Fast and Unsupervised Neural Architecture Evolution for Visual Representation Learning

Song Xue, and Hanlin Chen
Beihang University, CHINA

Chunyu Xie
Qihoo 360 AI Research, CHINA; Beihang University, CHINA

Baochang Zhang
Beihang University, CHINA

Xuan Gong, and David Doermann
University at Buffalo, USA

Abstract—Unsupervised visual representation learning is one of the hottest topics in computer vision, yet performance still lags behind compared with the best supervised learning methods. At the same time, neural architecture search (NAS) has produced state-of-the-art results on various visual tasks. It is a natural idea to explore NAS as a way to improve unsupervised representation learning, yet it remains largely unexplored. In this paper, we propose a Fast and Unsupervised Neural Architecture Evolution (FaUNAE) method to evolve an existing architecture, manually constructed or the result of NAS on a small dataset, to a new architecture that can operate on a larger dataset. This partial optimization can utilize prior knowledge to reduce search cost and improve search efficiency. The evolution is self-supervised where the contrast loss is used as the evaluation metric in

Digital Object Identifier 10.1109/MCI.2021.3084394
Date of current version: 15 July 2021

Corresponding author: Baochang Zhang (e-mail: bczhang@buaa.edu.cn). Co-first authors: Song Xue, and Hanlin Chen

1556-603X/21©2021IEEE

a student-teacher framework. By eliminating the inferior or least promising operations, the evolutionary process is greatly accelerated. Experimental results show that we achieve state-of-the-art performance for downstream applications, such as object recognition, object detection, and instance segmentation.

I. Introduction

Learning high-level representations from labeled data and deep learning models in an end-to-end manner is one of the biggest successes in computer vision in recent history. These techniques make manually specified features largely redundant and have greatly improved the state-of-the-art for many real-world applications, such as medical imaging, security, and autonomous driving. However, there are still many challenges. For example, a learned representation from supervised learning for image classification may lack information such as texture, which matters little for classification but can be more relevant for later tasks. Yet adding it makes the representation less general, and it might be irrelevant for tasks such as image captioning. Thus, improving representation learning requires features to be focused on solving a specific task. Unsupervised learning is an important stepping stone towards robust and generic representation learning [1]. The main challenge is the significant performance gap compared to supervised learning. Neural architecture search (NAS) has shown extraordinary potential in deep learning due to the customization of the network for target data and tasks. Intuitively, using the target data and searching the tasks directly without a proxy gap will result in the least domain bias. For performance reasons however, early NAS [2] methods only searched small datasets. Later, transfer methods were utilized to search for an optimal architecture on one dataset, then adapt the architecture to work on larger datasets [3]. ProxylessNAS [4] was proposed to search larger datasets after directly learning the architecture for the target task and hardware, instead of using a proxy. Recent advances in NAS show a surge of interest in neural architecture evolution (NAE) [3], [5]. NAE first searches for an appropriate architecture on a small dataset and then transfers the architecture to larger dataset, by simply adjusting weights. One key to the success of NAS or NAE is the use of large-scale data, suggesting that more data leads to better performance. The problem, however, is that the cost of labeling these larger datasets may become prohibitive as their size grows.

In scenarios where we can not obtain sufficient annotations, self-supervised learning is a popular approach to leverage the mutual information of unlabeled training data. However, the performance of unsupervised methods is still unsatisfactory compared with the supervised methods. One obstacle is that only the parameters are learned using conventional self-supervised methods. To address the performance gap, a natural idea is to explore the use of NAE to optimize the architecture along with parameter training. Tackling these bottlenecks will allow us to design and implement efficient deep learning systems that will help to address a variety of practical applications. Specifically, we can initialize with an architecture found using NAS on a small supervised dataset and then evolve the architecture on a larger dataset using unsupervised learning. Currently, existing architecture evolution methods [3], [5] are inefficient and can not deal effectively with unsupervised representation learning. Our approach is extremely efficient with a complexity of $O(n^2)$ where n is the size of the operation space. Here we propose our Fast and Unsupervised Neural Architecture Evolution (FaUNAE) method to search architectures for representation learning. Although UnNAS [6] discusses the value of a label and discovers that labels are not necessary for NAS, it cannot solve the aforementioned problems because it is computationally expensive and is trained using supervised learning for real applications. FaUNAE is introduced to evolve an architecture from an existing architecture manually designed or searched from one small-scale dataset on another large-scale dataset. This partial optimization can utilize the existing models to reduce the search cost and improve search efficiency. The strategy is more practical for real applications, as it can efficiently adapt to the new scenarios' minimal data labeling requirements.

First, we adopt a trial-and-test method to evolve the initial architecture, which is more efficient than the traditional evolution methods, which are computationally expensive and require large amounts of labeled data. Second, we note that the quality of the architecture is hard to estimate due to the absence of labeled data. To address this, we explore contrastive loss [1] as the evaluation metric for the operation evaluation. Although our method is built based on contrastive loss [1], we model our method on the teacher-student framework to mimic supervised learning and then estimate the operation performance even without annotations. Then the architecture can be evolved based on the estimated performance. Third, we address the fact that one bottleneck in NAS is its explosive search space of up to 14^8. The search space issue is even more challenging for unsupervised NAS built on an ambiguous performance estimation that further deteriorates the training process. To address this issue, we build our search algorithm based on the principles of survival of the fittest and elimination of the inferior. This significantly improves search efficiency. Our framework is shown in Fig. 1. Our contributions are as follows:

- ❏ We propose unsupervised neural architecture evolution, which utilizes prior knowledge to search for an architecture using large-scale unlabeled data.
- ❏ We design a new search block and limit the model size according to the initial architecture during evolution to deal with the huge search space of ResNet. The search space is further reduced through a contrastive loss in a teacher-student framework by abandoning operations with less potential to significantly improve the search efficiency.
- ❏ Extensive experiments demonstrate that the proposed methods achieve better performance than prior art on ImageNet, PASCAL VOC, and COCO.

II. Related Work

A. Unsupervised Learning

Recent progress on unsupervised/self-supervised[1] learning began with artificially designed pretext tasks, such as patch

[1] Self-supervised learning is a form of unsupervised learning.

relative location prediction [7] and rotation prediction [8]. Large scale experiments on these pretext tasks [9] reveal that neither is the ranking of architectures consistent across different methods nor is the ranking of methods consistent across architectures. Thus, pretext tasks for self-supervised learning should be considered in conjunction with their underlying architectures.

Self-supervised representation learning can also be formulated as learning an embedding where the semantically similar images are very close, and the semantically different images are far away. One method is to contrast positive pairs with negative pairs. Contrastive learning [10] learns mappings that are invariant to certain transformations of the inputs. He, et al. [1] build a dynamic dictionary with a queue and a moving-averaged encoder using a contrastive loss. He, et al. [1] update the dictionary in the form of a queue, decoupling the size of the dictionary and the batch size, so that the size of the dictionary does not need to be constrained by batch size. In addition, the Momentum-based Moving Average keeps the encoder updated all the time. Subsequently, Chen, et al. [11] combine the methods of [1], [12], and use an MLP projection head and more data Augmentation to further improve performance.

B. Neural Architecture Search

With the rapid development of computer vision, various network structures have achieved significant performance improvements, but most of them are manually designed. Recently, a method called NAS has attracted more and more attention, which can automate neural net architecture design [2], [13]. NAS performs full-fledged training and validation over thousands of architectures from scratch. Traversing all candidate architectures in a very large search space requires tremendous computation and memory resources. A cell-based architecture [3] is widely used to reduce the size of search space by stacking cells to make NAS more efficient. However, the search space is still noncontinuous and high-dimensional. A *one-shot* architecture search method [14], [15] is proposed for learning efficiency, making it possible to identify an optimal architecture within a few GPU days. Pham, et al. [15], share parameters among child models and trains a controller to search the network architecture. Liu, et al. [14] introduce a differentiable framework that enables NAS to use gradient descent for search. Due to the considerable memory requirement of [14], [16] proposes a partial channel connection method, which reduces memory consumption by sampling a small part of the network and improves search efficiency. Chen, et al. [17] successfully apply NAS to model defense, sampling the network through the combination of valid accuracy and bandit, and achieve good results in adversarial training. The use of NAS for unsupervised learning is a new research hotspot, which needs further exploration.

C. Object Detection and Segmentation

Object detection [18], [19] is one of the basic tasks of computer vision. Object detection algorithms typically include one-stage methods and two-stage methods. The two-stage methods use a sliding window detector. First, the algorithm generates a series of anchors as samples and then classifies the samples through the convolutional neural network. Common algorithms include R-CNN [20], Fast R-CNN [21], and Faster R-CNN [22]. The one-stage methods use a single shot detector, and the methods are divided into anchor-based method and anchor-free methods. The anchor-free method does not generate anchors and directly transforms the problem of anchor positioning into regression. Common algorithms are YOLO [23], SSD [24] and so on. Compared with object detection, instance segmentation requires more stringent requirements. It not only requires segmenting the object's mask and identifying its category but also needs to distinguish different instances in the same category. The main algorithms include Mask R-CNN [25], SOLO [26], and SOLOv2 [27]. For object detection and segmentation, we mainly use Faster R-CNN and Mask R-CNN. Faster R-CNN proposes the Region Proposal Network on the basis of Fast R-CNN and integrates feature extraction, region proposal, bounding box regression (rect refine), and classification into one network, thereby effectively improving detection accuracy and detection efficiency. Mask R-CNN is a framework for instance segmentation, which adds a branch to Faster R-CNN for segmentation. In addition, Mask R-CNN introduces a region of interest (ROI) alignment to replace ROI pooling in Faster R-CNN, which improves the accuracy of the mask.

D. Neural Architecture Evolution

Before the advent of the term "deep learning," neuroevolution [28], [29] used genetic algorithms to change the topology along with hyperparameters and connection weights of artificial neural networks (ANNs) by allowing high performing structures to reproduce and mutate. Recent works focus on evolving deep neural architecture on a larger scale. Real, et al. [13] start small and mutates architectures to navigate large search spaces. Encouraged by the idea of stacking the same modules in many successful architectures such as ResNet and DenseNet, a variant of this method [30] shows better results. It repeatedly evolves small neural network modules by using a larger hand-coded blueprint. Similarly, CoDeepNEAT [31] evolves the topology, the components, and hyperparameters simultaneously for cheaper computation. Cai, et al. [5] use reinforcement learning as a meta-controller and Bi-LSTM as an encoder for function-preserving transformations to make the network wider or deeper. Zoph, et al. [3] use a recurrent neural network as a controller to learn an architectural building block (cell) on a small dataset and then transfer the block to a larger dataset. Hierarchical genetic representation is proposed to mutate from a low level to a high level for complex topologies [32].

Although a lot of advances have been made in NAE, few works have investigated unsupervised learning, partially because of the inefficiency in the search process. We design a new search block and limit the model size according to the initial architecture during evolution. The search efficiency of our method can also be significantly improved by abandoning operations with less potential in terms of contrastive loss.

III. FaUNAE

In this section, we elaborate on how our network efficiently adapts to new datasets with unsupervised learning. We first introduce an evolution strategy to effectively evolve the architecture from an existing architecture to a new one. We then train our network by exploiting the contrastive learning to discriminate between positive samples and negative samples. Our framework is shown in Fig. 1.

A. Search Space

For different datasets, our architecture space is different. For CIFAR10 [33], we search for two kinds of computation cells, normal cells and reduction cells, to build the final architecture. The reduction cells are located at 1/3 and 2/3 of the total depth of the network, and the rest are normal cells. Following [14], the set of operations include skip connection (identity), 3×3 convolutions, 5×5 convolutions, 3×3 depth-wise separable convolutions, 5×5 depth-wise separable convolutions, 3×3 max pooling, 3×3 average pooling, and zero. The search space of a cell consists of operations on all edges.

On ImageNet, we have experimentally determined that for unsupervised learning, ResNet [34] is better than cell-based methods for building an architecture space. We denote this space as $\{\Omega^i\}$, where i represents a given block. Rather than repeating the bottleneck (building block in ResNet) with various operations, however, we allow a set of search blocks shown in Fig.2 (a) with various operations, including traditional convolution with kernel size $\{3, 5\}$, split-attention convolution (SAConv) [35] with kernel sizes $\{3, 5\}$ and radixes $\{2, 4\}$. SAConv is a Split-Attention block [35] which incorporates feature map split attention within the individual network blocks. Each block divides the feature map into several groups and fine-grained subgroups or splits, where the feature representation of each group is determined by the weighted combination of its split representations. This reduces the model size by sharing the 1×1 convolution to improve efficiency. To enable a direct trade-off between depth and block size (indicated by the parameters of the selected operations), we initiate a deeper over-parameterized network and allow a block to be skipped by adding the identity operation to the candidate set of its mixed operation. So the set of operations Ω^i in the ith block consists of $M = 7$ operations. With a limited model size, the network can either choose to be shallower by skipping more blocks and using larger blocks or choose to be deeper by keeping more blocks with smaller size.

The initial structure α_0 is first manually designed (e.g., ResNet-50, without weight parameters) or searched for by another NAS (e.g., ProxylessNAS [4]) on different datasets in a supervised manner[2], then remapped to the search space to accelerate the evolution process and make use of prior knowledge.

[2]The evolution is based on unsupervised learning.

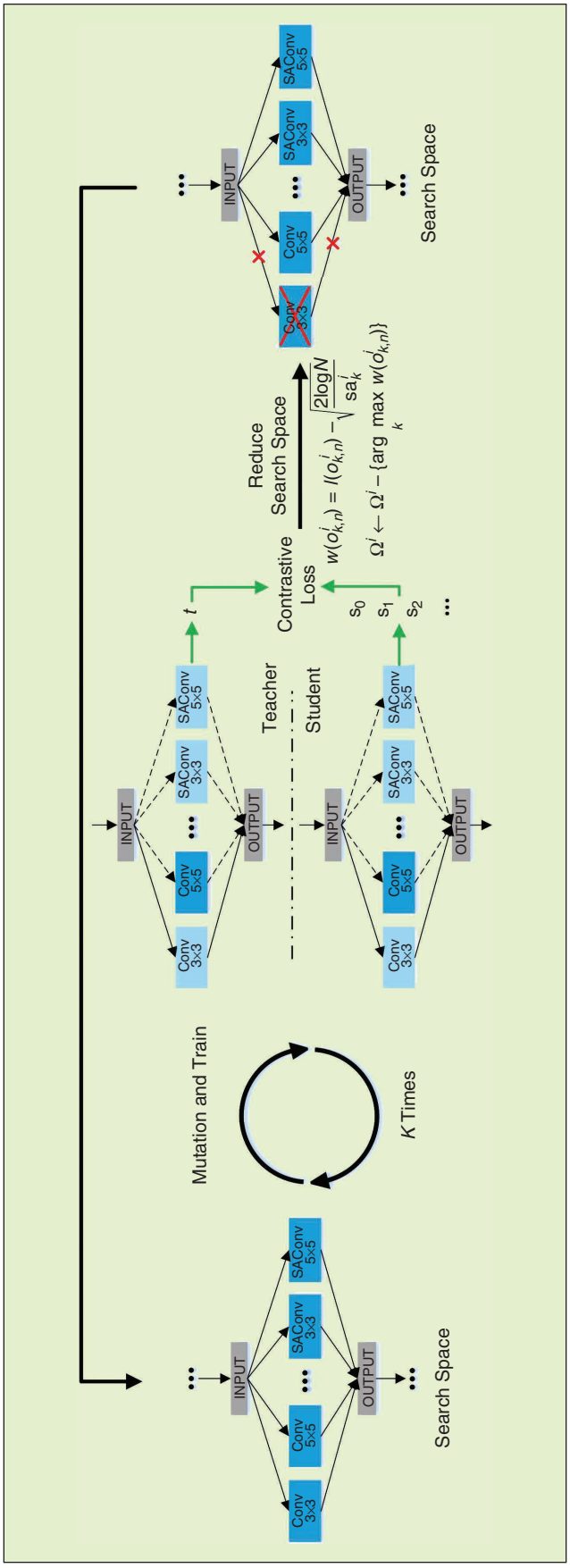

FIGURE 1 The main framework of the proposed Teacher-Student search strategy. Left: Search space of FaUNAE. Middle: Evolution process of FaUNAE. Right: FaUNAE reduces the search space by eliminating the operations with the least potential.

B. Evolution

The evolutionary strategy is summarized in Alg. 1. Unlike AmoebaNet [36], which evaluates the performance of the sub-networks sampled from the search space in a population, our method evolves the operation in each block in a trial-and-test manner. We first mutate the operation based on its mutation probability, followed by an evaluation step to make sure the mutation is ultimately used.

Mutation: An initial structure α_0 is manually designed (e.g., ResNet-50)[3] or searched by another NAS (e.g., ProxylessNAS [4]) on a different dataset using supervised learning. The initial subnetwork f_{θ_t}, which is generated by searching over-parameterized network based on α_0, is then trained using (6) for k steps to obtain the evaluation metric $l(o_k^i)$. A new architecture $\alpha_n(o_{k,n})$ is then constructed from the old architecture α_{n-1} $(o_{k,n-1})$ by a transformation or a mutation. The mutation probability p_{mt} is defined as

$$p_{\text{mt}}(o_{k,n}^i) = \begin{cases} 1-\epsilon, o_{k,n}^i = o_{k,n-1}^i \\ \dfrac{1}{K-1}\left(1 - \dfrac{sa_k^i}{\sum_{k'} sa_{k'}^i}\right)\epsilon, \text{otherwise} \end{cases} \quad (1)$$

where sa_k^i represents the sampling times of the operation o_k^i. The operation o_k^i represents the kth operation in the ith block and n represents current number of evolutions or the index of epoch. In general, the operation in each block is kept constant with a probability $1-\epsilon$ in the beginning. For the mutation, younger (less sample time) operations are more likely to be selected, with a probability of ϵ, Intuitively, keeping the operation constant can be thought of as providing exploitation, while mutations provide exploration [36]. We use two main mutations that we call the depth mutation and the op mutation, as in AmoebaNet [36], to modify the structure generated by the search space described above.

The operation mutation pays attention to the selection of operations in each block. Once the operation in a block is

[3] No weight parameters.

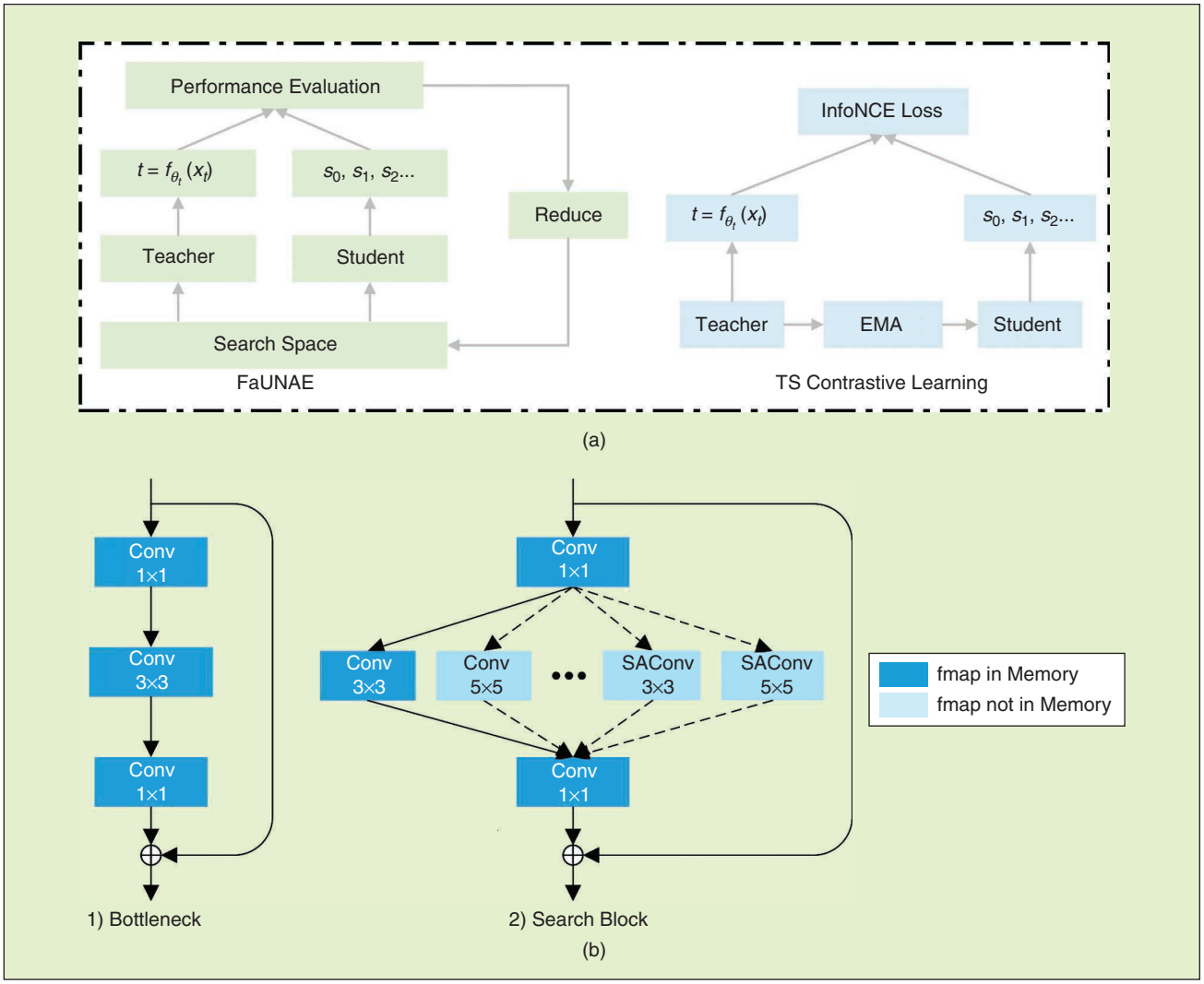

FIGURE 2 (a) The main framework of the Teacher-Student model, which focuses on both the unsupervised neural architecture evolution (left) and contrastive learning (right). (b) Compared with the original bottleneck (1) in ResNet, a new search block is designed for FaUNAE (2)." fmap in memory" means that we select the operation to participate in the feature map calculation.

chosen to mutate, the mutation picks one of the other operations based on (1). The depth mutation can change the depth of the subnetwork in the over-parameterized network by setting the operation of one block to the 'Identity' operation. To search more efficiently and evaluate operations that are more reasonable, we limit the model size as a restriction metric. The structure can then evolve into a subnetwork that has the same computational burden. The probability of the restriction metric p^i_{rm} is defined as

$$p_{\text{rm}}(o^i_{k,n}) = \frac{-\exp \text{MS}(o^i_{k,n})}{\sum_{k'} \exp \text{MS}(o^i_{k',n})}, \tag{2}$$

where $\text{MS}(o^i_{k,n})$ represents the number of parameters of the kth operation in the ith block and n represents the index of epoch or current number of evolutions. The final evolution probability p that combines p_{mt} and p_{rm} is defined as

$$p(o^i_{k,n}) = \lambda_1 * p_{\text{mt}}(o^i_{k,n}) + (1 - \lambda_1) * p_{\text{rm}}(o^i_{k,n}), \tag{3}$$

where λ_1 is a hyperparameter.

Mutation validation: After each evolution, the subnetwork is trained using (6), and the loss is used as the evaluation metric. We observe the current validation loss a and accordingly update the loss $l(o^i_{k,n})$, which historically indicates the validation loss of all the sampled operations $o^{(i,j)}_k$ as

$$l(o^i_{k,n}) = \lambda_2 * l(o^i_{k,n-1}) + (1 - \lambda_2) * a, \tag{4}$$

where λ_2 is a hyperparameter. If the operation which is mutated performs better (less loss), we apply it as the base of the next evolution; otherwise, we use the original operation as the base of the next evolution.

$$o^i_{k,n} = \begin{cases} o^i_{k,n}, l(o^i_{k,n}) < l(o^i_{k,n-1}) \\ o^i_{k,n-1}, \text{else} \end{cases} \tag{5}$$

C. Contrastive Learning

Contrastive learning [37] greatly improves the performance of unsupervised visual representation learning. The goal is to keep the negative sample pairs away and the positive samples close in the latent space. Prior works [1], [38] usually investigate contrastive learning by exploring the sample pairs calculated from the encoder and the momentum encoder [1]. Based on the investigation, we reformulate the unsupervised/self-supervised NAS as a teacher-student model, as shown in Fig. 2 (a). Following [1], we build dynamic dictionaries, and the "keys" t in the dictionary are sampled from data and are represented by the teacher network. In general, the key representation is $t = f_{\theta_t}(x_t)$, where $f_{\theta_t}(.) = f(o^i_{k,n}; \theta_t; .)$ is a teacher network, and x_t is a key sample. Likewise, the "query" x_s is represented by $s = f_{\theta_s}(x_s)$, where $f_{\theta_s} = f(o^i_{k,n}; \theta_s; .)$ is a student network. Unsupervised learning performs dictionary look-up by training a network of students. The student model and teacher model are subnetworks of NAE from the over-parameterized network described in 3.1.

We use contrastive loss to match an encoded query s with the encoded key dictionary to train the visual representation student model. The value of the contrastive loss is lower when s and t are from the same (positive) sample and higher when s and t are from different (negative) samples. The contrastive loss is also deployed in FaUNAE as a measure to guide the evolution of the structure to obtain the optimal structure based on the unlabeled dataset. InfoNCE [39] shown in Fig.2 (a), measures the similarity using the dot product and is used as our evaluation metric:

$$\mathcal{L} = -\log \frac{\exp(s \cdot t_+ / \tau)}{\sum_{n=0}^{N} \exp(s \cdot t_n / \tau)}, \tag{6}$$

where τ is a temperature hyperparameter per [40] and t_+ represents the feature calculated from the same sample with s. InfoNCE is over one positive and M negative samples. Our method is general and can be based on other contrastive loss functions [37], [40]–[42]. Following [1], [43], the teacher model is updated as an exponential moving average (EMA) of the student model:

$$\theta_t = m * \theta_t + (1 - m) * \theta_s, \tag{7}$$

where θ_s and θ_t are the weight of the student model and the teacher model, respectively, updated by back-propagation in contrastive learning, and $m \in [0, 1)$ is a smoothing coefficient hyperparameter.

Algorithm 1 **FaUNAE.**

Input: Training data, validation data, and the initial structure α_0
Parameter: Searching hypergraph
Output: Optimal structure α

```
Let n = 1.
while (K > 1) do
  for t = 1,...,T epoch do
    Evolve architecture α_n from the old architecture α_{n-1}
    based on the evolution probability p using (3);
    Construct the Teacher model and Student model with
    the same architecture α_n, and then train Student mod-
    els by gradient descent and update Teacher model by
    EMA using (7);
    Get the evaluation loss on the validation data using (6);
    Use (4) to calculate the performance and assign it to all
    sampled operations;
    Update α_n using (5);
  end for
  if t == K * E then
    Update w(o^i_{k,n}) using (9);
    Reduce the search space: Ω^i ← Ω^i − {argmax_k w(o^i_{k,n})};
    K ← K − 1;
  end if
end while
return α
```

D. Fast Evolution by Eliminating Operations

One of the most challenging aspects of NAS lies in the inefficient search process, and we address this issue by eliminating the operations with the least potential. The multi-armed bandit problem is similar to the NAS problem. The Upper Confidence Bound (UCB) is used for dealing with the exploration-exploitation dilemma in the multi-armed bandit problem.

$$s = r_k + \sqrt{\frac{2\log N}{n_k}}, \tag{8}$$

where r_k represents the average reward obtained by arm k. n_k represents the number of times arm k has been selected in the total number of trials N.

After $|\Omega^i| * E$ epochs, we remove the operations in each block based on performances (loss) and the sampling times. Inspired by the UCB algorithm, we define the combination of the two as

$$w(o^i_{k,n}) = l(o^i_{k,n}) - \sqrt{\frac{2\log N}{\mathrm{sa}^i_k}}, \tag{9}$$

where N is the total number of evolutions and mutations, and sa^i_k is the number of times the kth operation of the ith block has been selected. The first item $l(o^i_{k,n})$ in (9) is calculated based on an accumulation of the validation loss, and the second term is the exploration term. The operation with the maximum w

FIGURE 3 Some pictures and corresponding labels of the CIFAR10 dataset.

for every block is abandoned. This means that the operations that are given more opportunities but result in poor performance are removed. In order to improve search efficiency, we eliminate the operation with the maximum w for every block simultaneously. With this strategy, the search space which has v blocks is significantly reduced from $|\Omega^i|^v$ to $(|\Omega^i|-1)^v$, and the reduced space becomes

$$\Omega^i \leftarrow \Omega^i - \{\operatorname{argmax}_k w(o^i_{k,n})\}. \tag{10}$$

The reduction procedure is carried out repeatedly until the optimal structure is obtained when only one operation is left in each block.

IV. Experiments

In this section, we describe experiments on the ImageNet and CIFAR10 datasets and compare FaUNAE with other NAS methods and human-designed networks. In addition, we describe experiments on the PASCAL VOC and COCO datasets. The evolved architecture on ImageNet is applied as the backbone of object detection.

A. Datasets

1) CIFAR-10

CIFAR-10 (Fig. 3) is a dataset containing 60,000 images. Fifty thousand pictures are used for the training set, and the remaining ten thousand are used for the test set. Each photo is a 32 × 32 color photo, each pixel includes three values of RGB, and the value range is 0-255. All photos belong to 10 different categories.

2) ImageNet

ImageNet is currently the world's largest image recognition database. It is an ongoing research effort aimed at providing an easy-to-access image database for researchers around the world. At present, there are tens of millions of images in ImageNet, and more than 20000 categories.

3) PASCAL VOC

VOC2007 contains 9963 annotated pictures and 20 categories split into training, validation, and testing data with 24,640 objects marked. There are 11530 images in VOC2012. For the detection task, the trainval of VOC2012 has 11540 images and a total of 27450 objects.

4) MS COCO

The coco dataset is widely used in areas such as object detection and instance segmentation. COCO2017 contains 118,287 training images, 5000 validation images, and 40670 testing, with more than 80 object categories.

B. Evolution and Training Protocol

The evolution and training protocol used in our experiments are described in this section. We first set a global average pooling and a 2-layer MLP [12] head (hidden layer 2048-d, with ReLU), which has a fixed-dimensional output (128-d [40]) after the hypernet and searched network. The temperature τ in 6 is set to 0.2 [40], and the smoothing coefficient hyperparameter m in (7) is set as 0.999. We use the same data enhancement settings as MoCoV2 [11]

Experiments first evolve the initial structure α_0 on an over-parameterized network, which uses ResNet50 as the backbone to build the architecture space (details can be found

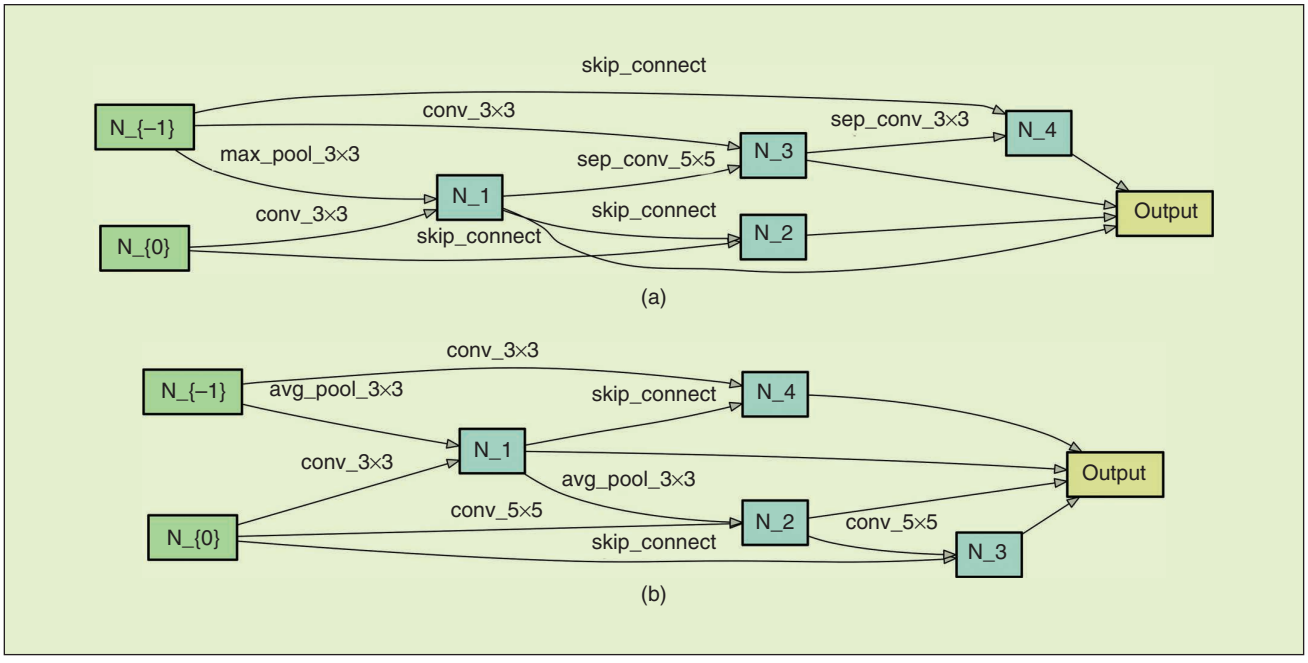

FIGURE 4 Detailed structure of the best cells discovered on CIFAR-10. (a) normal cells found on CIFAR10. (b) reduction cells found on CIFAR10.

in 3.1) on ImageNet. We set the initial structure α_0 as a random structure, ResNet50, and the structure searched by Proexyless on ImageNet100, respectively, to show the importance of prior knowledge. During the architecture search, the 128M training samples of ImageNet are divided into two subsets, 80% for the training set, and the remainder for the validation set for mutation validation and search space reduction. We set the channel as half of that of ResNet50 for efficiency and pay attention to the evolution of the operation rather than the channel. We set the hyperparameter $E = 3$, so the number of the epoch of the search is $\sum_{m=2}^{M} k * E$. The hyperparameter λ_1 and λ_2 are set to 0.9 and 0.3. After evolution, we train the searched network on ImageNet in an unsupervised manner for 200 epochs.

C. Results for Classification

In this section, we perform experiments on CIFAR10 and ImageNet. In the experiment using CIFAR10, every time we train an epoch, we immediately verify the performance on the validation set. In the experiment using ImageNet, we perform unsupervised pretraining and then train a supervised linear classifier for 100 epochs. Finally, we report top-1 classification accuracy on the validation set.

1) Results on CIFAR10

On CIFAR10, we use an experimental setting described in MoCo v1 [1], not the settings used in v2 [11] for ImageNet. In addition, we use cell-based search space. In the search process, we search for normal cells and reduction cells. The total number of layers is set to 6, and reduction cells are located at 1/3 and 2/3. We set the intermediate nodes to $M = 4$, the momentum to 0.9, the weight decay to 3×10^{-4} and the initial learning rate to 0.025 described in [14]. We use 90% of the training set as the training data in the search phase, and 10% of the training set as the validation data. After the search, we train the final architecture for 600 epochs on CIFAR10. We set the total number of cell layers to 10 and the batch size to 128.

We use two NVIDIA TitanV GPUs to search approximately 1.5 hours. We set the initial number of channels to 36 and 100 for better performance. It can clearly be seen in Table 1 whether it is FaUNAE (channel = 36) or FaUNAE (channel = 100), the model size of FaUNAE is smaller than that of $\text{AMDIM}_{\text{large}}$ and $\text{AMDIM}_{\text{small}}$. Note that the results in Table 1 are our best, but they are unstable and we need additional trials to obtain results. Compared with $\text{AMDIM}_{\text{large}}$, FaUNAE (channel = 100) achieves not only a better performance (91.2 vs. 92.26), but also has fewer parameters (626M vs. 69.8M). When compared with ResNet18, FaUNAE is slightly better than ResNet in terms of model size (11.1M vs. 9.1M), and FaUNAE improved performance by 4.74% (84.09 vs. 88.83).

TABLE 1 Comparison under the linear classification protocol on CIFAR10.

ARCHITECTURE	METHOD	ACCURACY (%)	PARAMS (M)	SEARCH COST (GPU DAYS)	SEARCH METHOD
$\text{AMDIM}_{\text{SMALL}}$	AMDIM [44]	89.5	194	–	Manual
$\text{AMDIM}_{\text{LARGE}}$	AMDIM [44]	91.2	626	–	Manual
RESNET18	MOCO V1 [1]	84.09	11.1	–	Manual
FAUNAE (CHANNEL = 36)	MOCO V1	88.83	9.1	**0.125**	Evolution
FAUNAE (CHANNEL = 100)	MOCO V1	92.26	69.8	**0.125**	Evolution

TABLE 2 Comparisons under the linear classification protocol on ImageNet.

ARCHITECTURE	METHOD	ACCURACY (%)	PARAMS (M)	SEARCH COST (GPU DAYS)	SEARCH METHOD
RESNET50	INSTDISC [40]	54.0	24	–	Manual
RESNET50	LOCALAGG [45]	58.8	24	–	Manual
RESNET101	CPC V1 [42]	48.7	28	–	Manual
$\text{RESNET170}_{\text{WIDER}}$	CPC V2 [46]	65.9	303	–	Manual
$\text{RESNET50}_{\text{L+AB}}$	CMC [47]	64.1	47	–	Manual
$\text{AMDIM}_{\text{SMALL}}$	AMDIM [44]	63.5	194	–	Manual
$\text{AMDIM}_{\text{LARGE}}$	AMDIM [44]	68.1	626	–	Manual
RESNET50	MOCO V1 [1]	60.6	24	–	Manual
RESNET50	MOCO V2 [11]	67.5	24	–	Manual
RESNET50	SIMCLR [12]	66.6	24	–	Manual
RANDOM	MOCO V2	66.2	23	–	Random
PROXYLESSNAS	MOCO V2	67.8	23	23.1	Gradient-base
FAUNAE (RANDOM)	MOCO V2	67.4	24	**15.3**	Evolution
FAUNAE (RESNET50)	MOCO V2	67.8	24	**15.3**	Evolution
FAUNAE (PROXYLESSNAS)	MOCO V2	68.3	30	**15.3**	Evolution

2) Results on ImageNet

During the search, we set an initial learning rate of 0.03, a momentum of 0.9, a weight decay of 0.0001, and then we use 8 Tesla V100 GPUs to search approximately 46 hours. In the process of training a supervised linear classifier, we set the initial learning rate as 30 and the weight decay 0 as described in [1]. Table 2 shows that FaUNAE outperforms ResNet50, ResNet101, ResNet170,

FIGURE 5 Detailed structures of the best structure discovered on ImageNet. "SAConv2" and "SAConv4" denote split-attention bottleneck convolution layer with radix of 2 and 4, respectively.

and $AMDIM_{small/large}$ with higher accuracy. Compared with $AMDIM_{large}$, which is by far the best method of manual design that we know of, FaUNAE (ProxylessNAS) achieves not only a better performance (68.1 vs. 68.3), but the model size was also reduced by about 20 times (626M vs. 30M). FaUNAE also performs better than the structure sampled randomly from the search space described in 3.1 on Top-1 accuracy (68.3 vs. 66.2). When compared with other NAS methods which use the same search space, such as ProxylessNAS, FaUNAE obtains a better performance with higher accuracy (67.8 vs. 68.3) and is much faster (23.1 vs. 15.3 GPU days). In addition, we calculated the FLOPs of different models separately. The FLOPs of ResNet50, FaUNAE (ResNet50) and FaUNAE (ProxylessNAS) are respectively: 4.11G, 3.87G and 4.08G. When the FLOPs are approximately the same, FaUNAE (ProxylessNAS) achieves better performance.

We also set different initial structures α_0 including random structure, ResNet50, and the structure searched by ProxylessNAS on ImageNet100. As shown in Table 2, we find that the better the initial structure, the better the performance, which shows the importance of the prior knowledge. For the structure (Fig. 5) obtained by FaUNAE on ImageNet, we find that the structure on unsupervised learning prefers a small kernel size and a split-attention convolution [35], which also shows the effectiveness of split-attention convolution and the rationality of FaUNAE. We searched the architecture on ImageNet for many times, which needs further research in our future work.

D. Results on Object Detection and Segmentation

Learning transferable features is the main goal of unsupervised learning. When used as fine-tuning initialization for object detection and segmentation, ImageNet supervised pre-training is the most influential (e.g., [21, 22, 20]). Next, we perform experiments on the COCO [18] data set and the PASCAL VOC [19] data set, and compared FaUNAE with other methods.

1) Results on PASCAL VOC

We use Faster R-CNN [22] as the detector and the evaluated network obtained on ImageNet as the backbone, with batch normalization tuned, implemented in [48]. All layers are fine-tuned end-to-end. We fine-tune for 18k iterations (~18 epochs) on PASCAL VOC trainval07+12. We evaluate the default VOC metric of AP50 on the VOC test2007 set.

In Table 3, we can see the results on trainval07+12. FaUNAE with ImageNet is better than ResNet50 with the same method (MoCo v2): up to + 0.9 AP50, + 2.3 AP, and + 2.6 AP75. Also, our FaUNAE is better than the ImageNet supervised counterparts, which shows the usefulness and effectiveness of unsupervised learning.

TABLE 3 Results under object detection on PASCAL VOC with Faster R-CNN.

ARCHITECTURE	METHOD	AP_{50}	AP	AP_{75}
RESNET50	Super.	81.3	53.5	58.8
RESNET50	MOCO V1 [1]	81.5	55.9	62.6
RESNET50	MOCO V2 [11]	82.4	57.0	63.6
FAUNAE	MOCO V2	83.3	59.3	66.2

TABLE 4 Results of object detection and instance segmentation on COCO with Mask R-CNN. AP^{bb} means bounding-box AP and AP^{mk} means mask AP.

ARCHITECTURE	METHOD	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
RESNET50	Super.	40.0	59.9	43.1	34.7	56.5	36.9
RESNET50	MOCO V1 [1]	40.7	60.5	44.1	35.4	57.3	37.6
FAUNAE	MOCO V2	43.1	63.0	47.2	37.7	60.2	40.6

2) Results on COCO

We apply Mask R-CNN [25] with the C4 backbone as the detector. The rest of the settings are the same as those for the PASCAL VOC experiment. In Table 4, we can see the results on the COCO dataset with the C4 backbone. With the $2\times$ schedule, FaUNAE is better than its ImageNet supervised counterpart on all metrics. Due to the absent result of MoCo v2 [11], we do not compare it with our FaUNAE. We run their code for this comparison, which is even worse than v1. Also, FaUNAE is better than ResNet50 trained with unsupervised MoCo v1 [1].

V. Conclusions

We proposed a fast and unsupervised neural architecture evolution (FaUNAE) to evolve an existing architecture manually designed or searched for on one small dataset to a new architecture on another larger dataset. The evolution is self-supervised, where contrast loss is used as the evaluation metric in the student-teacher framework. The evolution process is significantly accelerated by eliminating the inferior or the least promising operation. Experimental results demonstrate the effectiveness of our unsupervised NAE for the downstream applications, such as object recognition, object detection, and instance segmentation. FaUNAE addresses core technical challenges in improving the performance of unsupervised deep learning to make it more scalable and applicable, which will ultimately impact many meaningful applications.

Acknowledgments

This study was supported by Grant NO.2019JZZY011101 from the Key Research and Development Program of Shandong Province to Dianmin Sun. This work was supported in part by the National Natural Science Foundation of China under Grant 62076016 and 61876015.

References

[1] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. CVPR*, 2020, pp. 9726–9735.
[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR,* Toulon, France, Apr. 24–26, 2017.
[3] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. CVPR*, 2018, pp. 8697–8710.
[4] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. ICLR*, 2019.
[5] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI*, 2018, pp. 2787–2794.
[6] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?" 2020, arXiv:2003.12056.
[7] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. ICCV*, 2015, pp. 1422–1430.
[8] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. ICLR*, 2018.
[9] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proc. CVPR*.
[10] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. CVPR*, 2006.
[11] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," 2020, arXiv:2003.04297.
[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020, arXiv:2002.05709.
[13] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. ICML*, pp. 2902–2911.
[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. ICLR*, 2019.
[15] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proc. ICML*, 2018, pp. 4092–4101.
[16] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *Proc. ICLR*, 2020.
[17] H. Chen, B. Zhang, S. Xue, X. Gong, H. Liu, R. Ji, and D. Doermann, "Anti-bandit neural architecture search for model defense," 2020, arXiv:2008.00698.
[18] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, Springer-Verlag, 2014, vol. 8693, pp. 740–755.
[19] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010. doi: 10.1007/s11263-009-0275-4.
[20] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.
[21] R. Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015, pp. 1440–1448.
[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NeurIPS*, 2015, pp. 91–99.
[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR,* Las Vegas, NV, 2016, pp. 779–788.
[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Comput. Vision - ECCV 2016 - 14th Eur. Conf.*, Amsterdam, The Netherlands, Springer-Verlag, 2016, vol. 9905, pp. 21–37.
[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, 2017, pp. 2980–2988.
[26] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," *CoRR*, vol. abs/1912.04488, 2019.
[27] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic, faster and stronger," 2020, arXiv:2003.10152.
[28] D. Floreano, P. Durr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evol. Intell.*, vol. 1, pp. 47–62, 2008. doi: 10.1007/s12065-007-0002-4.
[29] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," MIT Press Journals, 2002.
[30] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI*, 2019, pp. 4780–4789.
[31] R. Miikkulainen et al., "Evolving deep neural networks," 2017, arXiv:1703.00548.
[32] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. ICLR*, 2018.
[33] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.
[35] H. Zhang et al., "Resnest: Split-attention networks," 2020, arXiv:2004.08955.
[36] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI*, 2019, pp. 4780–4789. Doi: 10.1609/aaai.v33i01.33014780.
[37] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. CVPR*, 2006, pp. 1735–1742.
[38] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in *Proc. ICLR*, 2020.
[39] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, arXiv:1807.03748.
[40] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proc. CVPR*, 2018, pp. 3733–3742.
[41] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proc. CVPR*, 2015, pp. 2794–2802.
[42] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *Proc. ICLR*, 2019.
[43] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. NIPS*, 2017.
[44] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Proc. NeurIPS*, 2019, pp. 15,509–15,519.
[45] C. Zhuang, A. L. Zhai, and D. Yamins, "Local aggregation for unsupervised learning of visual embeddings," in *Proc. ICCV*, 2019, pp. 6001–6011.
[46] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, "Data-efficient image recognition with contrastive predictive coding," 2019, arXiv:1905.09272.
[47] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," 2019, arXiv:1906.05849.
[48] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." 2019. https://github.com/facebookresearch/detectron2