Contrastive Self-supervised Neural Architecture Search

Nam Nguyen, Student Member, IEEE, J. Morris Chang, Senior Member, IEEE

Abstract—This paper proposes a novel cell-based neural architecture search algorithm (NAS), which completely alleviates the expensive costs of data labeling inherited from supervised learning. Our algorithm capitalizes on the effectiveness of self-supervised learning for image representations, which is an increasingly crucial topic of computer vision. First, using only a small amount of unlabeled train data under contrastive self-supervised learning allow us to search on a more extensive search space, discovering better neural architectures without surging the computational resources. Second, we entirely relieve the cost for labeled data (by contrastive loss) in the search stage without compromising architectures' final performance in the evaluation phase. Finally, we tackle the inherent discrete search space of the NAS problem by sequential model-based optimization via the tree-parzen estimator (SMBO-TPE), enabling us to reduce the computational expense response surface significantly. An extensive number of experiments empirically show that our search algorithm can achieve state-of-the-art results with better efficiency in data labeling cost, searching time, and accuracy in final validation.

Impact Statement—Although transfer learning has achieved many successes in computer vision tasks, finding a customized neural architecture for a specific dataset is still a promising solution for higher performance. However, the process of searching a neural architecture usually demands extensive computational resources. In this paper, we introduce a neural architecture search algorithm that leverages newly developed contrastive self-supervised learning for image representations. Thus, our proposed approach entirely alleviates the cost of data labeling in the search stage. Moreover, our approach outperforms state-of-the-art NAS algorithms in mainstream datasets in terms of predictive performance and computational expense.

Index Terms—Neural architecture search; Self-supervised learning; Sequential model-based optimization.

I. INTRODUCTION

Automated neural search algorithms have significantly enhanced the performance of deep neural networks on computer vision tasks. These algorithms can be categorized into two subgroups: (1) flat search space, where automated methods attempt to fine-tune the choice of kernel size, the width (number of channels) or the depth (number of layers), and (2) (hierarchical) cell-based search space, where algorithmic solutions search for smaller components of architectures, called cells. A single neural cell of deep neural architectures possesses a complex graph topology, which will be later stacked to form a larger network.

Although state-of-the-art NAS algorithms have achieved an increasing number of advances, several problematic factors should be considered. The main issue is that most NAS algorithms use the accuracy in validation inherited from supervised learning as the selection criteria. It leads to a computationally expensive search stage when it comes to sizable datasets. Hence, recent automated neural search algorithms usually did not search directly on large datasets but instead searching on a smaller dataset (CIFAR-10), then transferring the found architecture to bigger datasets (ImageNet). Although the performance of transferability is remarkable, it is reasonable to believe that searching directly on source data may drive better neural solutions. Besides, entirely relying on supervised learning requires the cost of data labels. It is not considered a problematic aspect in mainstream datasets, such as CIFAR-10 or ImageNet, where the labeled samples are adequate for studies. However, it may become a considerable obstacle for NAS when dealing with data scarcity scenarios. Take a medical image database as an example, where studies usually cope with expensive data curation, especially involving human experts for labeling. Hence, the remedy for such problems is vital to deal with domain-specific datasets, where the data curation is exceptionally costly. Finally, the time complexity of cell-based NAS algorithms increases when the number of intermediate nodes within cells is ascended in the prior configurated search space. Previous works show that searching on larger space brings about better architectures. However, the trade-off between performance and search resources is highly concerned (See Sect II-A for details).

We propose an automated cell-based NAS algorithm called Contrastive Self-supervised Neural Architecture Search (CS-NAS). Our work’s primary motivation is to improve the performance of the NAS algorithm by expanding the search space with the same search cost. It is clear that searching larger space potentially increases the chance to discover better solutions. To realize that goal, our first strategy is to employ the advances of self-supervised learning, which only requires a small number of samples used for the search stage to learn image representations. Besides, thanks to the nature of self-supervised learning, we entirely relieved the cost for labeled data in the search stage. It is significant to mention since, in many domain-specific computer vision tasks, the unlabeled data is abundant and inexpensive, while labeled samples are typically scarce and costly. Thus, we are now able to use a cost-efficient strategy for neural architecture search. Furthermore, we directly address the natural discrete search space of the NAS problem by SMBO-TPE, which evaluates the costly contrastive loss by computationally inexpensive surrogates. Our empirical experiments (Sect IV-B) show that architectures searched by CS-NAS on CIFAR-10 outperform hand-crafted...
architectures [1]–[3] and can achieve current state-of-the-art results.

We summarize our contributions as follows:

- We introduce a novel algorithm for cell-based neural architecture search, which relieves the expensive cost of data labeling/collecting by utilizing the arising cutting-edges of self-supervised learning for image representations.
- Our approach is the first NAS framework based on Tree Parzen Estimator (TPE) [4], which is well-designed for discrete search spaces of cell-based NAS. Moreover, the prior distribution in TPE is non-parametric densities, which allows us to sample many neural architectures to evaluate the expected improvement for surrogates, which is computationally efficient.
- We can improve search efficiency and accuracy with CS-NAS even when the search space is expanded. CS-NAS achieves state-of-the-art results with 2.66% and 25.6% test error in CIFAR-10 [5] and ImageNet [6], without any trade-off of search costs.

We organize our work as follows: Sect. III mathematically and algorithmically illustrates our proposed approach, while Sect. IV will give the experimental results and comparison with state-of-the-art NAS. Finally, we will discuss the analysis and future work in Sect. V. We have been made our implementation for public availability, hoping that there will be more research investigating our algorithm’s efficiency concerning computer vision applications.

II. RELATED WORK

A. Neural Architecture Search

State-of-the-art neural architectures require not only an extensively huge amount of time but also substantial expertise. Recently, there are an emerging growth of interest in developing automated algorithms for neural architecture design. The searched architectures have established highly competitive benchmarks in both image classification tasks [7]–[10] and object detection [7]. The best state-of-the-art neural architecture search algorithms are extremely computationally expensive despite their remarkable results. A reason for inefficient searching process is lied on the dominant approaches: reinforcement learning [11], evolutionary [10], sequential model-based optimization (SMBO) [9], MCTS [12] and Bayesian optimization [13]. For instance, searching for state-of-the-art models took 2000 GPU days under reinforcement learning framework [11], while evolutionary required 3150 GPU days [10]. Several well-established algorithmic solutions for neural architecture search have overcome expensively computational requirements without the lack of scalability: differentiable architecture search [14] enables gradient-based search by using continuous relaxation and bilevel optimization; progressive neural architecture search utilized heuristic search to discover the structure of cell’s [9]; sharing or inheriting weights across multiple child architectures [15]–[18] and predicting performance or weighting individual architecture [19], [20]. Although these latter approaches can reach state-of-the-art results with efficiency concerning searching time, they may be affected by the inherited issue from gradient-based approaches, local minimum solution. Apart from that, all state-of-the-art neural architecture search algorithms require full knowledge of training data. Thus, the searching process is frequently performed on a smaller dataset (e.g., CIFAR-10 or CIFAR-100), then the discovered architecture will be trained on a bigger dataset (e.g., ImageNet) to evaluate the transferability.

B. Self-supervised Learning for Learning Image Representation

Self-supervised learning has established extremely remarkable achievements in natural language processing [21]–[23]. Hence learning visual representation has drawn great attention with a large scale of literature that has explored the application of self-supervised learning for video-based and image-based classification. Within the scope of this paper, we only focus on self-supervised learning for an image classification task. Generally, mainstream approaches for learning image representations can be categorized into two classes: generative, where input pixels are generated or modeled [24]–[27]; and discriminative, where networks are trained on a pretext task with a similar loss function. The key idea of discriminative learning for visual representations is the designation of pretext task, which is created by pre-assigned targets and inputs derived from unlabeled data: distortion [28], rotation [29], patches or jigsaws [30], colorization [31]. Moreover, the most recent research interests of self-supervised learning have been drawn from contrastive learning in the input latent space [32]–[36], which promisingly showed comparable achievement to supervised learning. Besides, the most impressive achievement from contrastive self-supervised learning is that there is only a small proportion of training data (1% to 10%) required for establishing the same state-of-the-art results with supervised learning [35], [36].

In this work, we adopt the contrastive self-supervised learning framework from Pretext-Invariant Representation Learning (PIRL) for neural architecture search task, which will be discussed hereafter in Sect. III-B.

III. METHODOLOGY

We will generally describe two main fundamental components of our study: neural architecture search and contrastive self-supervised visual representation learning in Sect. III-A and Sect. III-B. Finally, we will formulate the Contrastive Self-supervised Neural Architecture Search (CS-NAS) as a hyper-parameters optimization problem and establish its solution by tree-structured Parzen estimator in Sect. III-C.

A. Neural Architecture Search

1) Neural Architecture Construction: We employ the architecture construction from [7], [11], where searched cell are stacked to form the final convolutional network. Each cell can be represented as a directed acyclic graph of \( N + 3 \) nodes, which are the feature maps and each corresponding operation \( o^{(i,j)} \) forms directed edges \((i,j)\). Following [7]–[10], we assume that a single cell consists of two inputs (outputs of
Fig. 1: A graph representation of cell architecture. Left figure: dashed lines represent connections between nodes via a choice of operations; solid lines depict fixed connections. Right figure: the adjacency matrix of the corresponding cell architecture. All shaded entries are zeros since it is impossible to establish corresponding connections. Each $\theta_i$ is a random variable representing a choice of operation.

the two previous layers $C_{k-2}$ and $C_{k-1}$), one single output node $C_k$ and $N$ intermediate nodes. Latent representations in intermediate nodes are included, which computed as in [14]:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^i)$$

The generating set $O = \{ o^{(i,j)} \}$ for operations between nodes is employed from the most selected operators in [7], [10], [11], where includes seven non-zero operations: $3 \times 3$ and $5 \times 5$ dilated separable convolution, $3 \times 3$ and $5 \times 5$ separable convolution, $3 \times 3$ max pooling and average pooling, identity and zero operation (skip-connection). We retain the intermediate nodes with a set of operators $\{C_k\}$ are separable convolution, max pooling, and reduction cells. Therefore the total number of viable cells (normal and reduction) is enormously expanded by a factor of $N!$ when increasing the number of intermediate nodes from $N$ to $N+1$. Consequently, state-of-the-art NAS can only achieve a low search time (in days) when using $N = 4$, while ascending $N$ to 5 usually takes a much longer search time, up to hundreds of days. Our approach treats the high time complexity when expanding the search space by (1) using only a small proportion of data under contrastive learning for visual representations and (2) evaluating the loss by surrogate models, which requires much less computational expense. Details of these methods will be discussed in the next sections.

B. Contrastive Self-supervised Learning

We employ recent contrastive learning framework in PIRL [37], allowing multiple views on a sample. Let $D = \{x_i\}_{i=1}^{|D|}$ be a train set (for searching) and $F = \{f_\theta(.)\}_{\theta \in \Theta}$ be a set of all neural architecture candidates:

- Each sample $x_i \in \mathbb{R}^{H \times W \times 3}$ is first taken as input of a stochastic data augmentation module, which results in a set of correlated views $x_i^t = \{x_i^{(1)}, \ldots, x_i^{(M)}\}$. Within the scope of this study, three simple image augmentations are applied sequentially for each data sample, including: random/center cropping, random vertical/horizontal flipping and random color distortions to grayscale. The set $X = \{x_i\} \cup x_i^t$ is called positive pair of sample $x_i$.
- Each candidate architecture $f_\theta(.)$ is used as a base encoder to extract visual representations from both original sample $x_i$ and its augmented views $x_i^t$. We use the same multilayer perceptron $g(.)$ at the last of all neural candidates, projecting its feature maps of original $x_i$ image under $f_\theta(.)$ into a vector $\mathbb{R}^p$. For augmented views, another intermediate MLP $l(.) : \mathbb{R}^{M \times p} \rightarrow \mathbb{R}^p$ is applied on concatenation of $\{f_\theta(x_i^{(m)})\}$ for $m \in M$. We denote the representing vectors of original image and augmented views as $z_i$ and $z_i^t$, respectively.

We also use the cosine similarity as the similarity measurement as in [36], [37], yielding $s(u, v) = u^T v / ||u|| \cdot ||v||$. Each minibatch of $K$ instances is randomly sampled from $D$, and each pair $(x_i, x_j)$ is passed through the model $f_\theta(.)$ to get their feature representations $z_i$ and $z_j$.
giving $M \times K$ data points. Similar to [38], a positive pair is corresponding to in-batch negative examples, which are other $M(K - 1)$ augmented samples, forming a set of negative sample $\mathcal{X}'$. Similarly, each negative samples are extracted visual representations as $z_i' = g(f_\theta(x_i'))$. Following [37], we compute the noise contrastive estimator (NCE) of a positive pair $x_i$ and $x_i'$ using their corresponding $z_i$ and $z_i'$, given by

$$\ell(z_i, z_i') = \frac{\exp\left(\frac{s(z_i, z_i')}{\tau}\right)}{\exp\left(\frac{s(z_i, z_i')}{\tau}\right) + \sum_{z_i' \in \mathcal{X}'} \exp\left(\frac{s(z_i, z_i')}{\tau}\right)}. \quad (1)$$

The estimators are used to minimize the loss:

$$\mathcal{L}_{\text{NCE}}(x_i, x_i') = -\log \ell(z_i, z_i') - \sum_{z_i' \in \mathcal{X}'} \log [1 - \ell(z_i', z_i')] \quad (2)$$

The NCE loss maximizes the agreement between the visual representation of the original image $x_i$ and its augmented views $x_i'$, together with minimizes the agreement between $x_i$ and $x_i'$. We use memory-bank approach in [37], [39] to cache the representations of all samples in $\mathcal{D}$. The representation $r_x$ in memory-bank $\mathcal{M}$ is the exponential moving average of $z_i$ from prior epochs. The final objective function for each neural candidate is a convex function of two losses as in Equation 2:

$$\mathcal{L}_\theta(x, x') = \lambda \mathcal{L}_{\text{NCE}}(r_x, z') + (1 - \lambda) \mathcal{L}_{\text{NCE}}(r_x, z) \quad (3)$$

Finally, these loss values establish the scoring criteria for neural architectures under sequential model-based optimization, which will be discussed in the next section.

C. Tree-structured Parzen Estimator

As mentioned in the previous sections, we aim to search on a larger space in order to discover a better neural solutions. Nevertheless, the main difficulty when expanding the search space due to the exponential surge in the time complexity. Thus, we are motivated to study an optimization strategy that might reduce the computational cost. We employ the sequential model-based optimization (SMBO), which has been widely used when the fitness evaluation is expensive. This optimization algorithm can be a promising approach for cell-based neural architecture search, since current state-of-the-art NAS algorithms use the loss in validation as the fitness function, which is computational expensive. The evaluation time for each neural candidate tremendously surges when the number of training samples or sample’s resolution increases. In the current literature, PNAS [9] is the first framework which apply SMBO for cell-based neural architecture search. In contrast to their approach, where the fitness function is in-validation accuracy, we model the contrastive loss in Equation 3 by a surrogate function $S(\cdot)$, which requires less computational expenses. Specifically, a large number of candidates will be drawn to evaluate the expected improvement at each iteration. The surrogate function approximated the contrastive loss over the set of drawn points, resulting in cheaper computational cost. Mathematically, the optimization problem is formulated as

$$\theta^* = \arg \min_{\theta \in \Theta} \mathcal{L}_\theta(x, x')$$

The SMBO framework is summarized in Algorithm 1.

**Algorithm 1 Sequential Model-Based Algorithm [4]**

Given $(\mathcal{L}_\theta, \mathcal{M}_0, I, S)$:

1) Initialize history $\mathcal{H} \leftarrow \emptyset$

2) For iteration $i = 1$ to $I$:

- $\theta^* \leftarrow \arg \min_{\theta \in \Theta} S(\theta, M_{i-1})$
- Evaluate $\mathcal{L}_\theta(x, x')$
- Update $\mathcal{H} \leftarrow \mathcal{H} \cup (\theta^*, \mathcal{L}_\theta(x, x'))$
- Fit $M_i$ to $\mathcal{H}$

3) Return $\mathcal{H}$

which attempts to optimize the Expected Improvement (EI) criterion [4]. Given a threshold value $t^*$, EI is the expectation under an arbitrary model $M$, that $\mathcal{L}_\theta(x, x')$ will exceed $t^*$.
Mathematically, we have:

$$\text{EI}_t := \int_{-\infty}^{\infty} \max(t^* - t, 0) p_M(t|\theta) dt. \quad (4)$$

While Gaussian-process approach models $p(t|\theta)$, the tree-structured Parzen estimator (TPE) models $p(\theta|t)$ and $p(t)$, the it decomposes $p(\theta|t)$ to two density functions:

$$p(\theta|t) = \begin{cases} l(\theta) & \text{if } t < t^* \\ g(\theta) & \text{if } t \geq t^* \end{cases}, \quad (5)$$

where $l(\theta)$ is the density function of candidate architectures corresponding to $\{\theta^{(i)}\}$, such that $L^{(i)}(x, x^t) < t^*$ and $g(\theta)$ is formed by the remaining architectures. As a result, the EI in Equation 4 is reformed as:

$$\text{EI}_t(\theta) = \int_{-\infty}^{t^*} (t^* - t) p(t|\theta) dt$$
$$= \int_{-\infty}^{t^*} (t^* - t) \frac{p(\theta|t)p(t)}{p(\theta)} dt \propto \left( \gamma + \frac{g(\theta)}{l(\theta)} (1 - \gamma) \right)^{-1}, \quad (6)$$

where $\gamma$ denotes $p(t < t^*)$. The tree structure in TPE allows us to draw multiple candidates according to $l(\cdot)$ and then evaluate them based on $g(\theta)/l(\theta)$.

IV. EXPERIMENTS AND RESULTS

Our experiments on each dataset include two phases, neural architecture search (Sect IV-A) and architecture evaluation (Sect IV-B). In the search phase, we used only 10% of unlabeled data (5000 samples from CIFAR-10 and $\approx 12k$) to search for neural architectures having the lowest contrastive loss mentioned in Equation 3 by CS-NAS. The best architecture is scaled to a larger architecture in the validation phase, then trained from scratch on the train set and evaluated on a separate test set.

A. Architecture Search for Convolution Cells

We initialize our search space by the operation generating set $O$ as in Sect III-A1, which has been obtained by the most frequently chosen operators in [7], [9], [10], [14]. Each convolutional cell includes two inputs $C_{k-2}$ and $C_{k-1}$ (feature maps of two previous layers), a single concatenated output $C_k$ and $N$ intermediate nodes.

We create two searching spaces for CIFAR-10, denoted as CS-NAS$_{N=4}$ and CS-NAS$_{N=5}$, which are corresponding to the number of intermediate nodes $N$. With $N = 4$, a configurable vector $\theta$ representing a neural architecture have the length of $28$ ($\theta_{\text{normal}} = \theta_{\text{reduced}} = 14$), resulting in a search space of size $(8!^4)^2 \approx 2 \times 10^{25}$. We expand our search space by adding a single intermediate node, in the hope of finding a better architecture. $N = 5$ is corresponding to configurable vector $\theta$ of length 40, which tremendously surges the total number of possible architectures to $10^{36}$. Experiments involved ImageNet only use the latter search space with $N = 5$.

We configure our contrastive self-supervised learning by two augmented views ($M = 2$) for each sample with methods mentioned in Sect. III-B, producing $2 \times (K - 1)$ negative examples for each data instance in a minibatch of size $K$. We also investigate the effect of $M$ on the search efficiency, which we will discuss in detail in Sect. B. The other two hyperparameters $\tau$ and $\lambda$ in Equation 1 and Equation 2 are taken from the best experiment in [37], where $\tau = 0.07$ and $\lambda = 0.5$. Besides, MLPs $g(\cdot)$ and $l(\cdot)$ project encoded convolutional maps to a vector of size $p = 128$. Although we expected a minor impact of the above hyper-parameters, we will leave this tuning problem for further study.

We initialize the same prior density for each component
of $\theta$, which is that all operations have the same chance to be picked up at a random trail. 20 random samplings start TPE, then 20K sample points are suggested to compute the expected improvement in each subsequent trial. We select only 20% of best-sampled points having the greatest expected improvement to estimate next $\theta$. We also observed that the number of starting trials is insensitive to the searching results while increasing the number of sampling points for computing expected improvement and lowering their chosen percentage ameliorate the searching performance (lower the overall contrastive loss).

The experiment details will be given hereafter in Sect A.

B. Neural Architecture Evaluation

We select the architecture having the best score from the searching phase and scale it for the validation phase. Within this paper’s scope, we only scale the searched architecture to the same size as baseline models in the literature ($\approx 3$M). All weights learned from the searching phase had been discarded before the validation phase, where the chosen architecture is trained from scratch with random weights.

C. Results Analysis

Regarding CIFAR-10, we report the performance of architecture searched by CS-NAS in Table I. It is remarkable to highlight that CS-NAS $N=4$ achieved a slightly better result than DARTS with $16\times$ faster (0.25 in comparison to 4), even though these algorithms share the same search space complexity. Moreover, CS-NAS $N=5$ can reach comparable results with AmoebaNet and NASNet in a tremendously less computational expense (1 vs. 3150 and 2000, respectively).

The results of CS-NAS on ImageNet are reported in Table II. Instead of transferring architecture from CIFAR-10, we directly search for the best architecture using 10% of unlabeled samples from ImageNet (list of the image can be found in [36]. The performance of the network found by CS-NAS appears to outperform most of the state-of-the-art NAS methods (except AmoebaNet and PNAS), reaching 25.8% in test error. However, CS-NAS possesses a significantly higher search efficiency inaccuracy, search cost, and ignorance of data labels.

V. Conclusion

We have introduced CS-NAS, an automated neural architecture search that completely alleviates the expensive cost of data labeling. Furthermore, CS-NAS performs searching on natural discrete search space of NAS problem via SMBO-TPE, enabling competitive/matching results with state-of-the-art algorithms.

There are many directions to conduct further study on CS-NAS. For example, computer vision tasks, which involve medical images, are usually considered to lack training samples. This task requires substantially expensive data curation cost, including data gathering and labeling expertise. Another possible CS-NAS improvement is investigating further baseline self-supervised learning methods, which potentially ameliorates current CS-NAS benchmarks.

ACKNOWLEDGMENTS

Effort sponsored in part by United States Special Operations Command (USSOCOM), under Partnership Intermediary Agreement No. H92222-15-3-0001-01. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon.

REFERENCES

[1] 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, pages 4700–4708, 2017.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Special Operations Command.
TABLE I: The performance (in term of test error) of state-of-the-art NAS algorithms on CIFAR-10. The search cost includes only searching time by SMBO-TPE algorithm, excluding the final architecture evaluation cost (See B).

| Neural Architecture                  | Test Error (%) | Params (M) | Search Cost (GPU days) | # Ops | Search Strategy |
|--------------------------------------|----------------|------------|------------------------|-------|-----------------|
| DenseNet-BC [1]                      | 3.46           | 25.6       | -                      | -     | Manual          |
| VGG11B (2×) [2]                      | 3.60           | 42.0       | -                      | -     | Manual          |
| ResNet-1001 [3]                      | 4.62           | 10.2       | -                      | -     | Manual          |
| AmoebaNet-A + cutout [10]            | 3.12           | 3.1        | 3150                   | 19    | Evolution       |
| AmoebaNet-B + cutout [10]            | 2.55 ± 0.05    | 2.8        | 3150                   | 19    | Evolution       |
| Hierarchical evolution [10]          | 3.15 ± 0.12    | 15.7       | 300                    | 6     | Evolution       |
| BlockQNN [40]                        | 3.54           | 39.8       | 96                     | 8     | RL              |
| NASNet-A [7] + cutout                | 2.65           | 3.3        | 2000                   | 13    | RL              |
| ENAS [17] + cutout                   | 2.89           | 4.6        | 0.5                    | 6     | RL              |
| DARTS (1st order) [14] + cutout      | 3.00 ± 0.14    | 3.3        | 1.5                    | 7     | Gradient-based  |
| DARTS (2nd order) [14] + cutout      | 2.76 ± 0.09    | 3.3        | 4                      | 7     | Gradient-based  |
| PC-DARTS [41] + cutout               | 2.57 ± 0.07    | 3.6        | 0.1                    | 7     | Gradient-based  |
| P-DARTS [42] + cutout                | 2.50           | 3.4        | 0.2                    | 7     | Gradient-based  |
| SNAS (moderate) [43] + cutout        | 2.85 ± 0.02    | 2.8        | 1.5                    | 7     | Gradient-based  |
| BayesNAS [44] + cutout               | 2.81 ± 0.04    | 3.4        | 0.2                    | -     | Gradient-based  |
| PNAS [9]                             | 3.41 ± 0.09    | 3.2        | 225                    | 8     | SMBO            |
| CS-NAS  \(N = 4\) + cutout \(†\)    | \(2.71 \pm 0.11\) | 3.5       | 0.25                   | 7     | SMBO-TPE        |
| CS-NAS  \(N = 5\) + cutout \(†\)    | \(2.66 \pm 0.07\) | 3.4       | 1                      | 7     | SMBO-TPE        |

\(†\) Results based on 10 independent runs.

TABLE II: The performance (in term of test error) of state-of-the-art NAS algorithms on ImageNet.

| Neural Architecture                  | Test Err. (%) | Params (M) | Search Cost (GPU days) | Search Strategy |
|--------------------------------------|---------------|------------|------------------------|-----------------|
| Inception-v1 [45]                    | 30.2 (10.1)   | 6.6        | 1448                   | Manual          |
| Inception-v2 [46]                    | 25.2 (9.22)   | 11.2       | -                      | Manual          |
| MobileNet [47]                       | 29.4 (10.5)   | 4.2        | 569                    | Manual          |
| ShuffleNet (2×)-v1 [48]              | 26.4 (10.2)   | 5            | 524                    | Manual          |
| ShuffleNet (2×)-v2 [49]              | 25.1 (--)     | 5            | 591                    | Manual          |
| EfficientNet-B0 [50]                 | 23.7 (6.8)    | 5.3        | -                      | Manual          |
| NASNet-A [7]                         | 26.0 (8.4)    | 5.3        | 564                    | 2000            | RL              |
| NASNet-B [7]                         | 27.2 (8.7)    | 5.3        | 488                    | 2000            | RL              |
| NASNet-C [7]                         | 27.5 (9.0)    | 4.9        | 558                    | 2000            | RL              |
| AmoebaNet-A [10]                     | 25.5 (8.0)    | 5.1        | 555                    | 3150            | Evolution       |
| AmoebaNet-B [10]                     | 26.0 (8.5)    | 5.3        | 555                    | 3150            | Evolution       |
| AmoebaNet-C [10]                     | 24.3 (7.6)    | 6.4        | 570                    | 3150            | Evolution       |
| PNAS [9]                             | 25.8 (8.1)    | 5.1        | 588                    | 255             | SMBO            |
| DARTS (2nd order) [14]               | 26.7 (8.7)    | 4.7        | 574                    | 4               | Gradient-based  |
| CS-NAS  \(N = 5\)                    | \(25.8 (8.3)\) | 5.1        | 590                    | 2.5             | SMBO-TPE        |
cells are located in 1/2 and 2/3 of the depth, which has stride two for all operations linked to the input node.

We construct a large network for the CIFAR-10 dataset, including 24 layers with 42 initial num channels. The train setting is employed from existing studies [7], [9]–[11], [14], offering more regularization on training, which includes: cutout [51] of length 16, linearly path drop out with probability 0.2 and auxiliary classifier (located in 2/3 maximum depth of the network) with weight 0.4. We train the network for 1200 epochs using batch size 128. The chosen optimizer is momentum SGD with learning rate $\eta = 0.025$, momentum = 0.9, weights decay $3 \times 10^{-4}$ and gradient clip of 5. The entire training process takes three days on one single GPU.

Regarding the ImageNet dataset, the input resolution is set to be $224 \times 224$, and the allowed number of multiply-add operations is less than 600, which is restricted for mobile settings. We train a network having 20 cells and 54 initial channels for 600 epochs with a batch size of 128. The learning rate is set at $\eta = 0.1$ with decay rate 0.97 and decay period of 2.5. We use the same auxiliary module as the invalidation phase of CIFAR-10. We set the SGD optimizer at momentum of 0.9, and weights decay $3 \times 10^{-5}$.

C. Efficiency Assessment of Search Strategy

We assess the effectiveness of TPE samplers by comparison to basic random search using only the CIFAR-10 dataset. Under the same search ($N = 5$) and validation set, the neural architecture found by random search strategy (Figure 6) achieves 2.98% test error.

![Cell founded by random search](image-url)