CSNAS: Contrastive Self-Supervised Learning Neural Architecture Search Via Sequential Model-Based Optimization

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

***Abstract*—This article proposes a novel contrastive self-supervised neural architecture search (NAS) algorithm, 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 allows 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, enabling us to significantly reduce the computational expense response surface. 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 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. This article introduces 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 regarding predictive performance and computational expense.**

***Index Terms*—Neural architecture search (NAS), 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 more minor 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 more extensive network.

Although state-of-the-art (SOTA) neural architecture search (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 do not search directly on large datasets but instead search on a smaller dataset (CIFAR-10), then transferring the found architecture to more enormous 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 well-collected 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, cell-based NAS algorithms' time complexity 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 tradeoff between predictive performance and search resources is highly considerable.

We propose an automated cell-based NAS algorithm called contrastive self-supervised neural architecture search (CSNAS). Our work's primary motivation is to offer high-performance neural architectures by expanding the search space without any trade-off of search cost. It is clear that searching larger space potentially increases the chance to discover better solutions. We

Manuscript received June 9, 2021; revised September 4, 2021; accepted October 17, 2021. Date of publication October 29, 2021; date of current version July 21, 2022. This work was supported in part by United States Special Operations Command (USSOCOM), under Partnership Intermediary Agreement H92222-15-3-0001-01. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon. This paper was recommended for publication by Associate Editor C. L. Philip Chen upon evaluation of the reviewers' comments. *(Corresponding author: Nam Nguyen.)*

The authors are with the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: namnguyen2@usf.edu; chang5@usf.edu).

Digital Object Identifier 10.1109/TAI.2021.3121663

realize that goal by employing the advances of self-supervised learning (SL), which only requires a small number of samples used for the search stage to learn image representations. Besides, thanks to the nature of SL, 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 NAS. Furthermore, we directly address the natural discrete search space of the NAS problem by sequential model-based optimization via the tree-parzen estimator (SMBO-TPE), which evaluates the costly contrastive loss by computationally inexpensive surrogates. We have made our implementation for public availability, hoping that there will be more research investigating our algorithm's efficiency concerning computer vision applications. The code for implementation can be found at GitHub repo:[1]

II. Related Work

In this section, we first introduced SOTA supervised NAS in Section II-A. Second, the overview of SL will be discussed in Section II-B. We then outline SOTA self-supervised NAS algorithm in Section II-C. Finally, we provide our point-of-view on the merits of NAS algorithms in Section II-D, followed by a highlight to distinguish our CSNAS from other approaches. The summary of contribution will also be given at the end of the section.

A. Supervised NAS

We briefly outline the advantages of SOTA supervised NAS algorithms in this section. The discovered architectures searched by supervised NAS algorithms have established highly competitive benchmarks in both image classification tasks [1]–[4] and object detection [1]. The best SOTA supervised NAS algorithms are extremely computationally expensive despite their remarkable results. A reason for inefficient searching process is due to the dominant approaches: reinforcement learning [5], evolutionary algorithms [4], sequential model-based optimization (SMBO) [3], Monte Carlo Tree Search (MCTS) [6], and Bayesian optimization [7]. For instance, searching for SOTA models took 2000 GPU days under reinforcement learning framework [5], while evolutionary NAS required 3150 GPU days [4]. The main reason for such extreme computational expense can be attributed to the selection of dataset for the search phase. In the pioneers NAS algorithms, the dataset used for both search and evaluation phase is typically ImageNet [8], which includes a considerable large number of samples. As a consequence, the computational burden for the search phase is tremendously massive. Directly tackling this issue, following NAS algorithms introduce the usage of proxy dataset for the search phase in order to reduce the computational expense. The common choice for proxy of ImageNet is CIFAR-10 [9] in the NAS-related study. In particular, we search the optimal neural architectures on CIFAR-10 and investigate the transferability to ImageNet. As a result, the search cost for NAS algorithms significantly reduced to an affordable GPU hours. For example, RelativeNAS [10] introduces an efficient population-based NAS algorithm which can discover high predictive performance neural architecture within 0.4 GPU days. Moreover, several well-established algorithmic solutions for NAS using proxy dataset for the search phase have overcome expensive computational requirements without the lack of scalability: Differentiable architecture search [11]–[13] enables gradient-based search by using continuous relaxation and bilevel optimization; progressive NAS utilized heuristic search to discover the structure of cells [3]; sharing or inheriting weights across multiple child architectures [14]–[17]; and predicting performance or weighting individual architecture [18], [19]. Although these latter approaches can reach SOTA results with efficiency concerning searching time, they may be affected by the inherited issue from gradient-based approach, which finds the local minimum solution. Apart from that, most SOTA NAS 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), and then the discovered architecture will be trained on a bigger dataset (e.g., ImageNet) to evaluate the transferability. Although we can relax the constraint of using the same dataset in the search phase and evaluation phase by transferring the learned architecture, it is reasonable to believe that training with the constrain may result in better solutions.

B. Self-Supervised Learning for Learning Image Representation

SL has established extremely remarkable achievements in natural language processing [20]–[22]. Hence, learning visual representation has drawn great attention with a large scale of literature that has explored the application of SL for video-based and image-based classification. Within the scope of this article, we only focus on SL 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 [23]–[26]; 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 preassigned targets and inputs derived from unlabeled data: distortion [27], rotation [28], patches or jigsaws [29], and colorization [30]. Moreover, the most recent research interests of SL have been drawn from contrastive learning in the input latent space [31]–[35], which promisingly showed comparable achievement to supervised learning. Grill *et al.* [34]–[36] provide convincing evidence of the capacity to learn image representations of contrastive SL, which only require a small proportion of training data (1%–10%) on linear evaluation to reach supervised learning performance. This work also studied the effect of contrastive self-supervised learning (CSSL) of different backbone architectures, which empirically shows that the predictive performance consistently gains when scaling

[1][Online]. Available: "https://github.com/namnguyen0510/CSNAS." Moreover, the detailed hyperparameter setting is given in Sections A1 and A2

backbone networks. Moreover, a solid theoretical analysis of SL (or self-training) is introduced by [37]. Under simplified but practical assumptions that 1) a subset of low-confident samples must expand to a neighborhood with higher confidence concerning the subset and 2) neighborhoods of samples from different classes are minimally overlapped, SL on unlabeled data with input-consistency regularization will result in higher accuracy in comparison to labeled data.

C. *Self-Supervised Learning NAS*

First, we would like to give a simplified design for self-supervised NAS algorithms, including two main components: 1) choices of the SL method and 2) search strategy. In the current literature, SSNAS [38] and UnNAS [39] also introduced self-supervised NAS algorithms that leverage the DART algorithm for search strategy. Moreover, the choice of the SL method of SSNAS is SimCRL, while UnNAS utilized invariant pretext tasks. It is worth mentioning that SL based on pretext tasks is distinguishable from CSSL SimCRL. The former approach assigns pretext task labels to unlabeled data (rotation angle, colorization type, suffer and solve jigsaw puzzles), and then trains neural architecture based on these generated labels under supervised learning using the conventional loss function of supervised learning. To some extent, we still observe supervised learning within SL based on pretext tasks since it can be considered as supervised training. On the other hand, CSSL trains neural architectures to maximize the agreement between positive pairs (augmented views from the same instance) while minimizing the agreement between negative pairs (augmented view from different instances). Hence, CSSL eliminates the trait of supervised learning by noise contrastive loss while achieving much better results (very closed to supervised learning) in comparison to pretext task SL [34]–[36].

SSNAS is inspired by the fact that all architectures in the search space are overfitted on the training data. Thus, their selection criteria are based on the architecture's generalization over the training data. To realize their goal, they used the margin-based search, which involves two splits of training data. Each neural architecture candidate is trained on one split to increase the margin between samples. The selected candidate is the neural architecture maintaining the most significant distance between instances.

On the other hand, UnNAS attempted to answer a fascinating question: "Are labels necessary for Neural Architecture Search?" The work searched neural architecture by DART algorithm using the pretext-task-assigned dataset, including rotation, colorization, and solving a jigsaw puzzle. Both studies provide compelling experimental results, showing that self-supervised NAS can search for high-performance neural solutions while relieving the cost of data annotations in the search stage.

D. *Merits of NAS Algorithms and Contribution of CSNAS*

In this section, we will discuss the merits and evaluation criteria for NAS algorithms. [4] and [5] are frontiers in supervised NAS algorithms, which successfully searched neural architecture while remaining the constraint of searching data and evaluating data. However, they need to make a massive tradeoff with thousands of searching hours. Thus, such an approach's disadvantage is that it is nearly impossible to implement with limited computational resources. With the growth of research interest from the field, many subsequence works successfully reduce the computational requirements while maintaining high predictive power [11], [16], [40], [41]. However, such algorithms still have several weaknesses. For example, DART and subsequence approaches such as PC-DART and P-DART result in different cell architectures even though they share the same initial search space. Thus, DART could find a local optimum for the NAS problem. However, we cannot deny that DARTS offers us the first efficient gradient-based NAS algorithms in terms of accuracy and searching resources.

Moreover, recent work such as SSNAS and UnNAS successfully relieve the cost of labeled data in the search stage while achieving comparable results with supervised NAS. Any supervised NAS algorithm can perform self-supervised NAS if we take the labels for granted in the search stage. Hence, it is not reasonable to compare supervised NAS and self-supervised NAS based on the test accuracy. Intuitively, supervised NAS algorithms should achieve a better predictive performance since they have access to the train set's full knowledge, which includes data with annotations. However, we are more often than not dealing with computer vision problems involving data-scarcity scenarios, in which the cost for data annotations is expensive while unlabeled data are much more abundant. In this case, self-supervised NAS algorithms surpass every supervised NAS algorithm since they can leverage the additional unlabeled samples, which are entirely taken for granted by supervised NAS algorithms. Nevertheless, SL NAS also inherits several issues from its backbone training procedure. First, there could be a loss of information while using pretext tasks labels or contrastive noise estimators compared to the conventional loss of supervised learning. Recent self-supervised NAS algorithms UnNAS and SSNAS show a minimal gap of such loss on a nicely collected database (CIFAR-10 and ImageNet), including well-represented training samples with equal distribution. However, such loss of information is challenging to indicate since it is data-dependent universally. We can approximate the loss of information by comparing supervised NAS algorithms on a specific database. Second, the time complexity of search under CSSL is more considerable than supervised learning search since it requires multiple inputs to evaluate the contrastive loss. This issue does not appear in pretext task SL from UnNAS since it trains a given neural architecture under supervised learning with labels generated by given pretext tasks. However, the increase in the time complexity due to CSSL is extremely minor in comparison to the whole time complexity of NAS algorithms, since we only need to perform backpropagation on a sole neural candidate

We distinguish our proposed CSNAS from other self-supervised NAS algorithms by two main points. First, we leverage a different contrastive SL PIRL [36] for learning image representations within the search phase. SSNAS used SimCRL, which benefits from an extensive training batch size (initially 4096 samples per mini-batch) and a long searching process.

Thus, it is a clear obstacle for practical purposes due to its computationally expensive resources. In contrast, by using a memory-bank, PIRL allows SL with computationally affordable batch size while remaining the adequate negative samples per mini-batch ($4\times$ smaller compared to SimCRL). Second, it is clear that there will be an increase in the searching time if we directly employ supervised NAS's search strategy for self-supervised NAS. Hence, we compromise the surge by SMBO with a tree-structured Parzen estimator (TPE). Our empirical experiments (see Section IV-B) show that architectures searched by CSNAS on CIFAR-10 outperform hand-crafted architectures [42]–[44] and can achieve high predictive performance in comparison to SOTA NAS algorithms. We also investigate our proposed approach's domain adaption on a medical imaging database under a data scarcity scenario. The case study provides us a deeper insight into the effectiveness of contrastive learning and gives us a promising result of CSNAS in practice. We summarize our contributions as follows.

1) We introduce a novel algorithm for self-supervised NAS. CSSL enables efficient search with access to a small proportion of data without annotations. Thus, our proposed approach relieves the computational cost for the search stage, including searching time and labeling cost.
2) Our approach is the first NAS framework based on TPE [45], which is well-designed for discrete search spaces of cell-based NAS. Moreover, the prior distribution in TPE is nonparametric densities, which allows us to sample many neural architectures to evaluate the expected improvement (EI) for surrogates, which is computationally efficient. Moreover, surrogate models' usage reduces the expensive cost of true loss function during the search phase. Thus, we can improve search efficiency and accuracy with CSNAS even when the search space is expanded. CSNAS achieves SOTA results with 2.66% and 25.6% test error in CIFAR-10 [9] and ImageNet [46], without any trade-off of search costs.
3) We also evaluate the domain adaption ability of our proposed approach on a medical imaging database involving skin lesion classification. Our searched neural architecture on a limited dataset without annotations is well designed and well customized for skin lesion classification, which outperforms SOTA architectures under transfer learning, reaching 88.68% of accuracy in the test set.

The rest of this article is organized as follows. Section III mathematically and algorithmically illustrates our proposed approach, while Section V-B will give the experimental results and comparison with SOTA NAS on CIFAR-10 and ImageNet. Moreover, we report a detailed analysis of a case study involving skin lesions in Section V. Finally, Section VII concludes this article.

III. Methodology

We will generally describe two main fundamental components of our study: NAS and contrastive self-supervised visual representation learning in Sections III-A and III-B, respectively. Finally, we formulate the CSNAS as a hyperparameters optimization problem and establish its solution by TPE in Section III-C.

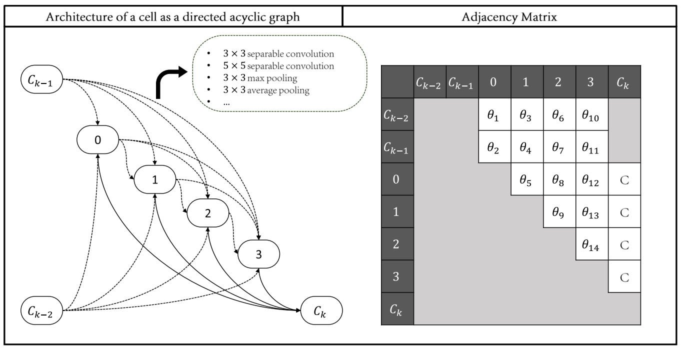
Architecture of a cell as a directed acyclic graph
Adjacency Matrix
3 × 3 separable convolution
5 × 5 separable convolution
3 × 3 max pooling
3 × 3 average pooling
...
C_{k-1}
C_{k-2}
0
1
2
3
C_k
θ_1 θ_3 θ_6 θ_{10}
θ_2 θ_4 θ_7 θ_{11}
θ_5 θ_8 θ_{12} C
θ_9 θ_{13} C
θ_{14} C
C

Fig. 1. 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 θ_i is a random variable representing a choice of operation.

A. Neural Architecture Search

1) Neural Architecture Construction: We employ the architecture construction from [1], [5], where searched cells 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) (see Fig. 1). Following [1]–[4], we assume that a single cell consists of two inputs (outputs of 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 are computed as in [11]

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

The generating set $\mathcal{O} = \{o^{(i,j)}\}$ for operations between nodes is employed from the most selected operators in [1], [4], [5], which includes seven non-zero operations: 3×3 and 5×5 dilated separable convolution, 3×3 and 5×5 separable convolution, 3×3 max pooling and average pooling, and identity and zero operation (skip-connection). We retain the same search space as in DARTS [11]. The total number of possible DAGs (without graph isomorphism) containing N intermediate nodes with a set of operators $\mathcal{O}$ is

$$\prod_{k=1}^{N} \frac{(k+1)k}{2} \times (|\mathcal{O}|^2).$$

We encode each cell's structure as a configurable vector $\boldsymbol{\theta}$ of length $\sum_{i=2}^{N+1} i$ and simultaneously search for both normal and reduction cells. Therefore, the total number of viable cells will be raised to the power of 2. Thus, the cardinality of $\mathcal{F}$—set of all possible neural architectures—is $(|\mathcal{O}|^{\sum_{i=2}^{N+1} i})^2$. Also, we observed that the discrete search space for each type of cells (normal and reduction) is enormously expanded by a factor of $|\mathcal{O}|^{(N+1)}$ when increasing the number of intermediate nodes from N to $N+1$. Consequently, SOTA 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 of search space expansion 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.

2) Evaluation Criteria for NAS Algorithms: The evaluation metric for NAS algorithms considers three components, which are 1) versatility in different data scenarios, 2) computational expense for the search phase, and 3) the power of representation learning from discovered neural architecture. *First*, self-supervised NAS algorithms completely relieve data annotations for the search phase, enabling a reduced cost in terms of data curation. *Second*, a good NAS algorithm should require affordable computational resources, which results in a reasonable computational cost for the search phase. *Finally*, a good NAS algorithm should derive a neural architecture with high representation learning ability with reasonable model complexity in terms of number of parameters. The representations produced by searched architecture should have a high level of generalization. In other words, an intermediate-sized representations can capture complex concepts from an extensive number of inputs, while disentangle the variation factor and mitigate the variance in the data distribution. Although evaluation of representation learning ability remains an open question and mainly relies on the training task [47], we assume a simple but practical assumption that is a good NAS algorithm should result in a high performance neural architecture, in terms of accuracy in validation. In particular, the validation accuracy is given as

$$\text{Test error} = \left(1 - \frac{(\text{TP}+\text{TN})}{(\text{TP}+\text{TN}+\text{FP}+\text{FN})}\right) \times 100\%. \quad (1)$$

B. Contrastive Self-Supervised Learning

We employ a recent contrastive learning framework in PIRL [36], allowing multiple views on a sample. Let $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{i=|\mathcal{D}|}$ be a train set (for searching) and $\mathcal{F} = \{f_{\boldsymbol{\theta}}(.)\}_{\boldsymbol{\theta}\in\Theta}$ be a set of all neural architecture candidates (see Fig. 2).

1) Each sample $\boldsymbol{x}_i \in \mathbb{R}^{H\times W\times 3}$ is first taken as input of a stochastic data augmentation module, which results in a set of correlated views $\boldsymbol{x}_i^t = \{\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(M)}\}$. 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 $\mathcal{X} = \{\boldsymbol{x}_i\} \cup \boldsymbol{x}_i^t$ is called positive pair of sample $\boldsymbol{x}_i$.
2) Each candidate architecture $f_{\boldsymbol{\theta}}(.)$ is used as a base encoder to extract visual representations from both original sample $\boldsymbol{x}_i$ and its augmented views $\boldsymbol{x}_i^t$. We use the same multilayer perceptron $g(.)$ at the last of all neural candidates, projecting its feature maps of original $\boldsymbol{x}_i$ image under $f_{\boldsymbol{\theta}}(.)$ into a vector $\mathbb{R}^p$. For augmented views, another intermediate multi perceptron (MLP) $l(.) : \mathbb{R}^{Mp} \to \mathbb{R}^p$ is applied on concatenation of $\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i^{(m)})\}$ for $m \in M$. We denote the representing vectors of original image and augmented views as $\boldsymbol{z}_i$ and $\boldsymbol{z}_i^t$, respectively.

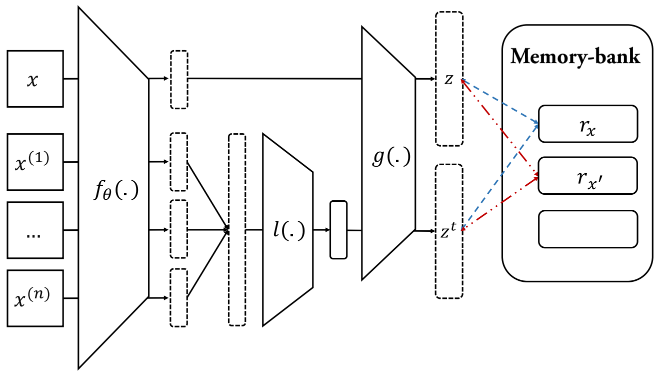

Fig. 2. PIRL genetic framework: Input image along with its augmented views are fed forward the same neural candidate parameterized by a collection of model's weight $\boldsymbol{\theta}$. The derived feature maps are then projected onto lower dimensional representations by $l(.)$ and $g(.)$. Contrastive loss maximizes the similarity between the image representation of original image x and its augmented views with its visual presentations r_x in the memory-bank (dashed green line), while the agreement with negative samples r'_x is minimized (dashed-dotted red line). The usage of memory-bank enables us to cache all representations of all samples in the input data, which is the exponential moving average of representations from prior epochs.

We also use the cosine similarity as the similarity measurement as in [35], [36], yielding $s(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}/||\boldsymbol{u}||.||\boldsymbol{v}||$. Each minibatch of K instances is randomly sampled from $\mathcal{D}$, giving $M\times K$ data points. Similar to [48], 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 sample is extracted visual representations as $\boldsymbol{z'}_i = g(f_{\boldsymbol{\theta}}(\boldsymbol{x'}_i))$. Following [36], we compute the noise contrastive estimator (NCE) of a positive pair $\boldsymbol{x}_i$ and $\boldsymbol{x}_i^t$ using their corresponding $\boldsymbol{z}_i$ and $\boldsymbol{z}_i^t$, given by

$$\ell(\boldsymbol{z}_i, \boldsymbol{z}_i^t) = \frac{\exp(\frac{s(\boldsymbol{z}_i,\boldsymbol{z}_i^t)}{\tau})}{\exp(\frac{s(\boldsymbol{z}_i,\boldsymbol{z}_i^t)}{\tau}) + \sum_{\boldsymbol{x'}_i\in\mathcal{X}'}\exp(\frac{s(\boldsymbol{z}_i^t,\boldsymbol{z'}_i)}{\tau})}. \quad (2)$$

The estimators are used to minimize the loss

$$\mathcal{L}_{\text{NCE}}(\boldsymbol{x}_i, \boldsymbol{x}_i^t) = -\log \ell(\boldsymbol{z}_i, \boldsymbol{z}_i^t) - \sum_{\boldsymbol{x'}\in\mathcal{X}'} \log\left[1 - \ell\left(\boldsymbol{z}_i^t, \boldsymbol{z'}_i\right)\right]. \quad (3)$$

The NCE loss maximizes the agreement between the visual representation of the original image $\boldsymbol{x}_i$ and its augmented views $\boldsymbol{x}_i^t$, together with minimizing the agreement between $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$. We use memory-bank approach in [36], [49] to cache the representations of all samples in $\mathcal{D}$. The representation $\boldsymbol{r_x}$ in memory-bank $\mathcal{M}$ is the exponential moving average of $\boldsymbol{z_i}$ from prior epochs. The final objective function for each neural candidate is a convex function of two losses as in (3)

$$\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^t) = \lambda\mathcal{L}_{\text{NCE}}(\boldsymbol{r_x}, \boldsymbol{z}^t) + (1-\lambda)\mathcal{L}_{\text{NCE}}(\boldsymbol{r_x}, \boldsymbol{z}). \quad (4)$$

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

Algorithm 1: Sequential Model-Based Algorithm [45].

```
Given $(\mathcal{L}_{\boldsymbol{\theta}^*}, M_0, I, S)$:
  1)  **Initialize** history $\mathcal{H} \leftarrow \emptyset$
  2)  **For** iteration $i = 1$ **to** $I$:
        • $\boldsymbol{\theta}^* \leftarrow \arg\min_{\boldsymbol{\theta}} S(\boldsymbol{\theta}, M_{i-1})$
        • Evaluate $\mathcal{L}_{\boldsymbol{\theta}^*}(\boldsymbol{x}, \boldsymbol{x}^t)$
        • Update $\mathcal{H} \leftarrow \mathcal{H} \cup (\boldsymbol{\theta}^*, \mathcal{L}_{\boldsymbol{\theta}^*}(\boldsymbol{x}, \boldsymbol{x}^t))$
        • Fit $M_i$ to $\mathcal{H}$
  3)  **Return** $\mathcal{H}$
```

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 solution. Nevertheless, the main difficulty when expanding the search space is 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 SMBO, which has been widely used when the fitness evaluation is expensive. This optimization algorithm can be a promising approach for cell-based NAS, since current SOTA 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 [3] is the first framework which applies SMBO for cell-based NAS. The surrogate model of PNAS predicts the performance of neural architectures without training them. In contrast to their approach, where the fitness function is in-validation accuracy, we model the contrastive loss in (4) by a surrogate function $S(.)$, which requires less computational expenses. Specifically, a large number of candidates will be drawn to evaluate the EI at each iteration. The surrogate function approximates the contrastive loss over the set of drawn points, resulting in cheaper computational cost. Mathematically, the optimization problem is formulated as

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

The SMBO algorithm is summarized in Algorithm 1, which attempts to optimize the EI criterion [45]. Given a threshold value t^*, EI is the expectation under an arbitrary model M, that $\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^t)$ will exceed t^*. Mathematically, we have

$$\mathrm{EI}_{t^*} := \int_{-\infty}^{\infty} \max(t^* - t, 0) p_M(t|\boldsymbol{\theta}) dt. \tag{5}$$

In constrast with Gaussian-process, the TPE estimator models $p(\boldsymbol{\theta}|t)$ and $p(t)$ instead of directly modeling $p(t|\boldsymbol{\theta})$. Then we decomposes $p(\boldsymbol{\theta}|t)$ to two density functions

$$p(\boldsymbol{\theta}|t) = \begin{cases} l(\boldsymbol{\theta}) & \text{if } t < t^* \\ g(\boldsymbol{\theta}) & \text{if } t \geq t^* \end{cases} \tag{6}$$

where $l(\boldsymbol{\theta})$ is the density function of candidate architectures corresponding to $\{\boldsymbol{\theta}^{(i)}\}$, such that $\mathcal{L}_{\boldsymbol{\theta}^{(i)}}(\boldsymbol{x}, \boldsymbol{x}^t) < t^*$ and $g(\boldsymbol{\theta})$ is formed by the remaining architectures. TPE leverages multiple observations in the search space of NAS under nonparametric densities, enabling a learning process that derives multiple densities over the search space simultaneously. Besides, other difference between TPE and Gaussian-based approach is the selection of t^*. The TPE algorithm favors t^* larger than the best observation of neural candidates, and then utilizes several points to construct the density $l(\boldsymbol{\theta})$, while Gaussian process favors more aggressive t^* less than the best observation in the history $\mathcal{H}$. Thus, TPE can choose t^* corresponding to some quantile of t, such that $p(t < t^*) = \gamma$. As a result, the EI in (5) is reformed as

$$\begin{aligned}\mathrm{EI}_{t^*}(\boldsymbol{\theta}) &= \int_{-\infty}^{t^*} (t^* - t) p(t|\boldsymbol{\theta}) dt \\ &= \int_{-\infty}^{t^*} (t^* - t) \frac{p(\boldsymbol{\theta}|t)p(t)}{p(\boldsymbol{\theta})} dt \propto \left(\gamma + \frac{g(\boldsymbol{\theta})}{l(\boldsymbol{\theta})}(1-\gamma)\right)^{-1}\end{aligned} \tag{7}$$

where γ denotes $p(t < t^*)$. The tree structure in TPE allows us to draw multiple candidates according to $l(.)$ and then evaluate them based on $g(\boldsymbol{\theta})/l(\boldsymbol{\theta})$. The TPE employed for the search strategy of NAS involves discrete-valued valuables, which represent the operations within neural cell's structure. The estimator samples a model for the search space by adaptively replacing the density in the vicinity of K observations $\{\boldsymbol{\theta}^{(i)}\}_1^K$. The TPE treats the prior distribution of discrete variables as a vector of probability p_i, which has the same length as neural architecture's genotype vector. As a result, the posterior vector is proportional to $Lp_i + C_i$, where L is the vector length of the neural genotype and C_i is the counts of occurrences of choice i in $\{\boldsymbol{\theta}^{(i)}\}_1^K$. Finally, the search time of each iteration of TPE can be scaled linearly in $|\mathcal{H}|$ and the genotype vector length with sorted query of observation in $\mathcal{H}$.

IV. Experimental Results on CIFAR-10 and ImageNet

Our experiments on each dataset include two phases, NAS (see Section IV-A) and architecture evaluation (see Section IV-B). It is worth mentioning that NAS algorithms have different strategy for selecting dataset for the search phase, while the same validation set is used in the evaluation phase. Pioneers such as NASNet and AmoebaNet assume the data constrain, which require the same dataset in search and evaluation. Although achieving remarkable results, the search cost of those approaches is extensively expensive due to the massive size of ImageNet ($\approx$ 14 million samples). Following NAS algorithms neglect the constraint, allowing search on smaller proxy dataset. The most popular proxy for ImageNet is attributed to CIFAR-10, which is widely used in later NAS algorithm. Regarding our work, we would like to preserve the data constrain since it is reasonable to assume that the neural solution found under the constrain may have higher performance. In the search phase, we used only 10% of unlabeled data (5000 samples from CIFAR-10 [9] and approximately 12 000 samples from ImageNet [8]) to search for neural architectures having the lowest contrastive loss mentioned in (4) by CSNAS. The best architecture is scaled to a larger architecture in the validation phase, and 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 $\mathcal{O}$ as in Section III-A1, which has been obtained by the most frequently chosen operators in [1], [3], [4], [11]. 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 $\text{CSNAS}_{N=4}$ and $\text{CSNAS}_{N=5}$, which are corresponding to the number of intermediate nodes N. With $N = 4$, a configurable vector $\boldsymbol{\theta}$ representing a neural architecture has the length of 28 ($\boldsymbol{\theta}_{\text{normal}} = \boldsymbol{\theta}_{\text{reduced}} = 14$), resulting in a search space of size $(8^{14})^2 \approx 2 \times 10^{25}$. 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 $\boldsymbol{\theta}$ of length 40, which tremendously surges the total number of possible architectures to 10^{36}. Experiments involving ImageNet only use the latter search space with $N = 5$.

We configure our CSSL by two augmented views ($M = 2$) for each sample with methods mentioned in Section III-B, producing $2 \times (K - 1)$ negative examples for each data instance in a minibatch of size K. The other two hyperparameters τ and λ in (2) and (3) are taken from the best experiment in [36], where $\tau = 0.07$ and λ $= 0.5$. Besides, MLPs $g(.)$ and $l(.)$ project encoded convolutional maps to a vector of size $p = 128$. Although we expect a minor impact of the above hyperparameters, we will leave this tuning problem for further study.

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

We summarize the parameter settings and discovered cell architecture for all experiments in Section A1.

B. Effectiveness Evaluation

We select the architecture having the best score from the searching phase and scale it for the validation phase. Within this article's scope, we only scale the searched architecture to the same size as baseline models in the literature (≈ 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.

Before analyzing the experimental results of CSNAS, we would like to outline several evaluation metrics for a NAS algorithm briefly. To begin with, we emphasize that predictive performance is a sufficient condition for good NAS algorithms. However, it is mandatory further to consider the versatility of NAS algorithms in different scenario. As mentioned in Section II-D, supervised NAS algorithms likely result in better neural architectures than SL NAS since they leverage full knowledge of training data (with annotations). On the other hand, self-supervised NAS offers us the opportunity to utilize additional unlabeled out-of-training samples, potentially lifting the curse of data when it comes to a scarcity scenario. Another evaluation metric for NAS algorithms is based on their ability to implement with limited computational resources. Finally, reported results from NAS algorithms are sensitive to the hyperparameters setting of *evaluation phase*, which may be attributed to the gain in the overall performance. For example, DART used the same hyperparameter setting (or reproduce other NAS with the same setting) with [1], [3], [4], [16]; thus, the gain in accuracy can be entirely attributed to the effectiveness of search strategies. On the other hand, P-DART and PC-DART used a slightly higher regularization (increment of 0.1 drop path probability) and larger batch size but remained the same learning rate as DART. The overall improvement may be slightly gained by such random effects in the evaluation phase. However, it cannot be deniable that P-DART and PC-DART offer us very highly efficient gradient-based NAS algorithms, tremendously reducing the computational expense in the search phase.

In the first block of Table I, we compare our search with neural architectures designed manually. Searched model by CSNAS gains approximately 2% in test accuracy when compared with ResNet-1001, while smaller gap (about 1%) on comparison to DenseNet-BC and VGG11B. We consider the results from manual designs a baseline for evaluating NAS algorithms since NAS's motivation is to search for better neural solutions automatically.

In the second block, we report the results from SOTA supervised NAS algorithms. We perform the evaluation phase for CSNAS based on the exact setting used in [3]–[5], [11], [16] to draw a fair comparison with these supervised NAS algorithms. Regarding CIFAR-10, we report the performance of architecture searched by CSNAS in Table I. It is highlighted that $\text{CSNAS}_{N=4}$ 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, $\text{CSNAS}_{N=5}$ can reach comparable results with AmoebaNet and NASNet in a tremendously less computational expense (1 vs. 3150 and 2000, respectively). Similarly, the results of CSNAS 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 images can be found in [35]). The performance of the network found by CSNAS appears to outperform DART and NASNET-A but to be slightly lower than AmoebaNet-C. Moreover, from both Tables I and II, the overall observation is that CSNAS possesses the ability to search for high-performance neural architectures, reaching comparable results with supervised NAS algorithms while using limited knowledge of data in search.

The third block reports the performance of self-supervised NAS algorithms, including SSNAS and UnNAS-DART. It is noted that the two algorithms and our CSNAS search

TABLE I
PERFORMANCE (IN TERMS OF TEST ERROR) OF SOTA NAS ALGORITHMS ON CIFAR-10

Neural Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	# Ops	Search Strategy
DenseNet-BC [42]	3.46	25.6	-	-	Manual
VGG11B (2×) [43]	3.60	42.0	-	-	Manual
ResNet-1001 [44]	4.62	10.2	-	-	Manual
AmoebaNet-A + cutout [4]	3.12	3.1	3150	19	Evolution
AmoebaNet-B + cutout [4]	2.55 ± 0.05	2.8	3150	19	Evolution
CARS-I [50]	2.62	3.6	0.4	7	Evolution
Hierarchical evolution [4]	3.75 ± 0.12	15.7	300	6	Evolution
LEMONADE [51]	3.05	4.7	80	-	Evolution
NSGANet [52]	3.85	3.3	8	7	Evolution
BlockQNN [53]	3.54	39.8	96	8	RL
ENAS [16] + cutout	2.89	4.6	0.5	6	RL
NASNet-A [1] + cutout	2.65	3.3	2000	13	RL
BayesNAS [54] + cutout	2.81 ± 0.04	3.4	0.2	-	Gradient-based
DARTS (1^{st} order) [11] + cutout	3.00 ± 0.14	3.3	1.5	7	Gradient-based
DARTS (2^{nd} order) [11] + cutout	2.76 ± 0.09	3.3	4	7	Gradient-based
GDAS [55] + cutout	2.93	3.4	0.21	7	Gradient-based
P-DARTS [40] + cutout	2.50	3.4	0.2	7	Gradient-based
PC-DARTS [12] + cutout	2.57 ± 0.07	3.6	0.1	7	Gradient-based
ProxylessNAS [56] +cutout	2.08	5.7	4.0	-	Gradient-based
MiLeNAS [57]	2.51 ± 0.11	3.87	0.3	-	Gradient-based
SNAS (moderate) [41] + cutout	2.85 ± 0.02	2.8	1.5	7	Gradient-based
GP-NAS [58]	3.79	3.90	0.9	7	Gaussian-Process-based
PNAS [3]	$3.41 + \pm 0.09$	3.2	225	8	SMBO
SSNAS [38]	2.61	-	0.21	-	Gradient-based
CSNAS$_{N=4}$ *(ours)* + cutout † ‡	2.71 ± 0.11	3.5	0.25	7	SMBO-TPE
CSNAS$_{N=5}$ *(ours)* + cutout † ‡	2.66 ± 0.07	3.4	1	7	SMBO-TPE

†Results based on ten independent runs. The search cost includes only searching time by SMBO-TPE algorithm, excluding the final architecture evaluation cost. 🟩 Experiments use the same hyperparameters setting reported in Appendix A2. ⬜ Experiments with unknown/different evaluation settings.

without using any data annotations. However, both SSNAS and UnNAS-DART used the whole training data (neglecting labels) for search, while we only used a small proportion of original training data. In the experiment on CIFAR-10, our CSNAS reaches slightly lower predictive performance than SSNAS. However, the hyperparameter setting for evaluation phase is not reported within SSNAS's work, so it is hard to compare the performance of resulting neural architectures. Regarding ImageNet, our CSNAS obtains a gain of 2% in accuracy compared to SSNAS under the same evaluation setting. Since the evaluation of neural architecture derived by NAS algorithms is extremely sensitive to the evaluation setting, we adopt the same evaluation setting with UnNAS, which is reported in Section A3, for a fair comparison. First, under the same setting, the neural architecture derived by CSNAS is slightly better than UnNAS-DARTS with solving jigsaw-puzzle SL, achieving 23.8 ± 0.14 top-1 accuracy in comparison to 24.1 ± 0.15 in UnNAS-DARTS. Second, it is worth noting that the search space of our CSNAS is much larger than self-supervised NAS competitors, which allows us to discover more potential neural candidates. Although searching on more massive search space, the time complexity of CSNAS is nearly the same as UnNAS due to the usage of surrogate models. In other words, the SMBO-TPE estimator enables us to approximate the expensive contrastive loss by cheaper surrogates. We report the hyperparameter setting for evaluation phase in Sections A2 and A3. In addition, we report the classification activation maps (CAMs) from ImageNet samples in Fig. A4.

C. Robustness of CSNAS to Loss of Information

In this section, we study the robustness of CSNAS to the loss of information, which is due to two reasons: 1) limited access to data's knowledge and 2) usage of surrogate models to approximate the true cost function. The first cause is that CSNAS leverages CSSL, conducted on unlabeled and limited samples. Therefore, it should be outperformed by any other supervised NAS algorithms. Thus, we do not attempt to compare CSNAS with other supervised NAS algorithms. Instead, we would like to evaluate the loss of information caused by CSSL on the overall performance of CSNAS. It is worth mentioning that such loss is tough to measure since it depends on the dataset of interest. In other words, the loss of information will be different when it comes to different datasets. The second cause is due to surrogate models, which possess a high semblance to the true expensive loss function. Although surrogates' usage helps reduce the computational expense for the search phrase, the approximation process may induce other losses of information caused by surrogates. Since universally evaluating such loss is impossible, we only evaluate the loss of information from CSNAS on the CIFAR-10 dataset.

In the experimental design, we include two components: 1) CSSL and 2) usage of surrogate models, which both induce

TABLE II
PERFORMANCE (IN TERMS OF TEST ERROR) OF SOTA NAS ALGORITHMS ON IMAGENET. 🟩

Neural Architecture	Test Err. (%) Top-1 (Top-5)	Params (M)	$\times+$ (M)	Search Cost (GPU days)	Search Strategy
Inception-v1 [59]	30.2(10.1)	6.6	1448	-	Manual
Inception-v2 [60]	25.2(92.2)	11.2	—	-	Manual
MobileNet [61]	29.4(10.5)	4.2	569	-	Manual
ShuffleNet (2×)-v1 [62]	26.4(10.2)	≈ 5	524	-	Manual
ShuffleNet (2×)-v2 [63]	25.1 (−)	≈ 5	591	-	Manual
NASNet-A [1] ‡	26.0(8.4)	5.3	564	2000	RL
NASNet-B [1] ‡	27.2(8.7) ‡	5.3	488	2000	RL
NASNet-C [1] ‡	27.5(9.0) ‡	4.9	558	2000	RL
AmoebaNet-A [4] ‡	25.5(8.0)	5.1	555	3150	Evolution
AmoebaNet-B [4] ‡	26.0(8.5)	5.3	555	3150	Evolution
AmoebaNet-C [4] ‡	24.3(7.6)	6.4	570	3150	Evolution
AutoSlim [64]	24.6(−)	8.3	532	224	Greedy
AtomNAS-A [65]	25.4(7.9)	3.9	258	-	Dynamic network shrinkage
AutoNL-S [66]	22.3(6.3)	5.6	353	32	Single-path
Single-Path [67]	25.0(7.79)	-	-	0.14	Single-path
MnasNet-92 [68]	25.2(8.0)	4.4	388	-	RL
PNAS [3] ‡	25.8(8.1)	5.1	588	255	SMBO
PARSEC [69]	26.0(8.4)	5.6	548	1	SMBO
FBNet-C [70]	25.1(−)	5.5	375	20	SMBO
P-DART [40]	24.1(7.3)	5.4	5.97	2.0	Gradient-based
PC-DART [12]	24.2(7.3)	5.3	597	3.8	Gradient-based
ProxylessNAS [56]	24.9(7.5)	7.1	465	8.3	Gradient-based
MiLeNAS [57]	24.7(7.6)	5.3	584	0.3	Gradient-based
DARTS (2nd order) [11] ‡	26.7(8.7)	4.7	574	4	Gradient-based
SNAS (mild constraint) [41]	27.3(8.7)	4.3	522	1.5	Gradient-based
RCNet [71]	27.8(9.0)	3.4	294	8	Gradient-based
GDAS [55]	26.0(8.5)	5.3	581	0.8	Gradient-based
SSNAS [38]	27.75(9.55)	-	-	-	Gradient-based
UnNAS-DARTS [39] + Jigsaw + cutout §	24.1 ± 0.15(−)	5.2	567	2	Gradient-based
CSNAS$_{N=5}$ *(ours)* + cutout †	25.8 ± 0.17(8.3)	5.1	590	2.5	SMBO-TPE
CSNAS$_{N=5}$ *(ours)* + cutout §	23.9 ± 0.14(8.1)	5.1	590	2.5	SMBO-TPE

†Results based on ten independent runs.
§Results based on three independent runs. Experiments use the same hyperparameters setting reported in Appendix A2. 🟥 Experiments use the same hyperparameters setting reported in Appendix B. ⬜ Experiments with unknown/different evaluation settings.

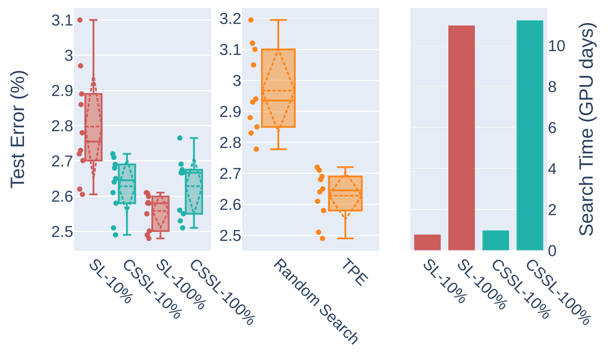

Fig. 3. Experimental results on the robustness of CSNAS to the loss of information due to usage of contrastive self-supervised learning and surrogates. SL and CSSL are supervised and contrastive self-supervised learning. $-a\%$ represents the percentage of data used for the search phase.

the loss of information. First, we evaluate such loss by fixing 2) and changing 1) to supervised search. As a result, we can approximate the loss of information caused by CSSL on CIFAR-10. The left panel of Fig. 3 shows that the loss of information caused by CSSL does not exist from experiments on 10% of data samples without any annotation (CSSL-10% vs. SL-10%) since the neural architecture searched by CSSL outperforms that derived by SL. On the other hand, the loss of information appears when it comes to 100% of the dataset (CSSL-100% vs. SL-100%). As expected, the model found by supervised learning achieves a better result than that from CSSL. However, we observe that such loss is minimal in terms of validation accuracy. Moreover, it appears that the loss of information caused by the proportion of samples used is extremely small (CSSL-10% vs. CSSL-100%) since the performance of derived architectures is nearly the same. The last detail of interest in this experiment is investigating the time complexity of the search phase due to CSSL. As mentioned before, CSSL requires multiple augmented views of a given input. From the right panel of Fig. 3, we can see that the computational expense increased by CSSL is extremely minimal. Hence, the additional search cost is inconsiderable.

Regarding the second component, we compare the result from SL-100% with other supervised NAS addressed in Table I. It is noting that SL-100% is conducted by SMBO-TPE under supervised learning with full access to CIFAR-10, achieving 2.56% in test error. Referring to Table I, SMBO-TPE mostly outperforms other search strategies (except MiLeNAS) under the same evaluation setting, which emphasizes the effectiveness of SMBO-TPE in comparison with other optimization strategies.

We also observe the effectiveness of the TPE in comparison to baseline random search, which is highlighted in the middle panel of Fig. 3. Moreover, SMBO-TPE constructs a probability model of the cost function in (3) and then uses it to find the most potential neural candidates to evaluate the true function. Before evaluating the true cost function, we sample $20K$ genotypes that defined neural candidates from the search space and only selected the top 20% of this population. The optimal neural architecture is founded by approximately 500 rounds of evaluating the true cost function. By periodically computing the true cost, the loss of information caused by surrogates is mitigated by providing a better approximation of the loss landscape.

In conclusion of this section, we observe no loss of information caused by CSSL on 10% of the dataset. However, such loss appears on 100% of samples, where supervised search outperforms C-SSL. Besides, the loss of information caused by using surrogates is nearly the same as other competitors. Finally, SMBO-TPE aggressively samples the most promising neural candidates for each exact evaluation of the true cost function, which offers well-approximation for the lost landscape. As a result, the loss of information caused by surrogates is alleviated.

V. Case Study

This section investigates the effectiveness of CSNAS on a practical case study, which involves skin lesion classification. The underinvestigated problem is an excellent example of a data-scarcity scenario, where labeled data are costly (requires expert knowledge), and unlabeled data have zero-cost for annotations. Self-supervised NAS algorithms are suitable for such cases. Moreover, at the same time as our study, CSNAS is the second work considering skin lesion classification. Kwasigroch et al. [72] used network morphism to search neural architecture for skin lesion classification. However, they only consider binary classification problems while we performed multiclass classification. We are hoping to compare our CSNAS to other self-supervised NAS. Unfortunately, we cannot find the official implementation of SSNAS and UnNAS in the meantime. It is noted that reproducing the work without official implementation may induce inaccurate observation and false comparison [73], [74]. Therefore, we can only compare our CSNAS to the conventional approach for skin lesion classification, which is transfer learning.

We organize the section as follows. Section V-A summarizes the classification problem with a detailed introduction of the ISIC-2019 database. Section V-B gives a detailed experimental setting for the case study. Finally, we report the quantitative results in Section V-C. To avoid confusion with searched architectures from CIFAR-10 and ImageNet, we named architecture searched on ISIC-2019 as DermoCSNAS.

A. ISIC-2019 Dataset

The dataset of interest is International Skin Imaging Collaboration database (ISIC 2019) [75]–[77], which includes a public train set of 25 331 labeled images and a private test set of 8238 unlabeled images. For the illustration of the proposed neural solution, the unlabeled samples in the test set are used for searching deep neural architecture in a self-supervised manner, and then the found architecture is conventionally trained on the train set to perform classification. It is highlighted that the public train set and private test set provided by ISIC 2019 are not overlapped. The ISIC 2019 public train set originally contains nine classes, which are melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC), squamous cell carcinoma (SCC) and unknown disease (UNK). Since the number of unknown training samples is zero, we only consider defined disease, resulting in a multi-label classification of eight classes. Table III depicts the distribution of classes and search-train-test-validation splits (follows ratio $80\% - 5\% - 15\%$) for our experiment. Moreover, we also publish the list of train/test/validation samples in the GitHub repository for reproduction purposes.

TABLE III
ISIC-2019 Dataset in Detail

Dataset	Used Phase	Number of samples	Split
ISIC-2019 private test set	Search	$8,238$	-
ISIC-2019 public train set	Validation	Train	20,265
		Validation	1,290
		Test	3,776
		Total	25,331
Skin Disease	**Annotation**		**Distribution**
Melanoma	MEL	4,522	17.85%
Melanocytic Nevus	NV	12,875	50.83%
Basal Cell Carcinoma	BCC	3,323	13.12%
Actinic Keratosis	AK	867	3.42%
Benign Keratosis	BKL	2,624	10.36%
Dermatofibroma	DF	239	0.94%
Vascular Lesion	VASC	253	1%
Squamous Cell Carcinoma	SCC	628	2.48%
Unknown	UNK	0	0%

The data used for search are out-of-training and accounted for 32.5% of training data.

B. Experimental Setup

Our experiment includes two phases, which are (1) searching neural architecture on unlabeled data (ISIC private test set) under a SL manner and (2) evaluating discovered neural net on labeled samples (ISIC public train set). To preserve our procedure's robustness, we wholly removed the learned model weights after the searching phase. Then, we found that neural architecture was trained from scratch using a random initialization in the validation phase.

1) Search Phase: Our configuration for this experiment is similar to CIFAR-10 and ImageNet. First, we investigate two configurations for searching spaces, corresponding to $N = 4$ and $N = 5$ intermediate nodes. The architectures found under these settings are denoted as DermoCSNAS-4 and DermoCSNAS-5, in which the encoded vector $\boldsymbol{\theta}$ of a neural candidate DermoCSNAS-4 is a 28-dimensional vector, while the corresponding vector for DermoCSNAS-5 is a 40-dimensional vector. As a result, the number of possible neural candidates

TABLE IV
COMPARISONS BETWEEN SOTA NEURAL ARCHITECTURES AND OUR PROPOSED MODEL, IN TERMS OF TEST ACCURACY AND MODEL COMPLEXITY

Neural Architecture	Params (%) (M)	FLOPS (G)	Test Acc. (%)	Melanoma F1-score
DPN-131 [79]	79.3	16.0	86.23	0.79
EfficientNet-B0 [80]	5.3	0.39	82.14	0.80
ResNet152 [81]	60.0	11.3	84.00	0.75
ResNet101 [81]	44.6	8.0	87.76	0.81
Inception-v4 [59]	46.0	12.3	85.99	0.79
Inception-ResNet-v2 [82]	55.8	11.75	87.53	0.80
NASNet [1]	88.9	24.0	87.80	0.82
PNASNet [3]	86.1	25.2	87.87	0.81
SENet101 [83]	49.2	8.0	87.55	0.81
SENet154 [83]	145.8	42.3	88.00	0.83
Xception [84]	23.0	8.4	87.18	0.80
DermoCSNAS$_{N=4}$	3.55	0.503	87.94 ± 0.04	0.83
DermoCSNAS$_{N=5}$	**4.15**	**0.560**	**88.68** ± 0.02	**0.84**

The results of our DermoCSNAS are mean and variance from 10 independent runs, which use the same set of training hyperparameters in Section V-B2. Moreover, we use the same experimental setting (data augmentation and training parameters) across all transfer learning experiments in order to provide unbiased results.

TABLE V
CLASSIFICATION REPORT OF OUR DERMOCSNAS$_{N=5}$

Classes	Precision	Recall	F-1 score	Support
Melanoma	0.84	0.81	0.83	647
Melanocytic Nevus	0.92	0.95	0.93	1991
Basal cell carcinoma	0.87	0.90	0.88	465
Actinic Keratosis	0.79	0.62	0.69	119
Benign Keratosis	0.76	0.76	0.76	381
Dermatofibroma	0.85	0.78	0.81	36
Vascular	0.97	0.72	0.82	39
Squamous cell carcinoma	0.75	0.67	0.71	98
Macro average	0.84	0.78	0.81	3776
Weighted average	0.88	0.88	0.88	

Champion device data are shown in the table. average values along with the standard deviations are shown in brackets.

exponentially increases (5×10^{10} times) when added to only one intermediate node. In Section III-A, we generate neural candidates using the operation-generating set $\mathcal{O}$ mentioned in section and perform CSSL with two augmented views ($M = 2$) for each sample, resulting in $2 \times (K - 1) = 298$ negative examples for each data point in a mini-batch of $K = 150$ samples. Each candidate contains only 8 layers with 16 initial channels, which is trained using momentum SGD with the learning rate of 0.001 and momentum of 0.9. The hyperparameter for NSE in equation is set as $[\tau, \lambda] = [0.07, 0.5]$. Finally, we initialize the TPE by 20 random samplings, followed by $20K$ suggested points for computing the EI of each trial. Therefore, only 20% of best candidates having the largest EI are mutated for estimating the next $\boldsymbol{\theta}$. The found neural architecture is illustrated in Figs. A1, A2, and A3.

2) Validation Phase: First, we construct the final neural architecture by expanding the depth (number of layers) and the width (number of initial channels) from the discovered neural cell in the search phase. In both configurations of $N = 4$ and $N = 5$, we stack 18 layers of the founded cell with 48 initial channels since we aim to provide a hardware-aware deep neural architecture, which is restricted under 600 number of multiply–add operation. Note that all of the reduction cells are in one-half and two-thirds of the depth of model, in which all chosen operations use a stride of 2. Second, we augmented training samples by downsampling to 256×256 before random-cropped into 224×224, and then randomly applying horizontal and vertical flipping. Moreover, we prevent overfitting by linearly increasing path dropout of 0.2 as in [1], [3]–[5], [11], cutout of length 16 [78], and a small auxiliary classifier (at two-thirds of model's depth) with weight of 0.4. Finally, the model is trained with a batch size of 128 using SGD optimizer, which is initialized by 0.02 learning rate and 0.9 weights decay. The learning rate has a decay rate of 0.99 for every 3.5 epoch.

C. Effectiveness Analysis

We depict the effectiveness of our approach by comparing it with SOTA models in Table IV . It is noted that other SOTA architectures are fine-tuned with pretrained weights (transfer learning), while our neural architecture is trained from scratch on the ISIC dataset. Hence, it is inevitable that lower level features in early layers of our model are learned from skin lesion images. In contrast, we need to accept inherited lower level features from domain datasets (ImageNet or CIFAR-10) once performing transfer learning and fine-tuning pretrained models. As discussed in the beginning of this section, we cannot reproduce other self-supervised NAS algorithms without significant inaccurate results. Thus, we instead evaluate the performance of DermoCSNAS by comparing to the most conventional approach of skin lesions classification, which is transfer learning. Within the scope of this article, we compare our discovered neural architecture with SOTA deep convolution networks, which include Efficient-B0 [80], ResNes-101 and ResNet-152 [81], Inception-v4 [59], Inception-ResNet-v2 [82], DPN-131 [79], Xception [84], SENet101 and SENet 154 [83], NASNet [1], and PNASNet [3].

First, the searched neural architecture under the best setting (DermoCSNAS$_{N=5}$) outperforms other human-crafted model, gaining 0.68%–4.68% (compared with SENet154 and ResNet152, respectively). We also observed no trade-off between the model complexity and the overall performance since our model possesses the smallest number of parameters and the number of multiply–add operations. Second, our model has the best F-1 score from malignant classes (melanoma) at 0.84, while others are ranging from 0.75 to 0.83. It is crucial since our neural intelligence makes no tradeoff between the overall accuracy and the balance of precision and recall from cancerous class. The detailed classification report is shown in Table V and the visualization of CAMs is shown in Fig. A5.

VI. DISCUSSION

A. Implication

Beyond the experimental results on mainstream datasets (CIFAR-10 and ImageNet) and the case study of skin lesion classification (ISIC-2019), we would like to discuss the general principles that can be taken from our CSNAS. The advantages of our CSNAS are mainly from the power of representation learning of CSSL, which can effectively learn the underlying representation with access to a small proportion of training data without labels. The advantage benefits the generalization of large-scale implementation. In a data-abundant scenario, we can randomly draw a small proportion of training data and ignore annotations for the search phase. Our study on ImageNet used a benchmarking sub-1% and 10% dataset (commonly used in SL and self-training results), while in ISIC-2019, we used additional unlabeled data for the search phase. It is worth mentioning that we often cope with computer vision problems involving data scenarios similar to our case study, in which data annotations are costly due to extensive human experts. Hence, CSNAS offers an efficient NAS algorithm in such scenarios.

B. Threats to Validity

Threats to internal validity include the consistency of reproducibility of work. The notorious challenge of NAS-related research is the reproducible ability [73], [74]. Early NAS algorithms require extensive computational resources to search for a neural architecture that is entirely unavailable for large-scale applications. Subsequence algorithms tackle the issue by search on a smaller configuration space and implement a more efficient search strategy. However, the results are not comparable to each other since the experimental setting is extremely sensitive to the final results. A partial solution for the issue can be delivered through the quantitative evaluation among pretrained models. However, we still hope to know a clear insight into the comparison of NAS algorithms. Several attempts in NAS-related research, including [85]–[88], tackle the reproducible issue efficiently. However, their implementation only narrows in the mainstream dataset—CIFAR-10.

Threats to external validity involve the generalization of our work on different domain-specific datasets and different computer vision tasks. First, our proposed CSNAS fundamentally is based on contrastive self-supervise learning, which possesses a strong image representation capability even when using a small proportion of training samples without annotations. Thus, CSNAS benefits NAS on data-scarcity scenario, where labeled data are costly. Our case study shows that DermoCSNAS achieves high predictive performance compared to the dominant competitor field—transfer learning. Moreover, we keep the data constrain for the search phase and evaluation phase, in which found neural architecture is discovered in the same dataset as the evaluation phase. Intuitively, the constrain offers us a robust neural architecture on the domain-specific dataset. Regarding the transferability of searched neural architectures to different computer vision tasks, the improvement is consistent when we investigate semantic segmentation [39], object detection [89], and adversarial learning [90].

VII. Conclusion

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

There are many directions to conduct further study on CSNAS. 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 CSNAS improvement is investigating further baseline SL methods, which potentially ameliorates current CSNAS benchmarks.

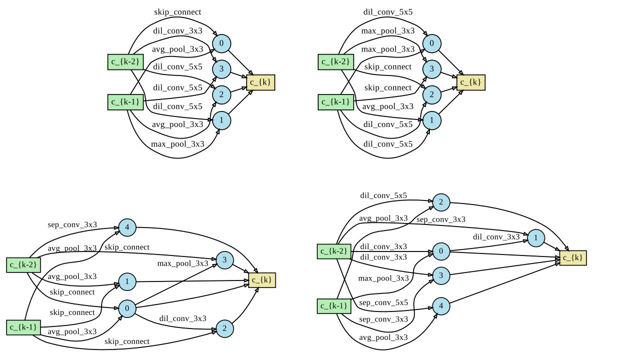

Fig. A1. Cell architecture of CSNAS$_{N=4}$ and CSNAS$_{N=5}$ searched on CIFAR-10. Each candidate cell's architecture is encoded to a vector $\boldsymbol{\theta}$ of length 28 ($\boldsymbol{\theta}_{\text{normal}} = \boldsymbol{\theta}_{\text{reduced}} = 14$) when $N = 4$. The length-encoded vector increases to 40 when we searched on the configuration space according to $N = 5$. Moreover, two inputs for every intermediate nodes are the feature maps produced by corresponding operations (edges). Produced feature maps will be concatenated at the end of a cell to creating input for subsequence cells.

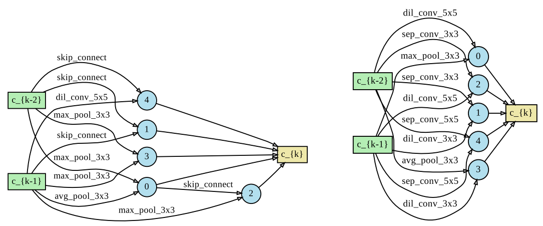

Fig. A2. Cell architecture of CSNAS$_{N=5}$ searched on ImageNet-1%. The vector $\boldsymbol{\theta}$ representing each neural candidate is of length 20.

Appendix A
Experimental Details

A. Neural Architecture Search

1) Experimental Setting: For CIFAR-10 dataset, we use 5000 class-balanced images to search for the best architecture. However, architecture search on ImageNet uses the same list of samples as Ref. [35].

We use a mini-batch of size 150, resulting in 298 negative samples corresponding to a single data point each mini-batch. We accelerate the searching time by using small architecture candidates, which include 8 layers and 32 channels. For optimizing the weights w in memory-bank $\mathcal{M}$, we use momentum SGD with learning rate $\eta_w = 0.001$ and momentum 0.9. We set $\tau = 0.07$ and $\lambda = 0.5$ for the NSE estimator and contrastive loss as in [36]. We set up the TPE sampler as in Section IV-A with zero initialization for $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{normal}}, \boldsymbol{\theta}_{\text{reduce}})$.

2) Neural Cell Discovered by CSNAS: We report neural cell discovered by CSNAS in Figs. A1– A3.

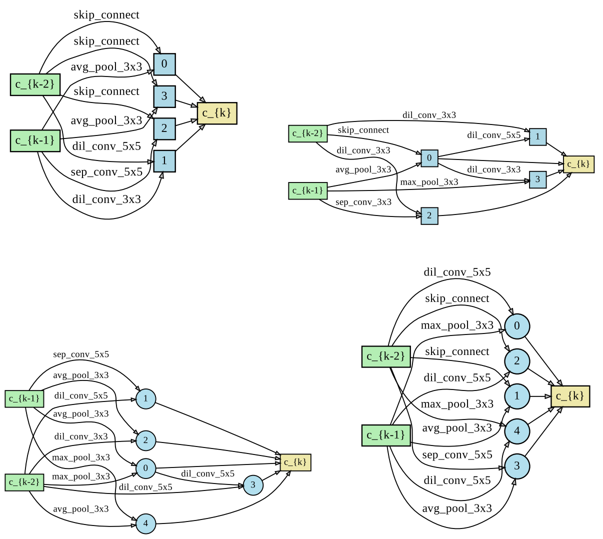

Fig. A3. Cell structures found in our proposed approach. Top panel shows discovered normal (left) and reduced (right) cell architectures from search space associated to DermoCSNAS$_{N=4}$, while bottom panels are associated to DermoCSNAS$_{N=5}$.

B) Neural Architecture Validation

We follow the setup used in [1], [3], [4], [11], where the first and second nodes of cell C_k are set to be compatible to the outputs of cell C_{k-2} and C_{k-1}, respectively. All reduction cells are located in half and two-third of the depth, which has stride equals two for all operations linked to the input node.

We construct a large network for the CIFAR-10 dataset, including 20 layers with 36 initial channels. The train setting is employed from existing studies [1], [3]–[5], [11], offering more regularization on training, which includes cutout [78] of length 16, linearly path dropout with probability 0.2, and auxiliary classifier (located in two-third maximum depth of the network) with weight 0.4. We train the network for 600 epochs using batch size 128. The chosen optimizer is momentum SGD with learning rate $\eta = 0.025$, momentum $= 0.9$, weights decay 3×10^{-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 224×224, and the allowed number of multiply–add operations is less than 600 number of operations, which is restricted for mobile settings. We train a network having 14 cells and 48 initial channels for 250 epochs with a batch size of 128. The learning rate is set at $\eta = 0.1$ with a decay rate of 0.97 and a decay period of 2.5. We use the same auxiliary module as the evaluation phase of CIFAR-10. We set the SGD optimizer at a momentum of 0.9, and weights decay 3×10^{-5}.

C) Hyperparameters Setting for Comparison Between UnNAS and CSNAS

This section reports the evaluation setting for comparison between UnNAS and CSNAS. The neural architecture includes 14 layers. We train the network for 250 epochs with initial learning rate of 0.5 under SGD optimizer. The number of warm-up epoch is 5 and cosine learning rate schedule is used. Moreover, we use batch size of 512 distributed over 4 GPUs. The weight for auxiliary loss is 0.4 and the drop-path-probability is 0.3.

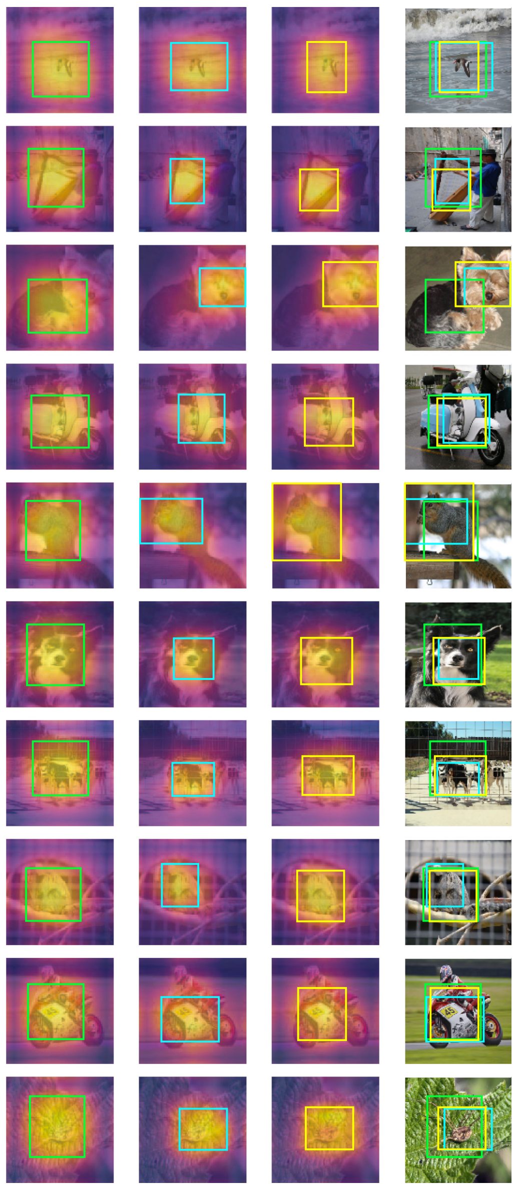

Fig. A4. Visualization of CAM. In the first three columns, we represent the classification activation maps (CAM) of ten random samples from DARTS, PDARTS, and CSNAS. The last column depicts the thresholded ROIs from CAMs with threshold of $\tau = 0.7$.

D. Visualization of CAM

See Figs. A4 and A5.

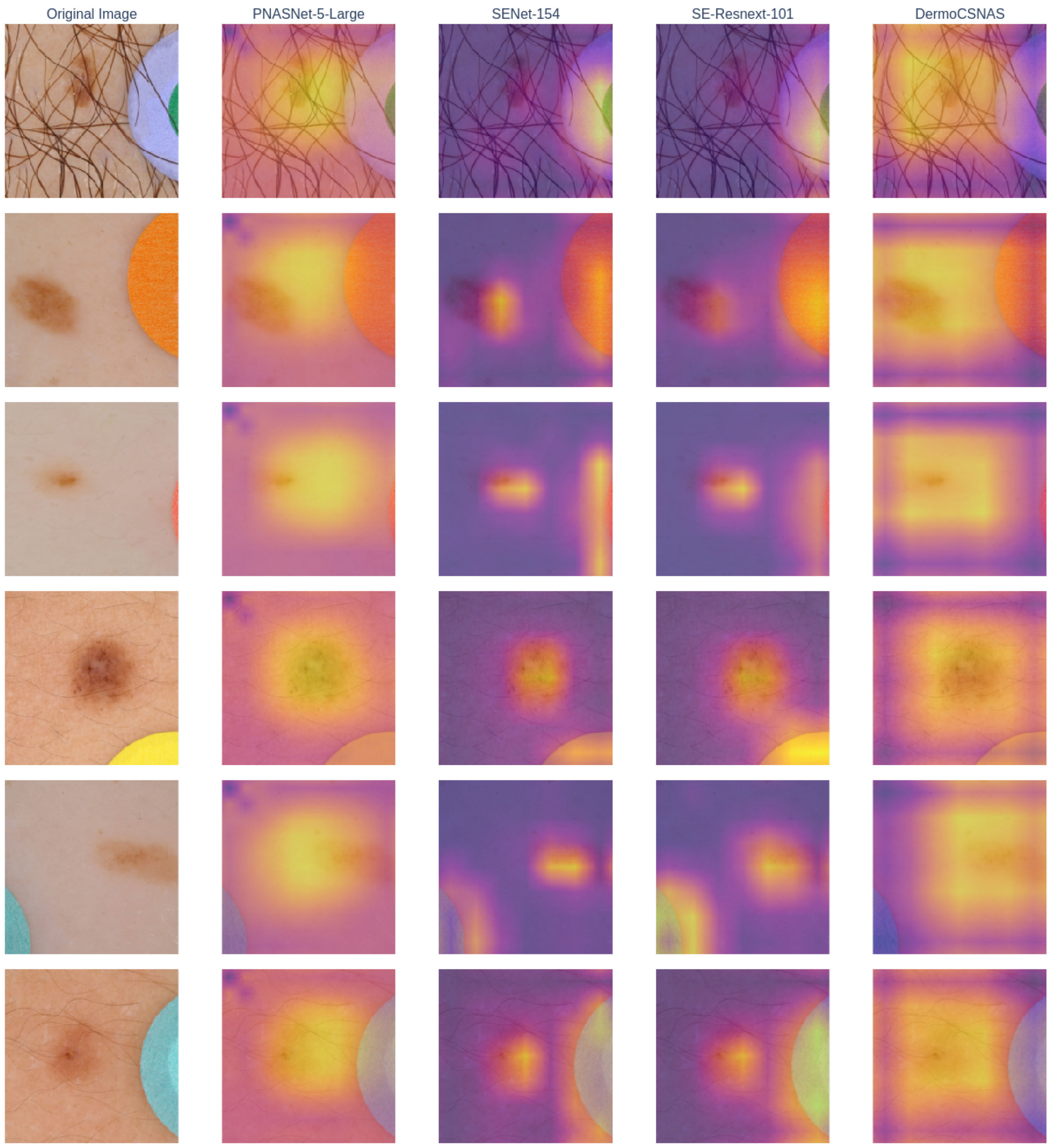

Fig. A5. Visualization of CAM from noisy test samples of ISIC-2019, which involve badges and hairs.

Acknowledgment

The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon. 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.

References

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[2] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.

[3] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[4] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 4780–4789.

[5] B. Zoph and Quoc V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[6] R. Negrinho and G. Gordon, "Towards modular and programmable architecture search," in *Proc. Conf. NIPS*, 2019, pp. 13733–13743.

[7] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with Bayesian optimisation and optimal transport," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2016–2025.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[9] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[10] H. Tan *et al.*, "RelativeNAS: Relative neural architecture search via slow-fast learning," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3096658.

[11] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[12] Y. Xu *et al.*, "PC-DARTS: Partial channel connections for memory-efficient differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[13] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive DARTS: Bridging the optimization gap for NAS in the wild," *Int. J. Comput. Vis.*, vol. 129, no. 3, pp 638–655, 2021.

[14] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[15] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 550–559.

[16] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*.

[17] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1.

[18] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," 2017, *arXiv:1705.10823*.

[19] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, " Smash: One-shot model architecture search through hypernetworks," 2017, *arXiv:1708.05344*.

[20] T. Mikolov, K. Chen, G. Corrado, J. Dean, L. Sutskever, and G. Zweig, "WORD2VEC," 22, 2013. [Online]. Available: https://code.google.com/p/word2vec

[21] A. Joulin *et al.*, "Zip: Compressing text classification models," 2016, *arXiv:1612.03651*.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.

[23] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2536–2544.

[24] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1058–1067.

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2016.

[26] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[27] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1734–1747, Sep. 2015.

[28] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. Int. Conf. Learn. Representations*, 2018.

[29] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1422–1430.

[30] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 649–666.

[31] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[32] J. Olivier *et al.*, "Data-efficient image recognition with contrastive predictive coding," 2019, *arXiv:1905.09272*.

[33] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5639–5650.

[34] J.-B. Grill *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[36] I. Misra and L. van der Maaten, "Self-supervised learning of pretext-invariant representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6707–6717.

[37] C. Wei, K. Shen, Y. Chen, and T. Ma, "Theoretical analysis of self-training with deep networks on unlabeled data," in *Proc. Int. Conf. Learn. Representations*, 2021.

[38] S. Kaplan and R. Giryes, "Self-supervised neural architecture search," 2020, *arXiv:2007.01500*.

[39] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 798–813.

[40] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.

[41] S. Xie, H. Zheng, C. Liu, and L. Lin, " SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.

[42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[43] A. Nøkland and L. H. Eidnes, "Training neural networks with local error signals," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[45] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[47] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[48] T. Chen, Y. Sun, Y. Shi, and L. Hong, "On sampling strategies for neural network-based collaborative filtering," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 767–776.

[49] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9729–9738.

[50] Z. Yang *et al.*, "Cars: Continuous evolution for efficient neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1829–1838.

[51] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *Proc. Int. Conf. Learn. Representations*, 2019.

[52] Z. Lu *et al.*, " NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 419–427.

[53] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.

[54] H. Zhou, M. Yang, J. Wang, and W. Pan, "BayesNAS: A Bayesian approach for neural architecture search," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, vol. 97, pp. 7603–7613.

[55] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.

[56] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Representations*, 2019.

[57] C. He, H. Ye, L. Shen, and T. Zhang, "Milenas: Efficient neural architecture search via mixed-level reformulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11993–12002.

[58] Z. Li, T. Xi, J. Deng, G. Zhang, S. Wen, and R. He, "GP-NAS: Gaussian process based neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11933–11942.

[59] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jul. 2015, vol. 37, pp. 448–456.

[61] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[62] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[63] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[64] J. Yu and T. Huang, "Autoslim: Towards one-shot architecture search for channel numbers," 2019, *arXiv:1903.11728*.

[65] J. Mei *et al.*, "Atomnas: Fine-grained end-to-end neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[66] Y. Li *et al.*, "Neural architecture search for lightweight non-local networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10297–10306.

[67] D. Stamoulis *et al.*, "Single-path NAS: Designing hardware-efficient convnets in less than 4 hours," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2019, pp. 481–497.

[68] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[69] F. P. Casale, J. Gordon, and N. Fusi, "Probabilistic neural architecture search," 2019, *arXiv:1902.05116*.

[70] B. Wu *et al.*, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10734–10742.

[71] Y. Xiong, R. Mehta, and V. Singh, "Resource constrained neural network architecture search: Will a submodularity assumption help?," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1901–1910.

[72] A. Kwasigroch, M. Grochowski, and A. Mikołajczyk, "Neural architecture search for skin lesion classification," *IEEE Access*, vol. 8, pp 9061–9071, 2020.

[73] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.

[74] C. Sciuto, K. Yu, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[75] P P. Tschandl, C. Rosendahl, and H. Kittler, "The Ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, Aug. 2018, Art. no. 180161.

[76] N. Codella *et al.*, "Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC)," in *Proc. IEEE 15th Int. Symp. Biomed. Imag.*, 2018, pp. 168–172.

[77] M. Combalia *et al.*, "BCN20000: Dermoscopic lesions in the wild," 2019, *arXiv:1908.02288*.

[78] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[79] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017.

[80] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[82] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-V4, inception-resNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017.

[83] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[84] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[85] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.

[86] X. Dong and Y. Yang, "NAS-bench-201: Extending the scope of reproducible neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[87] A. Zela, J. Siems, and F. Hutter, "NAS-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search," 2020, *arXiv:2001.10422*.

[88] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter, "NAS-bench-301 and the case for surrogate benchmarks for neural architecture search," in *Proc. 4th Workshop Meta-Learn. Neural Inf. Process. Syst.*, 2020.

[89] Y. Chen *et al.*, "Neural architecture search on object detection," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019.

[90] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3224–3234.