

Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. *arXiv preprint arXiv:2002.09128*, 2020.

G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, pp. 2016–2025, 2018.

Yognjin Lee. Differentiable sparsification for deep neural networks. *arXiv preprint arXiv:1910.03201*, 2019.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, pp. 19–34, 2018a.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pp. 82–92, 2019a.

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

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

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, pp. 7816–7827, 2018.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pp. 116–131, 2018.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. Atomnas: Fine-grained end-to-end neural architecture search. In *ICLR*, 2020.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pp. 293–312. Elsevier, 2019.

Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation search in channel-level for object detection. In *NeurIPS*, pp. 14290–14299, 2019.

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, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *ICML*, pp. 2902–2911. JMLR. org, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pp. 4780–4789, 2019.

Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pp. 2820–2828, 2019a.

Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019b.

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*, pp. 10734–10742, 2019.

Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, pp. 1379–1388, 2017.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. In *ICLR*, 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 differentiable architecture search. In *ICLR*, 2020.

Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. Efficient neural architecture search via proximal iterations. In *AAAI*, 2020.

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. *arXiv preprint arXiv:2003.11236*, 2020.

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, 2020.

Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, June 2018.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. BayesNAS: A Bayesian approach for neural architecture search. In *ICML*, volume 97, pp. 7603–7613. PMLR, 2019.

B. Zoph and Q. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, June 2018.

A APPENDIX: THE SEARCH ARCHITECTURE OF ENTRANNAS

An illustration of our EnTranNAS architecture for search on CIFAR-10 is shown in Figure 5. There are 8 cells in total. The first cells of normal and reduction cells are set as Engine-cells, while the others are Transit-cells. Different configurations are compared in Table 1 to ablate our design choice.

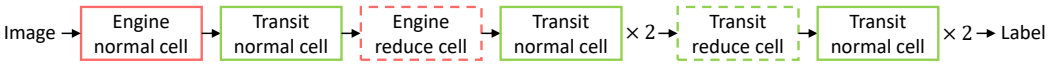

Figure 5: Our architecture for search. Engine-cell and Transit-cell are shown in red and green boxes, respectively. Normal and reduction cells are shown in solid and dotted boxes, respectively.

B APPENDIX: DATASETS AND IMPLEMENTATION DETAILS

B.1 DATASETS

The CIFAR-10 dataset consists of 60,000 images of size 32×32 in 10 classes. There are 50,000 images for training and 10,000 images for testing. In search, we use a half of the training set to optimize network weights and the other half as the validation set to optimize architecture parameters.

The ImageNet dataset contains 1.2 million training images, 50,000 validation images, and 100,000 test images. Following Xu et al. (2020), we directly perform the search on ImageNet by randomly sampling 10% images of the training set for network weights, and another 10% for architecture parameters.

The standard data augmentation methods are used for both CIFAR-10 and ImageNet.

B.2 SEARCH SPACE

Following Liu et al. (2019b); Chen et al. (2019a); Xu et al. (2020), we search on the commonly used operation space that includes 3×3 and 5×5 separable convolution, 3×3 and 5×5 dilated separable convolution, 3×3 max pooling, 3×3 average pooling, identity, and zero. Because our method is friendly to memory consumption and thus can be performed on a larger search space, we adopt an extended version of operation space that has 5 extra operations: 1×1 convolution, 3×3 convolution, 1×3 then 3×1 convolution, 1×5 then 5×1 convolution, and 1×7 then 7×1 convolution. The two versions of operation space are listed in Table 6. The zero operation is used to indicate the lack of connection between two nodes Liu et al. (2019b). For EngineNAS, we keep the zero operation as convention, while we remove the zero operation for EngineNAS-DST because it inherently has the ability to learn topology in an explicit manner.

Operation	Standard Space	Extended Space
zero	✓	✓
3×3 separable convolution	✓	✓
5×5 separable convolution	✓	✓
3×3 dilated separable convolution	✓	✓
5×5 dilated separable convolution	✓	✓
3×3 max pooling	✓	✓
3×3 average pooling	✓	✓
identity	✓	✓
1×1 convolution	-	✓
3×3 convolution	-	✓
1×3 then 3×1 convolution	-	✓
1×5 then 5×1 convolution	-	✓
1×7 then 7×1 convolution	-	✓

Table 6: The standard and extended operation space.

B.3 SEARCH AND EVALUATION DETAILS

Results on CIFAR-10. The super-net for search on CIFAR-10 is composed of 8 cells (6 normal cells and 2 reduction cells) with the initial number of channels as 16. There are 6 nodes in each cell. The first 2 nodes in cell k are input nodes, which are the outputs of cell $k-2$ and $k-1$, respectively. The output of each cell is the concatenation of all intermediate nodes. We train the super-net for 50 epochs with a batchsize of 512. SGD is used to optimize the super-net weights with a momentum of 0.9 and a weight decay of 3e-4. Its learning rate is set as 0.2 and is annealed down to zero with a cosine scheduler. We use the Adam optimizer for the architecture parameters $\{\alpha_k^{(i,j)}\}$ with a learning rate of 6e-4, a momentum of (0.5, 0.999) and a weight decay of 1e-3. The initial temperature in Eq. (5) is set as 5.0 and is annealed by 0.923 every epoch. We choose the architecture with the best validation accuracy as the searched one. We run our search for 5 times and select the best architecture. In evaluation, the target-net has 20 cells (18 normal cells and 2 reduction cells) with the initial number of channel as 36. We train for 600 epochs with a batchsize of 96, and report the mean error rate with the standard deviation of 5 independent runs. SGD optimizer is used with a momentum of 0.9, a weight decay of 3e-4, and a gradient clipping of 5. The initial learning rate is set as 0.025 and is annealed down to zero following a cosine scheduler. As convention, a cutout length of 16, a drop out rate of 0.2, and an auxiliary head are adopted.

Results on ImageNet. Following Xu et al. (2020), we perform three convolution layers of stride of 2 to reduce the resolution from the input size 224×224 to 28×28. The super-net for search has 8 cells with the initial number of channels as 16, while the target-net for evaluation has 14 cells and starts with 48 channels. We use a batchsize of 1,024 for both search and evaluation. In search, we train for 50 epochs with the same optimizers, momentum, and weight decay as that on CIFAR-10. The initial learning rate of network weights is 0.5 (annealed down to zero following a cosine scheduler). The learning rate of architecture parameters $\{\alpha_k^{(i,j)}\}$ (and $\{\beta^{(j)}\}$ for EnTranNAS-DST) is 6e-3. The initial temperature and its annealing ratio for EnTranNAS are the same as that on CIFAR-10. For EnTranNAS-DST, the initial temperature is set as 1 and is annealed by 0.9 every epoch. In evaluation, we train for 250 epochs from scratch using the SGD optimizer with a momentum of 0.9 and a weight decay of 3e-5. The initial learning rate is set as 0.5 and is annealed down to zero linearly. Following Xu et al. (2020), an auxiliary head and the label smoothing technique are also adopted.

B.4 BOTTLENECK

Similar to the partial channel connection strategy in Xu et al. (2020), we also try to reduce the number of channels to further save memory cost and reduce search time. Different from their method, we adopt the bottleneck technique as mentioned in Section 4.3 of our paper. The bottleneck technique is also widely used in traditional architectures He et al. (2016); Huang et al. (2017). Concretely, we perform a 1×1 convolution to reduce the number of channels by a ratio before feeding a node to all candidate operations. Another 1×1 convolution is appended to recover the number of channels to form each intermediate node. The reduction ratio is set as 4 in our experiments.

C APPENDIX: VISUALIZATION OF SEARCHED ARCHITECTURES

We visualize all searched architectures of our methods. Concretely, the EnTranNAS searched on the standard operation space of CIFAR-10 is shown in Figure 6 and 7. Its result on the extended space is shown in Figure 8 and 9. The EnTranNAS-DST searched on CIFAR-10 is shown in Figure 10 and 11. The EnTranNAS directly searched on ImageNet is shown in Figure 12 and 13. The results of EnTranNAS-DST with different λ on ImageNet are shown from Figure 14 to 19.

To better inspect the search process of EnTranNAS-DST, we further show the derived architecture of each epoch in the video in the supplementary file. It is shown that at the beginning of search, all connections are kept active by the initialization of architecture parameters $\{\alpha_k^{(i,j)}\}$ and $\{\beta^{(j)}\}$, *i.e.*, $\hat{q}_k^{(i,j)} > 0, \forall i, j, k$. As the optimization proceeds, we finally obtain an architecture with both operation and topology learnable. The topology is not subject to any hand-crafted rule.

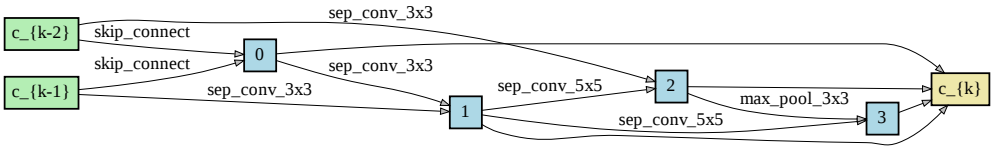

Figure 6: EnTranNAS normal cell searched on CIFAR-10

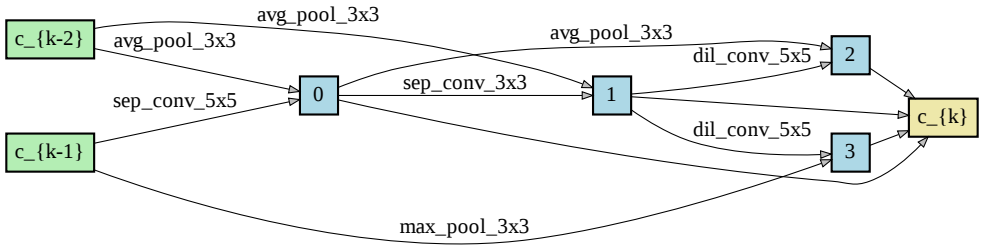

Figure 7: EnTranNAS reduction cell searched on CIFAR-10

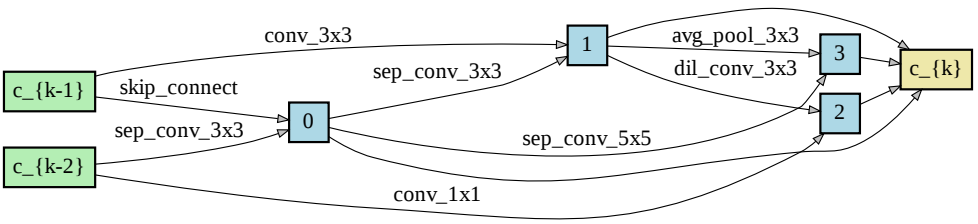

Figure 8: EnTranNAS (12 operations) normal cell searched on CIFAR-10

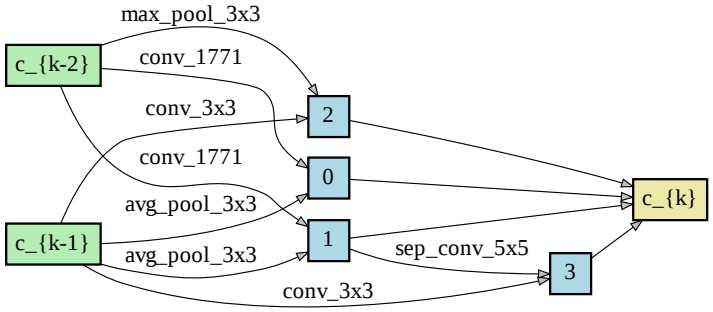

Figure 9: EnTranNAS (12 operations) reduction cell searched on CIFAR-10

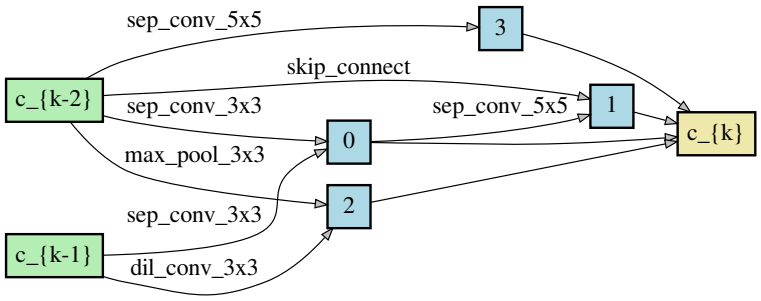

Figure 10: EnTranNAS-DST normal cell searched on CIFAR-10

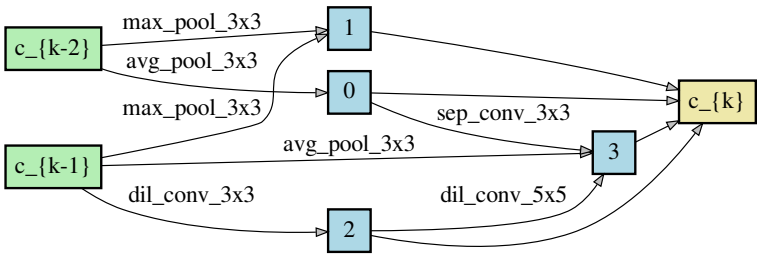

Figure 11: EnTranNAS-DST reduction cell searched on ImageNet

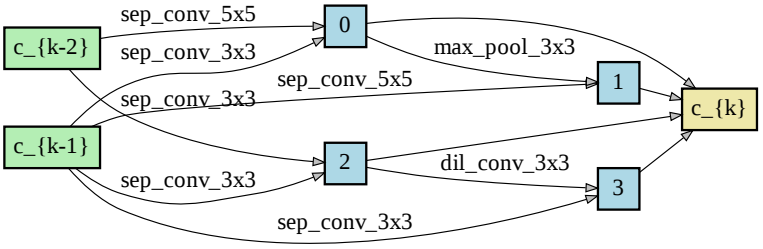

Figure 12: EnTranNAS normal cell searched on ImageNet

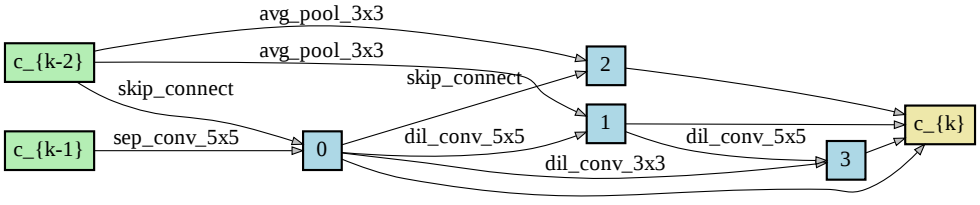

Figure 13: EnTranNAS reduction cell searched on ImageNet

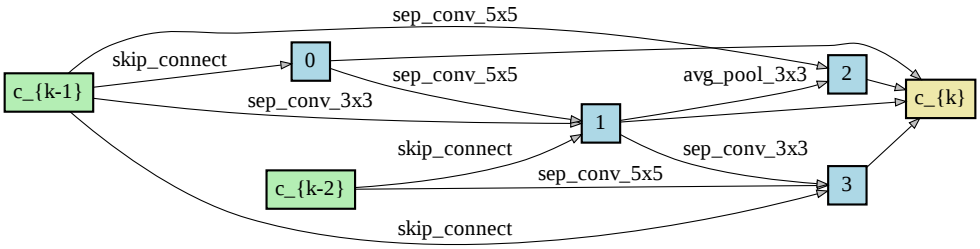

Figure 14: EnTranNAS-DST ($\lambda = 0.2$) normal cell searched on ImageNet

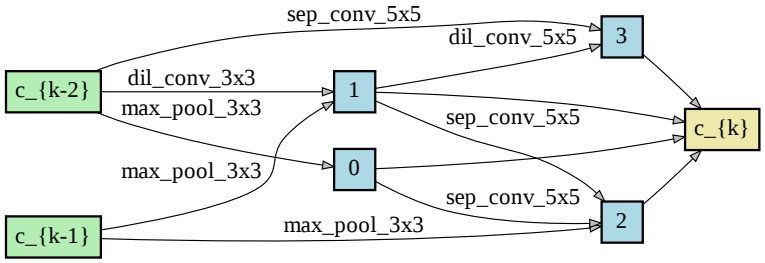

Figure 15: EnTranNAS-DST ($\lambda = 0.2$) reduction cell searched on ImageNet

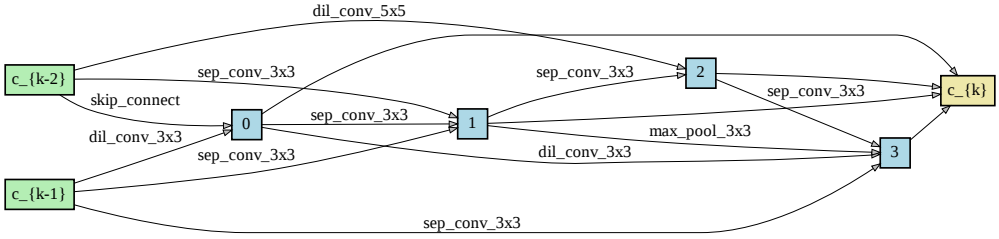

Figure 16: EnTranNAS-DST ($\lambda = 0.1$) normal cell searched on ImageNet

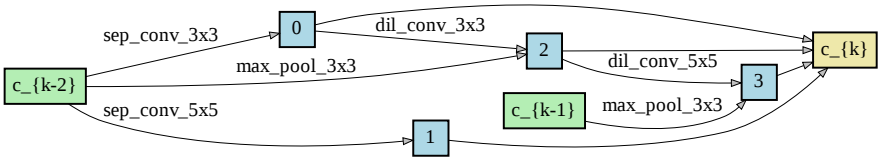

Figure 17: EnTranNAS-DST ($\lambda = 0.1$) reduction cell searched on ImageNet

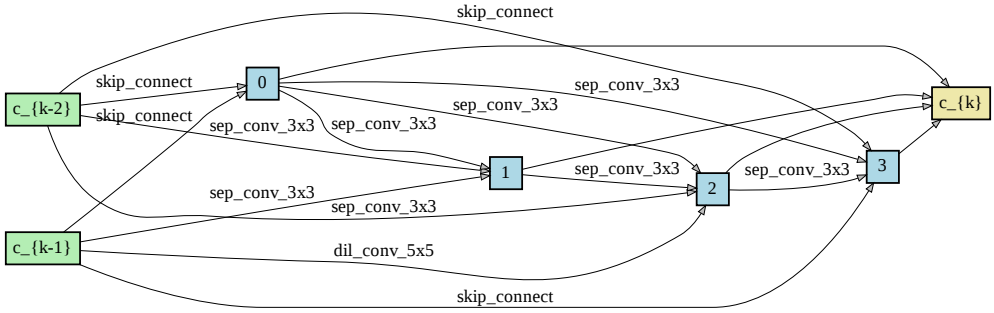

Figure 18: EnTranNAS-DST ($\lambda = 0.05$) normal cell searched on ImageNet

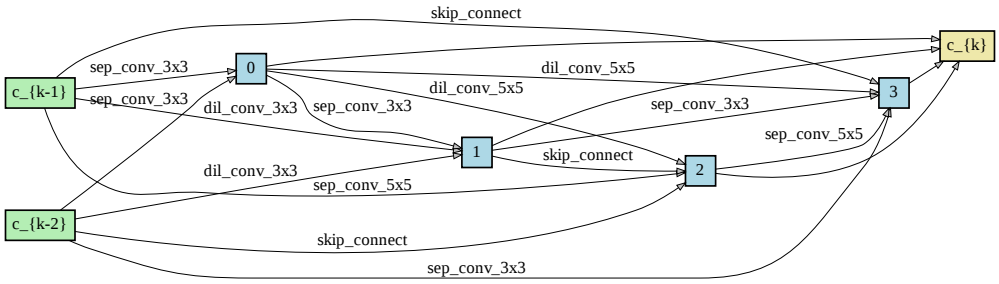

Figure 19: EnTranNAS-DST ($\lambda = 0.05$) reduction cell searched on ImageNet