ZeroNAS: Differentiable Generative Adversarial Networks Search for Zero-Shot Learning

Caixia Yan, Xiaojun Chang*, Zhihui Li, Weili Guan, Zongyuan Ge, Lei Zhu, and Qinghua Zheng

Abstract—In recent years, remarkable progress in zero-shot learning (ZSL) has been achieved by generative adversarial networks (GAN). To compensate for the lack of training samples in ZSL, a surge of GAN architectures have been developed by human experts through trial-and-error testing. Despite their efficacy, however, there is still no guarantee that these hand-crafted models can consistently achieve good performance across diversified datasets or scenarios. Accordingly, in this paper, we turn to neural architecture search (NAS) and make the first attempt to bring NAS techniques into the ZSL realm. Specifically, we propose a differentiable GAN architecture search method over a specifically designed search space for zero-shot learning, referred to as ZeroNAS. Considering the relevance and balance of the generator and discriminator, ZeroNAS jointly searches their architectures in a min-max player game via adversarial training. Extensive experiments conducted on four widely used benchmark datasets demonstrate that ZeroNAS is capable of discovering desirable architectures that perform favorably against state-of-the-art ZSL and generalized zero-shot learning (GZSL) approaches. Source code is at https://github.com/caixiay/ZeroNAS.

Index Terms—Differentiable Architecture Search, Generative Adversarial Networks, Zero-shot Learning

1 Introduction

Generative Adversarial Networks (GANs) have shown promising results in generating data that are indistinguishable from real data [1], [2], [3]. Recently, a trend has emerged of synthesizing Convolutional Neural Network (CNN) features using GAN architectures, which mitigates the lack of unseen samples in zero-shot learning (ZSL) [4], [5], [6]. Of these methods, f-CLSWGAN [4] is one of the first attempts to leverage GANs in order to push the ZSL performance forward. In an attempt to progress this field, some improved approaches (*e.g.*, LisGAN [7] and AFC-GAN [8]) that may potentially offer better performance, have been proposed. However, despite the empirical success of these approaches, it should be noted that they all rely heavily on hand-crafted GAN architectures designed by human experts, meaning that laborious trial-and-error testing is required (Fig. 1(a)). The instability issue in GAN training increases the difficulty of architecture design significantly. Once obtained, these manually designed architectures are fixed across all diversified data samples and application scenarios, which can easily lead to sub-optimal results. It is therefore highly valuable to automatically determine the GAN architectures customized for each specific ZSL task, rather than simply adopting a hand-crafted architecture.

Neural Architecture Search (NAS) [9], [10] is an effective method of automatic model design that has attracted increasing research attention in recent years. Most existing NAS methods focus on discovering state-of-the-art models for discriminative tasks, such as image classification [11] and object detection [12]. Some initial attempts have also been made in recent literature to develop NAS methods for GANs-based generative tasks. As can be seen from Fig. 1(b), AutoGAN [13] and AGAN [14] are based on Reinforcement Learning (RL), which takes the Inception Score (IS) of numerous candidate architectures as a reward to update the controller. The calculation of IS requires hundreds of images generated by the candidate to be classified by a pre-trained Inception model [15]. Under this evaluation, AGAN [14] requires 200 GPUs over six days to find a good model on CIFAR-10, which is computationally expensive

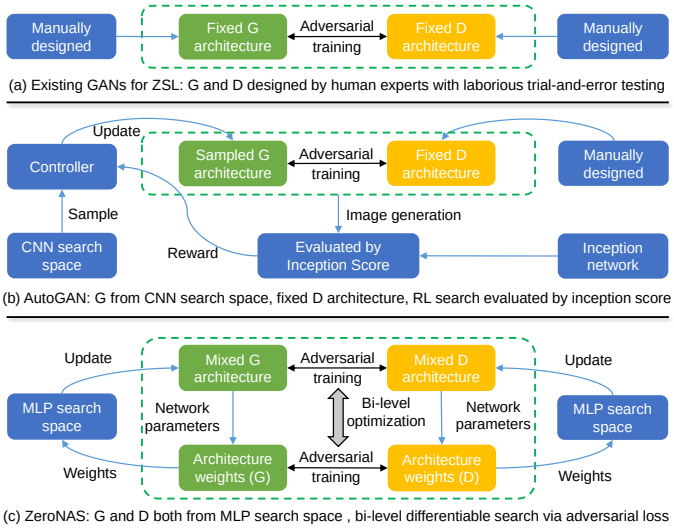


Fig. 1. Comparison of the architecture design and training method of (a) Existing GANs for ZSL, (b) AutoGAN, (c) ZeroNAS.

- *C. Yan and Q. Zheng are with Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China.*
- *X. Chang is with School of Computing Technologies, RMIT University, Melbourne, Australia.*
- *Z. Li is with Shandong Artificial Institute, Qilu University of Technology, Jinan, China.*
- *W. Guan and Z. Ge are with Faculty of Information Technology, Monash University, Melbourne, Australia.*
- *L. Zhu is with School of Information Science and Engineering, Shandong Normal University.*

Corresponding author: Xiaojun Chang (xiaojun.chang@rmit.edu.au).

Manuscript received October 29, 2020, revised June 23, 2021, accepted November 8, 2021.

to a prohibitive extent. To resolve this issue, Doveh *et al.* [16] propose DEGAS to exploit the differentiable search strategy for global latent optimization, resulting in a GAN search that is both efficient and stable. However, DEGAS is designed to search for a generator only, and then directly uses this generator to replace the generator in any given GAN framework, *e.g.*, CTGAN [17]. This strategy fails to consider the compatibility between generator and discriminator, which can be adversely affected by the training instability of GANs. Furthermore, AutoGAN, AGAN and DEGAS are all designed to search the optimal CNN for real image generation. Although the searched GAN architectures allow for the generation of realistic and sharp images conditioned on object categories, these approaches are not yet able to generate images of sufficient quality to train classifiers, as suggested by empirical evidences [4]. Thus, it is far from optimal to directly extend these methods to ZSL tasks for data augmentation.

To address the above mentioned challenges, we propose a differentiable generative adversarial network architecture search method for zero-shot learning, referred to as ZeroNAS (Fig. 1(c)). To adapt to the ZSL task, we design a Multi-Layer Perceptron (MLP) search space for both the generator and discriminator in order to enable feature synthesis. The MLP search space of ZeroNAS is represented as a directed acyclic graph, which can be specified as a MLP network with flexible number of hidden layers and dimensions. Considering the prohibitively high search cost of RL-based NAS methods [13], [14], we take advantage of the efficient differentiable search strategy [11] to jointly learn the network parameters and architecture weights via bi-level gradient descent optimization. Moreover, inspired by the GAN training process, we take the relevance and balance between the generator and discriminator into consideration, and further propose to jointly search their architectures via adversarial training. In each searching iteration, the discriminator will naturally provide supervision signals for evaluating and updating the generator architecture; moreover, the generator can produce fake samples to guide the optimization of discriminator architecture. Benefiting from this adversarial training process, the two architectures interact with each other in a way that continually improves both of their performances. After the searching phase is complete, we perform edge and operation pruning to derive the final network by removing redundant paths. Once the optimal architecture has been found, it can be flexibly plugged into any existing GAN frameworks for further enhancement of ZSL/GZSL performance. The contributions of the present work can be summarized as follows:

- Considering the varieties in datasets and tasks, we make the first attempt to bring NAS techniques into the realm of ZSL, and thus propose ZeroNAS to formulate the GAN architecture design for ZSL as a NAS problem.
- To endow the architectures with the ability to perform feature synthesis, we design an expressive yet compact MLP search space for both the generator and discriminator, which allows for the generation of novel GANs equipped with flexible hidden-layer dimensions and different kinds of operations.
- Inspired by the GAN parameter training process, we take advantage of the relevance between generator and discriminator to enable joint searching of their architectures via adversarial training.

2 Related Works

In this section, we briefly review related works on the fields relevant to our study: Generative Adversarial Networks for ZSL and Neural Architecture Search.

2.1 Generative Adversarial Networks for ZSL

The previous research literature on zero-shot learning exhibits great diversity. In this section, we focus on the most relevant methods using generative adversarial networks. Due to their excellent ability to generate samples, GAN-based ZSL approaches such as f-CLSWGAN [4] and cycle-CLSWGAN [18], have achieved significant improvements in accuracy. A vast majority of the advances made in the previous literature have emerged from the study of novel GAN frameworks or losses. Specifically, f-CLSWGAN [4] is one of the first attempts that leverages Wasserstein GAN [2] to facilitate ZSL/GZSL tasks. Felix *et al.* [18] incorporated a multi-modal cycle consistency loss term into the framework of [4] that enforces good reconstruction of the original semantic features from the synthetic visual representations. Huang *et al.* [6] combined the GAN architecture with a non-generative component, *i.e.*, a regressor network, that interacts with the discriminator through an additional dual adversarial loss. Moreover, many other GAN frameworks have been developed for ZSL, such as GAZSL [19], Lis-GAN [7] and ZSL-ABP [20], which will be introduced and compared in the experimental part. Despite their empirical success, it is still difficult for a single hand-crafted model to achieve consistently good performance across all the datasets. Inspired by NAS, we present ZeroNAS to automatically find the optimal GAN architectures customized for different ZSL scenarios.

2.2 Neural Architecture Search

Recent years have witnessed significant progress in the development of neural architecture search (NAS) [21], [22], [23], [24], [25] methods, which work to automatically discover good architectures for image classification [11] or segmentation [26] tasks. Most existing NAS approaches apply reinforcement learning [27], [28], evolutionary algorithm [29], [30] or Bayesian optimization [31] approaches to automatically design neural architectures, which treat architecture search as black-box optimization problem over a discrete domain. Despite their impressive empirical performance, however, these architecture search methods are based on a discrete and non-differentiable search space, which is computationally expensive. For example, Real *et al.* [30] takes 3150 GPU days for the whole evolution to obtain a state-of-the-art classification model for CIFAR-10 and ImageNet. To improve efficiency, Liu *et al.* [11] proposed DARTS, which relaxes the search space so that it is continuous and then uses gradient descent to effectively search the architecture. More recently, several differentiable NAS methods built upon [11] have been developed to further improve the performance [32], [33]. Although our work shares

a similar differentiable search strategy with these methods, it also differs from them significantly: while almost all existing methods focus on discovering state-of-the-art models for discriminative tasks (*e.g.*, image classification or object detection), we aim to develop an efficient differentiable NAS method for generative adversarial networks.

3 The Proposed Methodology

3.1 Preliminaries

Problem Definition. In the ZSL/GZSL setting, we are given a seen label set $\mathcal{Y}^s$ and an unseen label set $\mathcal{Y}^u$ with a disjoint constraint, *i.e.*, $\mathcal{Y}^s \cap \mathcal{Y}^u = \varnothing$. The visual examples corresponding to the unseen categories $\mathcal{Y}^u$ are unavailable during training; thus, the training dataset consists only of samples from the seen categories. Letting $\mathcal{X}^s$, $\mathcal{C}^s$ and $\mathcal{Y}^s$ denote the visual, semantic and label space corresponding to seen categories, the training data can be defined as $\mathcal{D}^{tr} = \{(x, y, c_y)|x \in \mathcal{X}^s, y \in \mathcal{Y}^s, c_y \in \mathcal{C}^s\}$, where x denotes the visual feature of an image produced by a pre-trained deep neural network and y refers to the corresponding class label, with c_y being the semantic embedding of class y. The testing set $\mathcal{D}^{te}$ is made up of two subsets, *i.e.*, $\{\mathcal{D}_s^{te}, \mathcal{D}_u^{te}\}$, where $\mathcal{D}_s^{te}$ and $\mathcal{D}_u^{te}$ denote the testing samples from seen and unseen classes respectively. During ZSL testing, only the samples from $\mathcal{D}_u^{te}$ are utilized; by contrast, all the samples in $\mathcal{D}^{te}$ are utilized for GZSL evaluation. Without loss of generality, the goal of ZSL/GZSL is to learn a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$, which can be specified as $f : \mathcal{X}^u \rightarrow \mathcal{Y}^u$ for ZSL and $f : \mathcal{X}^s \cup \mathcal{X}^u \rightarrow \mathcal{Y}^s \cup \mathcal{Y}^u$ for GZSL.

Model Framework. The framework of our model is flexible, meaning that we can build upon any generative model developed for ZSL. Due to its simplicity and effectiveness, we take advantage of the feature generating method f-CLSWGAN proposed by [4]. More specifically, the generator $G : \mathcal{Z} \times \mathcal{C} \rightarrow \tilde{\mathcal{X}}$ takes random Gaussian noise $z \in \mathcal{Z}$ and class embedding c_y ($y \in \mathcal{Y}^s$) as input, and outputs the CNN feature $\tilde{x} \in \tilde{\mathcal{X}}$ of class y. The discriminator $D : \mathcal{X} \cup \tilde{\mathcal{X}} \times \mathcal{C} \rightarrow \mathbb{R}$ transforms both the real and synthesized features, along with the corresponding class embeddings, into a real value in order to distinguish between them. The generator and discriminator are alternately trained to compete with each other via the following min-max optimization problem:

$$\min_{\theta_g} \max_{\theta_d} \mathcal{L}_{WGAN}(\theta_g, \theta_d) + \lambda \mathcal{L}_{CLS}(\theta_g|\theta_c^*), \tag{1}$$

where θ_g and θ_d refer to the training parameters of G and D respectively, while θ_c^* represents the optimal parameters of the pre-trained classifier C. $\mathcal{L}_{WGAN}$ denotes the loss for Wasserstein GAN, and $\mathcal{L}_{CLS}$ is the classification loss that enhances the discriminant ability of the synthesized features. λ is employed to balance the two losses. To adapt this framework from manual design to automatic search, we will elaborate the design of the search space, search algorithm, and pruning procedures of ZeroNAS in the following parts.

3.2 Search Space for GANs

The search space should cover as many candidate networks as possible. Unlike AutoGAN and AGAN, which search for the best CNN architecture, we design the searched generative adversarial networks in our work as Multi-Layer Perceptron (MLP) that can adapt to the ZSL task. To satisfy the requirements of both the generator and discriminator, the MLP search space is designed to involve hidden layers with flexible dimensions and different kinds of operations. As depicted in Figure 2, the network is represented as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an ordered sequence of input and intermediate nodes. Each node v_i corresponds to a latent feature vector f_i, while each edge (v_i, v_j) is associated with a set of candidate operations $\mathcal{O} = \{o\}$ that transform v_i to v_j. The candidate operation set $\mathcal{O}$ is composed of the following operations:

- FC+ReLU
- FC+BatchNorm+ReLU
- FC+LeakyReLU
- FC+BatchNorm+LeakyReLU
- FC+BatchNorm+ReLU+DropOut
- FC+ReLU+DropOut
- FC+BatchNorm+LeakyReLU+DropOut
- FC+LeakyReLU+DropOut.

Each intermediate node v_j can obtain information from its previous node $v_i (i < j)$ via $|\mathcal{O}|$ operation paths. The output of the mixed operation from node v_i to v_j can be formulated as the weighted sum of $\{o(f_i)\}$, where the weights are calculated by applying softmax to $|\mathcal{O}|$ real-valued architecture parameters as:

$$\mathcal{M}(v_i \rightarrow v_j) = \sum_{o_k \in \mathcal{O}} \frac{\exp(a_k^{\{v_i,v_j\}})}{\sum_m \exp(a_m^{\{v_i,v_j\}})} o_k(f_i), \tag{2}$$

where $a^{\{v_i,v_j\}} \in \mathbb{R}^{|\mathcal{O}|}$ refers to the weight vector that indicates the importance of different operations associated with edge (v_i, v_j), while $a_k^{\{v_i,v_j\}}$ denotes the kth element of $a^{\{v_i,v_j\}}$. Subsequently, the feature representation f_j that corresponds to node v_j can be obtained by aggregating the output from all of its predecessors $v_i (\forall i < j)$. Given the varying importance of previous nodes, we further incorporate edge weight parameters with a softmax operation to characterize the contribution of each predecessor; thus, f_j can be derived as follows

$$f_j = \sum_{i<j} \frac{\exp(b_i^{v_j})}{\sum_n \exp(b_n^{v_j})} \sum_{o_k \in \mathcal{O}} \frac{\exp(a_k^{\{v_i,v_j\}})}{\sum_m \exp(a_m^{\{v_i,v_j\}})} o_k(f_i), \tag{3}$$

where $b^{v_j} \in \mathbb{R}^{j-1}$ collects the weight coefficients corresponding to all in-edges of node v_j (*e.g.*, $b_i^{v_j}$ for edge $(v_i, v_j)(\forall i < j)$). Without loss of generality, all edge weights and operation weights can be gathered into parameter sets $\alpha = \{a^{\{v_i,v_j\}}\}$ and $\beta = \{b^{v_j}\}$ respectively. Benefiting from such a densely connected structure with mixed operation, the search space is relaxed so that it is continuous, thereby enabling the gradient descent optimization of GAN architectures.

3.3 Search Algorithm for GANs

Due to the continuous relaxation in Eqs. (2) and (3), both the network parameters θ and architecture weights ω can be explicitly involved into a unified gradient descent optimization process, which can be formulated as the following bi-level optimization problem:

$$\min_{\omega} \mathcal{L}_{val}(\theta^*(\omega), \omega), \quad s.t. \ \ \theta^*(\omega) = \arg\min_{\theta} \mathcal{L}_{tr}(\theta, \omega), \tag{4}$$

where $\mathcal{L}_{tr}$ and $\mathcal{L}_{val}$ represent the loss over training set $\mathcal{D}^{tr}$ and validation set $\mathcal{D}^{val}$ respectively. The formulation

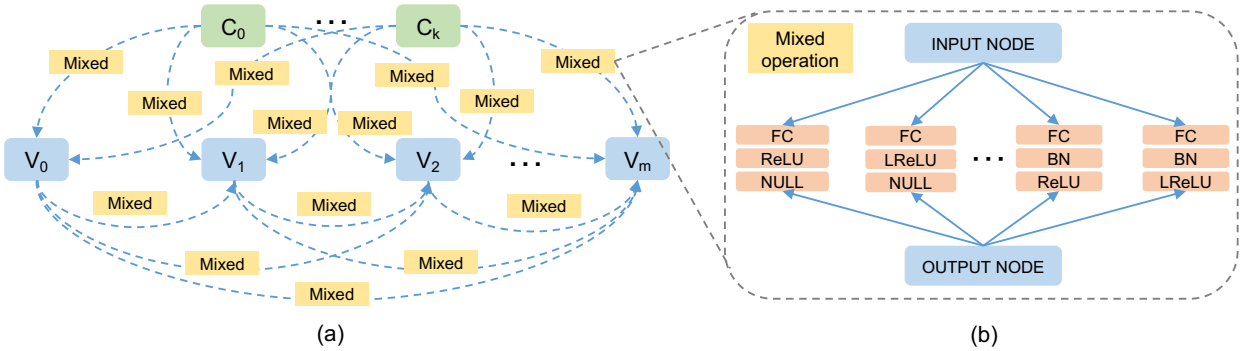


Fig. 2. The search space of the proposed ZeroNAS: (a) Directed acyclic graph with input nodes $C_0 \cdots C_k$, intermediate nodes $V_0 \cdots V_{m-1}$ and output node V_m. (b) Mixed operation composed of predefined candidate operations used for connecting the nodes in directed acyclic graph.

in Eq. (4) can represent NAS on purely discriminant tasks that use a single network (such as image classification). However, for generative adversarial networks that comprise both a generator and discriminator, an additional source of complexity should be considered to reformulate Eq. (4) as a bi-level adversarial learning problem:

$$\begin{aligned} &\min_{\omega_g} \max_{\omega_d} \mathcal{L}_{val}(\theta_g^*(\omega_g), \theta_d^*(\omega_d), \omega_g, \omega_d), \\ s.t. \quad &\theta_g^*, \theta_d^* = \arg\min_{\theta_g} \arg\max_{\theta_d} \mathcal{L}_{tr}(\theta_g, \theta_d, \omega_g, \omega_d), \end{aligned} \tag{5}$$

where θ_g, θ_d refer to two sets of network parameters and $\omega_g = \{\alpha_g, \beta_g\}, \omega_d = \{\alpha_d, \beta_d\}$ denote the architecture weights with respect to the generator and discriminator, respectively. In the lower-level stage, we aim to learn the optimal generator and discriminator parameters, *i.e.*, θ_g^* and θ_d^*, given the specific architecture ω_g and ω_d. In the upper-level stage, moreover, the optimal weights for any pair of architectures $\{\omega_g, \omega_d\}$ can be obtained through adversarial training conditioned on the learned network parameters θ_g^* and θ_d^*.

Given that the inner optimization of network parameters outlined in Eq. (5) is time-consuming, we employ a simplified searching strategy to optimize the four sets of parameters in an alternative manner; namely, one in which each set is optimized with the other three sets frozen in each iteration. More specifically, the parameters in $\{\omega_d, \theta_d\}$ can be optimized with the following losses respectively:

$$\begin{aligned} \mathcal{L}_{val}^d(\omega_d) = &- \mathbb{E}_{x^{val} \sim p_{data}}[D(x^{val}, c_y^{val} | \omega_d, \theta_d^*)] + \\ &\mathbb{E}_{z^{val} \sim p_z}[D(G(z^{val}, c_y^{val} | \omega_g^*, \theta_g^*), c_y^{val} | \omega_d, \theta_d^*)] + \\ &\gamma \mathbb{E}[(\|\nabla_{\hat{x}^{val}} D(\hat{x}^{val}, c_y^{val} | \omega_d, \theta_d^*)\| - 1)^2], \end{aligned} \tag{6}$$

$$\begin{aligned} \mathcal{L}_{tr}^d(\theta_d) = &- \mathbb{E}_{x^{tr} \sim p_{data}}[D(x^{tr}, c_y^{tr} | \omega_d^*, \theta_d)] + \\ &\mathbb{E}_{z^{tr} \sim p_z}[D(G(z^{tr}, c_y^{tr} | \omega_g^*, \theta_g^*), c_y^{tr}) | \omega_d^*, \theta_d)] + \\ &\gamma \mathbb{E}[(\|\nabla_{\hat{x}^{tr}} D(\hat{x}^{tr}, c_y^{tr} | \omega_d^*, \theta_d)\| - 1)^2], \end{aligned} \tag{7}$$

where $(x^{val}, y, c_y^{val}) \in \mathcal{D}^{val}$ and $(x^{tr}, y, c_y^{tr}) \in \mathcal{D}^{tr}$ denote the validation and training data respectively, as sampled from the real data distribution p_{data}; z^{val} and z^{tr} refer to the random noise vector for validation and training respectively, as sampled from Gaussian distribution $p_z \sim \mathcal{N}(0, 1)$; $\hat{x} = \eta x + (1 - \eta) G(z, c_y | \omega_g, \theta_g)$ with $\eta \sim U(0, 1)$; γ is a trade-off parameter. In a similar fashion, the parameters in $\{\omega_g, \theta_g\}$ can be optimized as follows:

$$\begin{aligned} \mathcal{L}_{val}^g(\omega_g) = &- \mathbb{E}_{z^{val} \sim p_z}[D(G(z^{val}, c_y^{val} | \omega_g, \theta_g^*), c_y^{val} | \omega_d^*, \theta_d^*)] - \\ &\lambda \mathbb{E}_{z^{val} \sim p_z}[\log P(y | G(z^{val}, c_y^{val} | \omega_g, \theta_g^*); \theta_c^*)], \end{aligned} \tag{8}$$

$$\begin{aligned} \mathcal{L}_{tr}^g(\theta_g) = &- \mathbb{E}_{z^{tr} \sim p_z}[D(G(z^{tr}, c_y^{tr} | \omega_g^*, \theta_g), c_y^{tr} | \omega_d^*, \theta_d^*)] - \\ &\lambda \mathbb{E}_{z^{tr} \sim p_z}[\log P(y | G(z^{tr}, c_y^{tr} | \omega_g^*, \theta_g); \theta_c^*)], \end{aligned} \tag{9}$$

where $P(y | G(z, c_y | \omega_g, \theta_g); \theta_c^*)$ indicates the probability of the synthesized feature $G(z, c_y | \omega_g, \theta_g)$ being predicted as its true class label y. The ZeroNAS architecture search process is briefly summarized in Algorithm 1.

Algorithm 1 ZeroNAS Architecture Search for GANs.

Input: Training set $\mathcal{D}^{tr}$ and validation set $\mathcal{D}^{val}$; Loop number in an epoch N_{iter}; batch size n_b; iteration number of D in a loop n_d.

Initialize parameters: $\theta_d, \theta_g, \omega_d, \omega_g$
while not converging **do**
  for iter = $1, 2, \cdots, N_{iter}$ **do**
    for i = $1, 2, \cdots, n_d$ **do**
      Sample n_b data from $\mathcal{D}^{val}$ and n_b noise vectors from p_z as validation data;
      Update ω_d using Eq. (6) on validation data: $\omega_d \leftarrow \text{Adam}(\nabla_{\omega_d} \mathcal{L}_{val}^d, \omega_d)$;
      Sample n_b data from $\mathcal{D}^{tr}$ and n_b noise vectors from p_z as training data;
      Update θ_d using Eq. (7) on training data: $\theta_d \leftarrow \text{Adam}(\nabla_{\theta_d} \mathcal{L}_{tr}^d, \theta_d)$;
    end for
    Update ω_g using Eq. (8) on validation data: $\omega_g \leftarrow \text{Adam}(\nabla_{\omega_g} \mathcal{L}_{val}^g, \omega_g)$;
    Update θ_g using Eq. (9) on training data: $\theta_g \leftarrow \text{Adam}(\nabla_{\theta_g} \mathcal{L}_{tr}^g, \theta_g)$
  end for
end while

Output: Optimal architecture weights ω_d, ω_g

3.4 Pruning, Training and Inference

Given an arbitrary dataset for ZSL, we first perform the procedures in Algorithm 1 to search for the optimal GAN

architecture. Once the training of architecture weights is complete, we can then derive the compact architecture by pruning the redundant paths. The pruning consists of two stages, namely edge pruning and operation pruning. In the first stage, for each intermediate node, we retain only two connections to its previous nodes; these connections are selected to be the edges with the top two highest weights. In the second stage, we prune the corresponding operations for each retained edge and choose the operation with the highest weight among all candidate operations. In this way, we can derive the final structure for both the generator and discriminator based on the learned edge and operation weights to form a complete GAN architecture.

To evaluate the final selected architectures, we employ the entire training set to train the architectures from scratch and then evaluate them on the test set. After training, we feed the unseen class embeddings and random noise into the generator so that it can be generalized to unseen classes. Given an unseen class y, the generator can synthesize an arbitrary number of CNN features $\tilde{\mathcal{X}} = \{\tilde{x}\}$ with $\tilde{x} = G(z, c_y; \theta_g^*)$ to solve the data scarcity problem (where θ_g^* denotes the trained generator parameters). Subsequently, the synthesized features can be exploited to train the standard softmax classifier with negative log likelihood loss, as follows:

$$\min_{\theta_c} -\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}\log\frac{\exp(\theta_c(y)^\top x_i)}{\sum_{y_j\in\mathcal{Y}}\exp(\theta_c(y_j)^\top x_i)}, \tag{10}$$

where $\mathcal{X} = \tilde{\mathcal{X}}^u,\ \mathcal{Y} = \mathcal{Y}^u$ for zero-shot learning and $\mathcal{X} = \tilde{\mathcal{X}}^u \cup \mathcal{X}^s,\ \mathcal{Y} = \mathcal{Y}^u \cup \mathcal{Y}^s$ for generalized zero-shot learning, while $\theta_c(y_j)$ refers to the training parameters with respect to class y_j. Once the classifier has been trained, we can utilize it for the prediction of new samples:

$$y^* = \arg\max_y \frac{\exp(\theta_c^*(y)^\top x)}{\sum_{y_j\in\mathcal{Y}}\exp(\theta_c^*(y_j)^\top x)}, \tag{11}$$

where x represents the visual feature of a new sample that has never been used during the training stage, while y^* is the predicted label for x.

4 Experiments

Dataset Description. We evaluate the effectiveness of the proposed method over four popular ZSL/GZSL datasets, including Caltech-UCSD Birds-200-2011 (CUB) [34], Oxford Flowers (FLO) [35], SUN attributes (SUN) [36] and Animals with Attributes (AWA) [37]. For fair comparison, we follow the zero-shot splits proposed by [4] to split each dataset into training and testing subsets, without overlapping categories between them. For class embedding, we adopt the per-class attributes for CUB (312-dim), SUN (102-dim), AWA (85-dim) and 1024-dim CNN-RNN features [38] for FLO. The real CNN features for each dataset are extracted by 101-layer ResNet [39] pre-trained on ImageNet 1K [40].

Implementation Details. We construct both the directed acyclic graph search space $\mathcal{G}_g = (\mathcal{V}_g, \mathcal{E}_g)$ for the generator and $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ for the discriminator using eight nodes: three input nodes, four intermediate nodes and one output node. Taking two types of general input information for the generator into account, we set the input of $\mathcal{G}_g$ as follows: 1) the Gaussian noise z; 2) the semantic embedding c_y; 3) the concatenation of z and c_y, which is used to investigate the stage at which information fusion should be conducted. Similarly, the input of $\mathcal{G}_d$ is composed of the following: 1) the real feature x or synthesized feature $\tilde{x}$; 2) the semantic embedding c_y; 3) the concatenation of x and c_y, or $\tilde{x}$ and c_y. For the output node, we set the output dimensions of $\mathcal{G}_g$ and $\mathcal{G}_d$ to 2048 and 1 respectively for feature generation and discrimination. Naturally, we can derive the dimensions of intermediate nodes as $[128, 256, 512, 1024]$ for $\mathcal{G}_g$ and $[1024, 512, 256, 128]$ for $\mathcal{G}_d$ respectively. For discriminator searching, we randomly split the entire training data $\mathcal{D}^{tr}$ into training set $\mathcal{D}^{tr}$ and validation set $\mathcal{D}^{val}$ with an equal number of samples. For the generator, both the training and validation data are sampled from the Gaussian noise distribution p_z. As suggested in [4], we set $\gamma = 10$, $\lambda = 0.01$, $n_b = 64$ and $n_d = 5$ respectively.

Competitors and Evaluation Metrics. In addition to f-CLSWGAN, we also compare the proposed ZeroNAS with other eight state-of-the-art generative approaches for ZSL/GZSL, including AFC-GAN [8], LisGAN [7], GAZSL [19], ZSL-ABP [20], GZSL-OD [41], f-VAEGAN-D2 [42], Meta-GZSL [43] and TF-VAEGAN [44]. We follow the widely adopted evaluation protocol proposed in [45] to evaluate the performance of each model.

4.1 Searched GAN Architectures

The discovered generator and discriminator pairs for each dataset are presented in Figure 3. From these results, we can make the following observations:

- Owing to the network pruning stage, ZeroNAS allows the discovered discrete architectures to have flexible hidden-layer numbers and dimensions for different datasets. As depicted in Figure 3, the CUB generator includes all four intermediate nodes, while the FLO generator connects the input and output directly without any intermediate nodes.
- It is also worth noting that all of the discovered architectures involve the input node with concatenated information, which indicates that the early fusion of information at the input layer is more beneficial to our task.
- Surprisingly, none of the discovered architectures contain batch normalization or dropout operations. This phenomenon suggests that simply stacking fully connected layers equipped with non-linear activations will ensure that the requirements of our task are satisfied.
- Except for the FLO generator without any intermediate nodes, all of the searched generators for CUB, SUN and AWA have an output layer with ReLU activation. It seems that this kind of structure is better able to learn the top max-pooling units of ResNet-101, as mentioned in f-CLSWGAN.

4.2 Architecture Evaluation

To evaluate the discovered architectures, we discard the weights learned during the searching process and train their parameters from scratch over the entire training set. Once trained, we report the ZSL/GZSL performance on the test

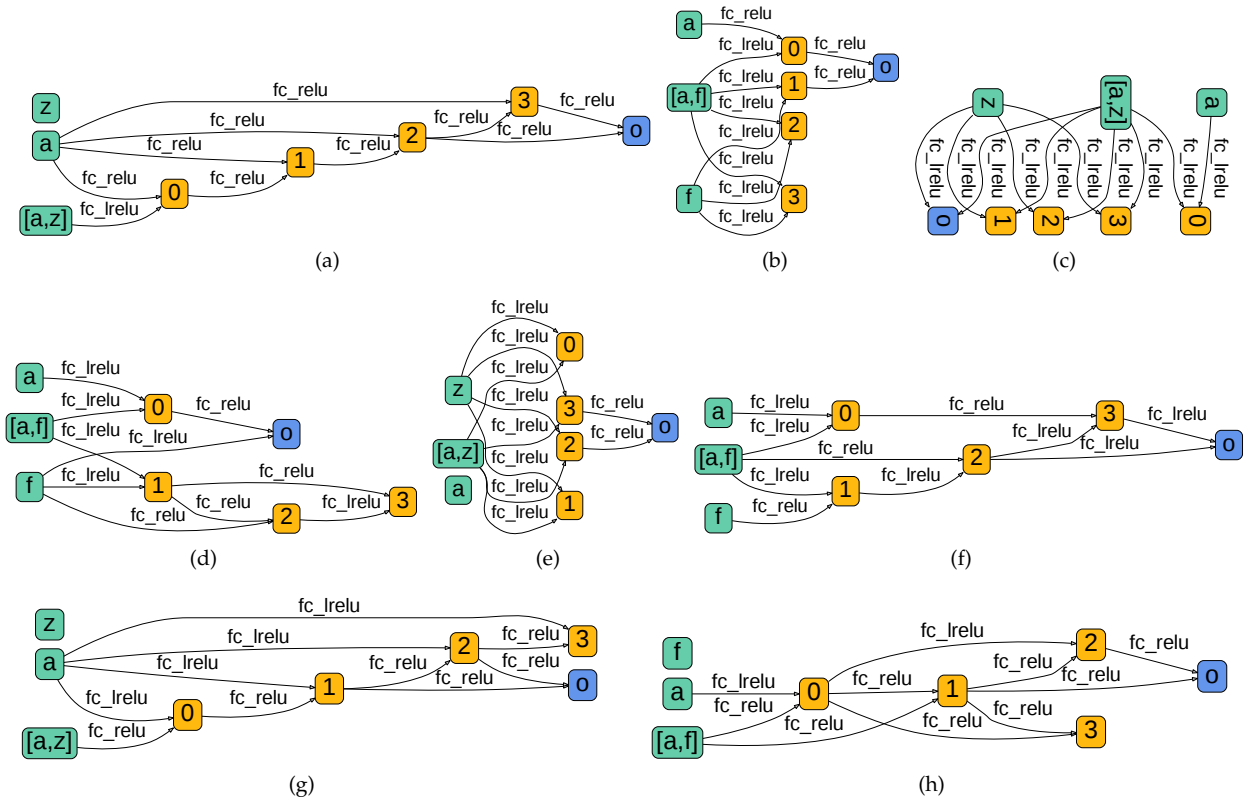


Fig. 3. The generator and discriminator architecture pairs learned for (a) (b) CUB, (c) (d) FLO, (e) (f) AWA, and (g) (h) SUN; here, "a", "z" and "f" represent the attribute, noise and feature vector respectively, while "'o" denotes the output node.

TABLE 1
ZSL performance (top-1 accuracy %) over benchmark datasets, where "FIX" and "NAS" refer to the model equipped with hand-crafted and searched GAN architectures respectively.

Methods	Arch	CUB	FLO	SUN	AWA
f-CLSWGAN	FIX	57.3	66.0	58.9	67.8
	NAS	$60.2^{+2.9}$	$69.0^{+3.0}$	$61.4^{+2.5}$	$71.3^{+3.5}$
LisGAN	FIX	57.9	68.5	60.8	70.1
	NAS	$62.6^{+4.7}$	$\mathbf{71.3}^{+2.8}$	$64.1^{+3.3}$	$72.8^{+2.7}$
GAZSL	FIX	55.7	60.6	60.7	65.2
	NAS	$58.0^{+2.3}$	$64.7^{+4.1}$	$63.4^{+2.7}$	$69.1^{+3.9}$
GZSL-OD	FIX	58.7	63.7	59.1	59.4
	NAS	$61.8^{+3.1}$	$67.2^{+3.5}$	$61.3^{+2.2}$	$65.6^{+6.2}$
ZSL-ABP	FIX	55.8	57.9	59.2	70.5
	NAS	$59.5^{+3.7}$	$63.2^{+5.3}$	$62.4^{+3.2}$	$73.6^{+3.1}$
AFC-GAN	FIX	62.0	66.8	61.8	70.4
	NAS	$64.5^{+2.5}$	$70.1^{+3.3}$	$63.7^{+1.9}$	$72.7^{+2.3}$
f-VAEGAN-D2	FIX	61.3	67.7	64.5	70.6
	NAS	$63.8^{+2.5}$	$70.5^{+2.8}$	$66.4^{+1.9}$	$72.6^{+2.0}$
Meta-GZSL	FIX	68.8	-	60.1	73.6
	NAS	$\mathbf{71.0}^{+2.2}$	-	$64.5^{+4.4}$	$\mathbf{75.4}^{+1.8}$
TF-VAEGAN	FIX	64.9	69.3	65.4	71.3
	NAS	$66.4^{+1.5}$	$71.1^{+1.8}$	$\mathbf{68.3}^{+2.9}$	$73.2^{+1.9}$

set that has never been used for architecture searching and training.

4.2.1 *Quantitative Analysis.*

The results of conventional ZSL are reported in Table 1. Compared to f-CLSWGAN with hand-crafted architecture, our method achieves up to 2.9%, 3.0%, 2.5%, and 3.5% improvements on CUB, FLO, SUN and AWA respectively. Apart from f-CLSWGAN, several improved GAN approaches (*e.g.*, AFC-GAN and LisGAN), have been proposed to further enhance the ZSL performance by developing a more complex model framework or leveraging external knowledge. For example, LisGAN improves the ZSL accuracy of f-CLSWGAN by 0.6%, 2.5%, 1.9%, and 2.3% over CUB, FLO, SUN and AWA respectively. Notably, when f-CLSWGAN is equipped with the GAN architectures discovered by ZeroNAS, it successfully achieves comparable or better performance compared with these state-of-the-art methods, *e.g.*, LisGAN and AFC-GAN, even without introducing any external knowledge or requiring additional auxiliaries. We can attribute this improvement to the automatic architecture search of the generator and discriminator pair, which can better capture the real data distribution. To explore the versatility of ZeroNAS, we further integrate it with other GAN based feature generation competitors. Instead of directly using the GAN architectures searched for f-CLSWGAN, we re-perform an architecture search for each method using the specific model framework and loss formulation. It is noteworthy from Table 1 that all of these methods have achieved varying degrees of performance improvement after adopting the re-searched architectures, leading to more advanced and competitive results. This phenomenon suggests that ZeroNAS could provide a generalized and promising way to enhance the performance of existing GAN-based ZSL methods.

We further report the experimental results of generalized zero-shot learning (GZSL) in Table 2. While ZSL predicts the possible categories of unseen samples from $\mathcal{Y}^u$ only, GZSL recognizes both seen and unseen samples by searching the

TABLE 2
GZSL performance (top-1 accuracy %) over benchmark datasets achieved by different methods. The results of seen, unseen classes and their harmonic mean are denoted as A_s, A_u and A_h respectively.

Methods	Arch	CUB			FLO			SUN			AWA		
		A_s	A_u	A_h	A_s	A_u	A_h	A_s	A_u	A_h	A_s	A_u	A_h
f-CLSWGAN	FIX	46.0	54.7	50.0	57.9	74.9	65.3	43.8	35.6	39.3	56.1	65.7	60.5
	NAS	48.8	57.8	$52.9^{+2.9}$	61.9	77.1	$68.7^{+3.4}$	45.8	36.4	$40.6^{+1.3}$	60.8	63.4	$62.1^{+1.6}$
LisGAN	FIX	44.9	59.3	51.1	55.8	80.5	65.9	44.6	35.5	39.6	52.6	76.3	62.3
	NAS	49.5	60.2	$54.3^{+3.2}$	61.1	83.7	$70.6^{+4.7}$	45.5	38.3	$41.6^{+2.0}$	65.2	67.0	$66.1^{+3.8}$
GAZSL	FIX	63.2	22.7	33.4	81.9	30.1	44.0	38.4	21.2	27.3	80.0	22.5	35.1
	NAS	62.3	31.5	$41.8^{+8.4}$	78.6	37.1	$50.4^{+6.4}$	40.4	27.5	$32.7^{+5.4}$	78.3	29.6	$43.0^{+7.9}$
GZSL-OD	FIX	46.2	35.8	40.3	57.8	55.2	56.4	32.5	26.6	29.3	47.9	69.1	56.6
	NAS	47.7	40.1	$43.6^{+3.3}$	60.5	62.3	$61.4^{+5.0}$	34.4	32.1	$33.2^{+3.9}$	52.1	74.7	$61.4^{+4.8}$
ZSL-ABP	FIX	45.6	54.1	49.5	50.5	77.0	61.0	41.5	36.7	38.9	69.1	55.7	61.7
	NAS	46.3	57.2	$51.2^{+1.7}$	53.8	80.3	$64.4^{+3.4}$	45.6	38.2	$41.6^{+2.7}$	71.6	57.4	$63.7^{+2.0}$
AFC-GAN	FIX	58.0	54.0	55.9	74.8	49.7	59.7	39.4	44.2	41.7	69.5	59.8	64.3
	NAS	59.5	57.3	$58.4^{+2.5}$	77.6	53.5	$63.3^{+3.6}$	39.9	48.6	$43.8^{+2.1}$	73.4	61.1	$66.7^{+2.4}$
f-VAEGAN-D2	FIX	60.3	48.0	53.5	75.5	57.2	65.1	37.6	45.3	41.1	71.3	57.2	63.5
	NAS	63.7	49.4	$55.6^{+2.1}$	78.1	60.8	$68.4^{+3.3}$	40.7	47.2	$43.7^{+2.6}$	71.8	60.5	$65.7^{+2.2}$
Meta-GZSL	FIX	51.6	61.9	56.3	-	-	-	-	-	-	73.8	58.4	65.2
	NAS	53.1	64.5	$58.2^{+1.9}$	-	-	-	-	-	-	76.5	61.3	$\mathbf{68.1}^{+2.9}$
TF-VAEGAN	FIX	64.5	52.8	58.1	83.2	62.1	71.1	40.3	45.5	42.7	74.5	59.1	65.9
	NAS	63.8	56.0	$\mathbf{59.6}^{+1.5}$	84.6	65.5	$\mathbf{73.8}^{+2.7}$	41.8	47.1	$\mathbf{44.3}^{+1.6}$	75.3	61.4	$67.6^{+1.7}$

possible categories from $\mathcal{Y}^s \cup \mathcal{Y}^u$. We test the performance of all comparison methods on both seen and unseen classes. Taking their harmonic mean as the overall performance, we can draw similar conclusions from Table 2 for GZSL as for conventional ZSL. In addition, it is worth noting that our method is able to boost the performance on unseen categories while simultaneously maintaining the accuracy on seen categories. Taking GAZSL as an example, the performance gain on unseen categories is much more significant than that on seen categories: *i.e.*, $+8.8\%$ vs -0.9%, $+7.0\%$ vs -3.3%, $+6.3\%$ vs $+2.0\%$, $+7.1\%$ vs -1.7% over CUB, FLO, SUN and AWA respectively. Thus, benefiting from the architectures discovered by ZeroNAS, the unbalance of A_s and A_u in GAZSL is dramatically alleviated; this is more likely to result in higher performance for A_h.

4.2.2 Qualitative Analysis.

To verify whether the learned architectures are indeed able to capture the data distribution of unseen classes, we utilize t-SNE [46] to visualize the synthesized features along with some real features of the AWA dataset in Figure 4. Specifically, we generate 100 samples for each unseen class, meaning that 100 real features are also randomly sampled for each unseen class. All these features are preprocessed with t-SNE from 2048-dim to 2-dim for visualization purposes. Compared to f-CLSWGAN, as expected, the unseen features synthesized by the proposed ZeroNAS demonstrate larger overlap with the real ones for most of categories, *e.g.*, *bat*, *rat* and *blue whale*, as depicted in Figure 4. This suggests that ZeroNAS can produce effective architectures to better capture the unseen data distribution, which further substantiates our improvements in Table 1 and 2.

4.3 Ablation Studies

4.3.1 Effect of generator and discriminator search.

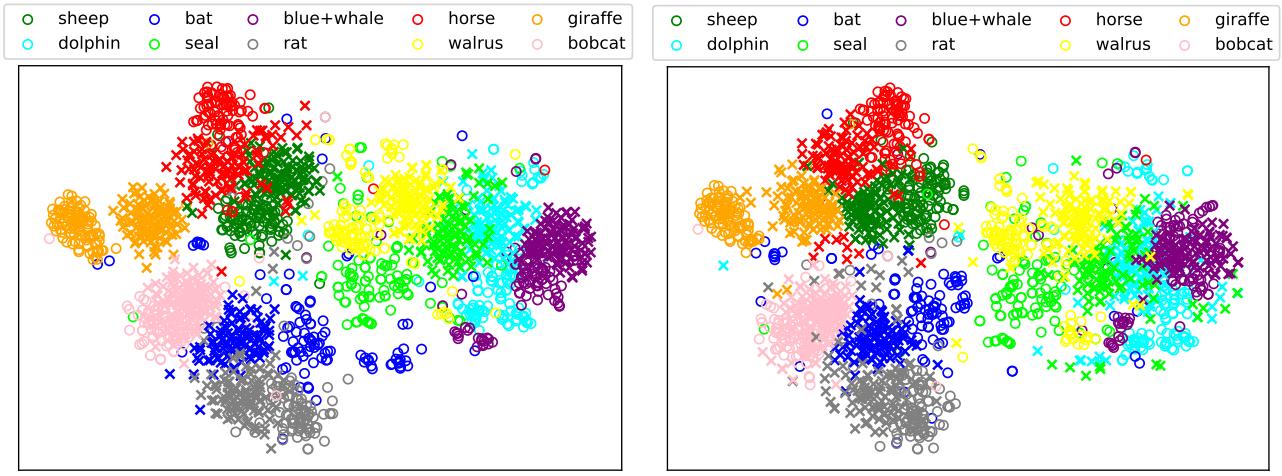
Table 3 examines the specific effect of each searching component, *i.e.*, G and D. We derive two variants of ZeroNAS, *i.e.*, "Fixed D" and "Fixed G", by fixing one of them to the original structure in f-CLSWGAN at a time. From Table 3, we can observe that the two variants achieve better results than the baseline f-CLSWGAN in most cases in terms of both ZSL and GZSL settings, which verifies the effectiveness of each component for our task. Moreover, by comparing the results in the two different settings, we can observe that the improvements achieved with "Fixed D" are more remarkable than those achieved with "Fixed G". This could be because the generator plays a more important role in capturing the true data distribution, and thus enables the model to learn higher-quality features. When compared with fixing one of G and D, our ZeroNAS takes advantage of both searched G and D to achieve the best ZSL/GZSL performance over all the datasets. This demonstrates that the searched generator and discriminator are complementary to each other in facilitating the ZSL/GZSL performance.

TABLE 3
ZSL/GZSL performance (top-1 accuracy %) with one of G and D fixed.

Dataset	Arch	ZSL	GZSL		
		A	A_s	A_u	A_h
CUB	Fixed D	59.2	45.9	57.8	51.1
	Fixed G	57.4	44.0	57.2	49.7
FLO	Fixed D	67.1	61.5	77.0	68.4
	Fixed G	64.7	56.9	77.9	65.8
SUN	Fixed D	60.8	47.1	34.8	40.0
	Fixed G	60.2	43.3	36.9	39.8
AWA	Fixed D	69.1	56.6	65.7	60.8
	Fixed G	68.6	57.8	64.1	60.8

4.3.2 Transferability.

To evaluate the transferability of the discovered architectures, we directly apply the GAN architecture searched over one dataset to the other datasets. The quantitative results are presented in Figure 5. We can easily observe that the transfered architectures suffer from significant performance degradation compared to their own architecture searched by ZeroNAS; in fact, some of the results are even worse than the baseline performance achieved by f-CLSWGAN. Taking FLO as an example, the architectures transfered from the other three datasets decrease the performance of



(a) f-CLSWGAN (b) ZeroNAS

Fig. 4. Visualization with t-SNE of the features synthesized by different methods on AWA, where ○ and × denote real and fake samples respectively.

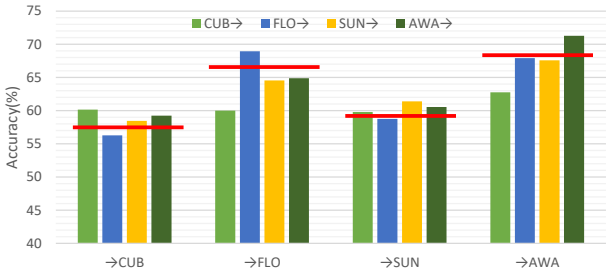


Fig. 5. Examination of architecture transferability in terms of ZSL accuracy, where the red line indicates the baseline performance achieved by f-CLSWGAN.

f-CLSWGAN from 66.04% to 60.01%, 64.56% and 64.90% respectively. This phenomenon indicates that the learned GAN architectures by ZeroNAS are data-dependent, and thus, that directly transferring the architectures searched for a specific dataset to other datasets is far from optimal.

4.3.3 *Accuracy during searching and training.*

Given that GAN is hard to train, we report the accuracy during both the searching and training processes over each dataset in Figure 6. When examining these results, it is worth noting that ZeroNAS always achieves better performance than f-CLSWGAN, while enjoying comparable convergence speed. The architecture search stage, which is a step that typically takes up a significant amount of time in previous NAS, only takes slightly more time to converge when compared with model training in our method. To be more specific, owing to the compact search space and differentiable search strategy, the proposed ZeroNAS only takes about 1.5 and 2 GPU hours to find the optimal GAN architecture for the FLO and CUB datasets respectively, and can thus be deemed very efficient. Moreover, we also observe that the searching stage always gains lower accuracy than the training stage. We believe this is because of the dynamically changing model architecture and incomplete training data for parameter optimization during the searching process.

5 Conclusion

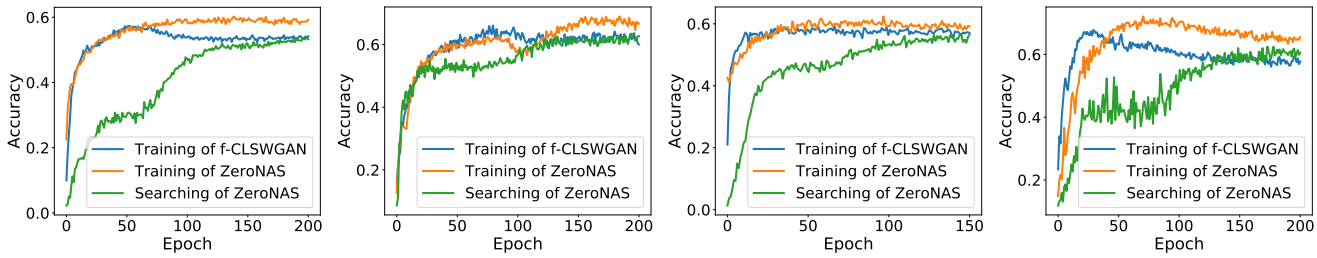
In this paper, we have presented ZeroNAS with the goal of making the first attempt to search the optimal GAN architecture tailored for zero-shot learning. Through analysis of the formulations of existing GANs for ZSL, we develop a model design that focuses on the following key points: 1) the relevance and balance between the generator and discriminator during searching; 2) expressive MLP search space for feature synthesis; 3) continuous relaxation of the search space for efficient differentiable search. Experiments on four benchmark datasets demonstrate that the proposed ZeroNAS is capable of discovering state-of-the-art GAN architectures that enhance the performance of existing GAN frameworks for ZSL/GZSL. Future work will concentrate on adapting the proposed architecture search method to other various kinds of architectures, such as cyclic consistency or reconstruction regularizer based GAN models and fully MLP-based models.

Acknowledgments

This work was supported by National Key Research and Development Program of China (No. 2018AAA0101400), National Nature Science Foundation of China (No. 62137002, No. 61872287, No. 62050194 and No. 61906109), Innovative Research Group of the National Natural Science Foundation of China (No. 61721002), Innovation Research Team of Ministry of Education (IRT_17R86), Project of China Knowledge Center for Engineering Science and Technology and The Consulting Research Project of Chinese Academy of Engineering "The Online and Offline Mixed Educational Service System for The Belt and Road" Training in MOOC China, and the Australian Research Council (ARC) under a Discovery Early Career Researcher Award (DECRA) No. DE190100626.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.



(a) CUB (b) FLO (c) SUN (d) AWA

Fig. 6. Top-1 accuracy on ZSL over benchmark datasets during the training of f-CLSWGAN, the searching of ZeroNAS and the training of ZeroNAS.

[2] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[3] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NIPS*, 2015.

[4] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *CVPR*, 2018.

[5] H. Zhang, Y. Long, L. Liu, and L. Shao, "Adversarial unseen visual feature synthesis for zero-shot learning," *Neurocomputing*, vol. 329, pp. 12–20, 2019.

[6] H. Huang, C. Wang, P. S. Yu, and C.-D. Wang, "Generative dual adversarial network for generalized zero-shot learning," in *CVPR*, 2019.

[7] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. Huang, "Leveraging the invariant side of generative zero-shot learning," in *CVPR*, 2019.

[8] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang, "Alleviating feature confusion for generative zero-shot learning," in *ACM MM*, 2019.

[9] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, and S. W. Su, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2921–2935, 2021.

[10] C. Li, G. Wang, B. Wang, X. Liang, Z. Li, and X. Chang, "Ds-net++: Dynamic weight slicing for efficient inference in cnns and transformers," *CoRR*, vol. abs/2109.10060, 2021.

[11] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *ICLR*, 2019.

[12] Y. Chen, T. Yang, X. Zhang, G. Meng, C. Pan, and J. Sun, "Detnas: Neural architecture search on object detection," in *NIPS*, 2019.

[13] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "Autogan: Neural architecture search for generative adversarial networks," in *ICCV*, 2019.

[14] H. Wang and J. Huan, "Agan: Towards automated design of generative adversarial networks," *arXiv preprint arXiv:1906.11080*, 2019.

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

[16] S. Doveh and R. Giryes, "Degas: Differentiable efficient generator search," *arXiv preprint arXiv:1912.00606*, 2019.

[17] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "Improving the improved training of wasserstein gans: A consistency term and its dual effect," in *ICLR*, 2018.

[18] R. Felix, V. B. Kumar, I. Reid, and G. Carneiro, "Multi-modal cycle-consistent generalized zero-shot learning," in *ECCV*, 2018.

[19] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal, "A generative adversarial approach for zero-shot learning from noisy texts," in *CVPR*, 2018.

[20] Y. Zhu, J. Xie, B. Liu, and A. Elgammal, "Learning feature-to-feature translator by alternating back-propagation for generative zero-shot learning," in *ICCV*, 2019.

[21] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[22] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 76:1–76:34, 2021.

[23] X. Cheng, Y. Zhong, M. Harandi, Y. Dai, X. Chang, H. Li, T. Drummond, and Z. Ge, "Hierarchical neural architecture search for deep stereo matching," in *NeurIPS*, 2020.

[24] M. Zhang, H. Li, S. Pan, X. Chang, Z. Ge, and S. Su, "Differentiable neural architecture search in equivalent space with exploration enhancement," in *NeurIPS*, 2020.

[25] M. Zhang, H. Li, S. Pan, X. Chang, and S. Su, "Overcoming multi-model forgetting in one-shot nas with diversity maximization," in *CVPR*, 2002.

[26] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019.

[27] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR*, 2017.

[28] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[29] L. Xie and A. Yuille, "Genetic cnn," in *ICCV*, 2017.

[30] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019.

[31] H. Jin, Q. Song, and X. Hu, "Auto-keras: Efficient neural architecture search with network morphism," *arXiv preprint arXiv:1806.10282*, 2018.

[32] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu, "Differentiable neural architecture search via proximal iterations," in *AAAI*, 2020.

[33] A. Hundt, V. Jain, and G. D. Hager, "Sharpdarts: Faster and more accurate differentiable architecture search," *arXiv preprint arXiv:1903.09900*, 2019.

[34] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[35] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICCVGI*, 2008.

[36] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *CVPR*, 2012.

[37] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 453–465, 2013.

[38] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *CVPR*, 2016.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[41] D. Mandal, S. Narayan, S. K. Dwivedi, V. Gupta, S. Ahmed, F. S. Khan, and L. Shao, "Out-of-distribution detection for generalized zero-shot action recognition," in *CVPR*, 2019.

[42] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "f-vaegan-d2: A feature generating framework for any-shot learning," in *CVPR*, 2019.

[43] V. K. Verma, D. Brahma, and P. Rai, "Meta-learning for generalized zero-shot learning," in *AAAI*, 2020.

[44] S. Narayan, A. Gupta, F. S. Khan, C. G. Snoek, and L. Shao, "Latent embedding feedback and discriminative features for zero-shot classification," in *ECCV*, 2020.

[45] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *CVPR*, 2017.

[46] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.