Evolving Semantic Prototype Improves Generative Zero-Shot Learning

Shiming Chen [1 2] **Wenjin Hou** [3] **Ziming Hong** [4] **Xiaohan Ding** [5] **Yibing Song** [6]
Xinge You [3] **Tongliang Liu** [2 4] **Kun Zhang** [1 2]

Abstract

In zero-shot learning (ZSL), generative methods synthesize class-related sample features based on predefined semantic prototypes. They advance the ZSL performance by synthesizing unseen class sample features for better training the classifier. We observe that each class's predefined semantic prototype (also referred to as semantic embedding or condition) does not accurately match its real semantic prototype. So the synthesized visual sample features do not faithfully represent the real sample features, limiting the classifier training and existing ZSL performance. In this paper, we formulate this mismatch phenomenon as the visual-semantic domain shift problem. We propose a dynamic semantic prototype evolving (DSP) method to align the empirically predefined semantic prototypes and the real prototypes for class-related feature synthesis. The alignment is learned by refining sample features and semantic prototypes in a unified framework and making the synthesized visual sample features approach real sample features. After alignment, synthesized sample features from unseen classes are closer to the real sample features and benefit DSP to improve existing generative ZSL methods by 8.5%, 8.0%, and 9.7% on the standard CUB, SUN AWA2 datasets, the significant performance improvement indicates that evolving semantic prototype explores a virgin field in ZSL.

[1]Carnegie Mellon University, Pittsburgh PA, USA [2]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi,UAE [3]Huazhong University of Science and Technology, Wuhan, China [4]University of Sydney, Sydey, Australia [5]Tencent AI LAB, Shenzhen, China [6]AI[3] Institute, Fudan University, China. Correspondence to: Xinge You <youxg@hust.edu.cn>.

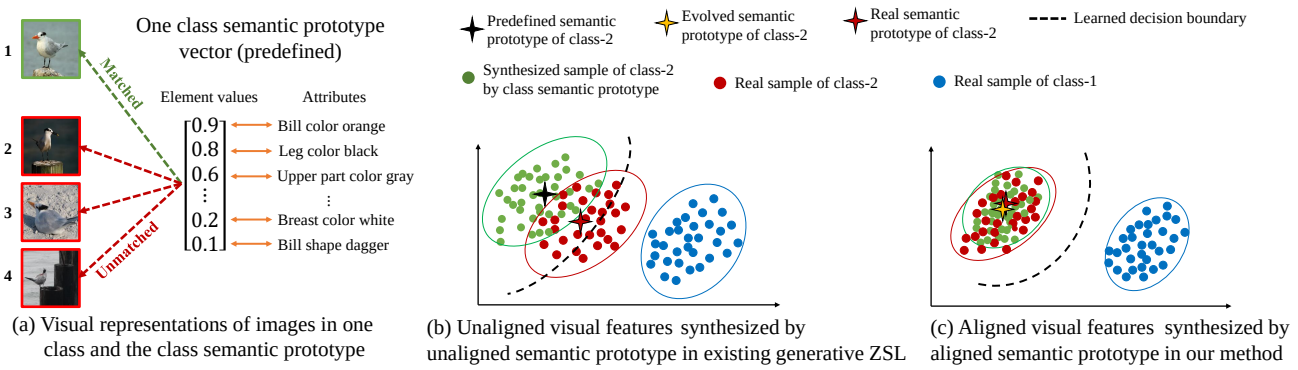
Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

Zero-shot learning recognizes unseen classes by learning their semantic knowledge, which is transferred from seen classes. Inspired by GAN (Goodfellow et al., 2014) and VAE (Kingma & Welling, 2014), the generative zero-shot learning methods (Arora et al., 2018) synthesize image sample features related to unseen classes based on the predefined class semantic prototypes (also referred to as semantic embedding in (Yan et al., 2021) and semantic condition in (Xian et al., 2019b)). The CNN features are typically utilized to synthesize unseen samples in generative ZSL. The synthesized sample features of unseen classes via generative ZSL benefit the classifier training process and improve the ZSL performance.

In generative ZSL, the unseen sample features for each class are synthesized by conditioning the class semantic prototype. As the class semantic prototype is predefined empirically, it is not able to accurately represent the real semantic prototype that is not available in practice. Fig. 1(a) shows four examples. The 'bill color orange' attribute in sample 2 does not accurately reflect the bird's bill color, which is mostly dark. Moreover, the attribute 'leg color black' in sample 3 does not accurately reflect the bird's leg, which is occluded by its body. A similar phenomenon appears in sample 4, where the attribute 'bill color orange' does not effectively describe the bird's bill that is mostly occluded as well. As such, the attributes with their element values defined in the semantic prototype do not properly represent image samples due to color variations or partial occlusions. For an unseen class, its semantic prototype and sample features differ from the real prototype and real sample features, respectively, as shown in Fig. 1(b). The feature discrepancy limits the classifier training and the generative ZSL performance. We formulate this phenomenon as a Visual-Semantic domain shift problem. To mitigate domain shift, we start by updating the semantic prototype and finally approaching the real one. Thus for an unseen class, sample features from the conditional generator will resemble the real features to benefit classifier training and improve the generative ZSL performance.

In this paper, we propose evolving the predefined semantic prototypes and aligning them with real ones. The align-



(a) Visual representations of images in one class and the class semantic prototype

(b) Unaligned visual features synthesized by unaligned semantic prototype in existing generative ZSL

(c) Aligned visual features synthesized by aligned semantic prototype in our method

Figure 1. Our motivation. For a specific class, generative ZSL methods synthesize sample features based on the predefined class semantic prototype vector. The misalignment between this prototype and the real one makes this prototype not always match the features of real visual samples, as shown in (a). The visual sample features synthesized by existing generative ZSL methods via this prototype do not faithfully represent the real features, as shown in (b). Our method evolves the predefined prototype by aligning it to the real one, improving the synthesized visual features close to the real visual features for learning a more discriminative classifier, as shown in (c).

ment is fulfilled in our dynamic semantic prototype evolving method that jointly refines prototypes and sample features. Through joint refinement, the conditional generator will synthesize visual sample features close to real ones for each class, as shown in Fig. 1(c). Specifically, our DSP consists of a visual→semantic mapping network (V2SM) and a visual-oriented semantic prototype evolving network (VOPE). The V2SM network takes sample features as input and outputs the class semantic prototypes. The VOPE network takes the semantic prototype at the current step as input and outputs the semantic prototype at the next step. During training, the generator produces sample features based on the semantic prototype at the current step. These features will be sent to the V2SM network to produce semantic prototypes. These prototypes will supervise VOPE output for prototype evolvement. The updated prototypes at each step will be sent back to the generator as a condition for visual sample feature synthesis. During inference, we concatenate the evolved semantic prototype vectors with visual sample features for semantic enhancement for classification. The visual features will be conditioned by the semantic prototype and mitigate the visual-semantic domain shift problem. The synthesized sample features resemble the real features based on the evolved prototype guidance and improve the classification performance.

We evaluate our DSP method on standard benchmark datasets, including CUB (Welinder et al., 2010), SUN (Patterson & Hays, 2012) and AWA2 (Xian et al., 2019a). The evaluation is conducted by measuring the performance improvement of state-of-the-art ZSL methods (i.e., CLSWGAN (Xian et al., 2018), f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020) and FREE (Chen et al., 2021a)) upon leveraging our DSP during training and inference. Experiments show that the average improvements of the harmonic mean over these methods are 8.5%, 8.0%, and 9.7% on CUB, SUN, and AWA2, respectively. The significant performance improvement, as well as the flexibility that DSP can be integrated into most generative ZSL methods, shows that evolving semantic prototype explores one promising virgin field in generative ZSL.

2. Related Works

Embedding-based Zero-Shot Learning. Embedding-based ZSL methods were popular early, they learn a visual→semantic projection on seen classes that is further transferred to unseen classes (Akata et al., 2016; 2015; Huynh & Elhamifar, 2020a; Chen et al., 2022a; Huynh & Elhamifar, 2020b), Considering the cross-dataset bias between ImageNet and ZSL benchmarks (Chen et al., 2021a), embedding-based methods were recently proposed to focus on learning the region-based visual features to enhance the holistic visual features[1] using attention mechanism (Yu et al., 2018; Zhu et al., 2019; Xie et al., 2019; Huynh & Elhamifar, 2020a; Xu et al., 2020; Chen et al., 2022b;d). However, these methods depends on the attribute features for attribute localization. As such, (Wang et al., 2021) introduces DPPN to iteratively enhance visual features using the category/visual prototype as supervision. Differently, we jointly and mutually refine the visual features and semantic prototype, enabling significant visual-semantic interactions in generative ZSL.

Generative Zero-Shot Learning. Since embedding-based methods learn the ZSL classifier only on seen classes, inevitably resulting in the models overfitting seen classes. To tackle this challenge, generative ZSL methods employ the

[1]Holistic visual features directly extract from a CNN backbone (*e.g.*, ResNet101 (He et al., 2016)) pretrained on ImageNet.

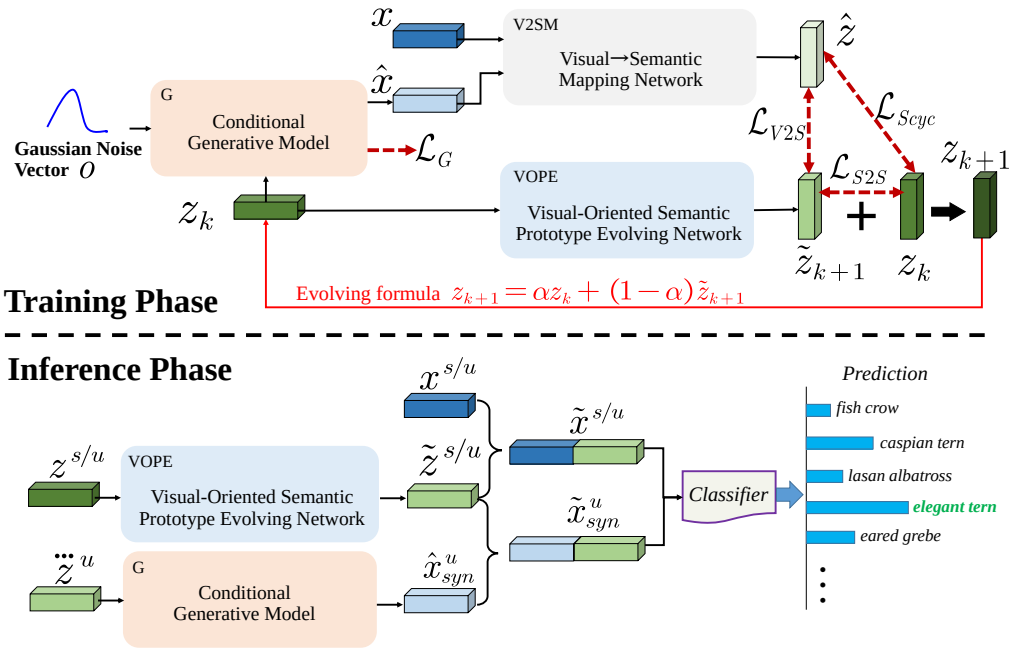


Figure 2. Dynamic semantic prototype evolvement (DSP). During training, we use V2SM and VOPE networks. V2SM maps sample features into semantic prototypes. VOPE maps dynamic semantic prototypes from the k-th step to the $(k+1)$-th step for evolvement. Based on the prototype z_k in the k-th step, we use the conditional generator to synthesize sample features $\hat{x}$ and map $\hat{x}$ to prototype $\hat{z}$ via V2SM. We use $\hat{z}$ to supervise VOPE output $\hat{z}_{k+1}$ during evolvement, which brings semantics from sample features. The V2SM and VOPE are jointly trained with the generative model. During inference, we use VOPE to map one input prototype $z^{s/u}$ to its evolved form $\tilde{z}^{s/u}$ for both seen and unseen classes. Besides, we take $\ddot{z}^u = \alpha \cdot z^u + (1-\alpha) \cdot \tilde{z}^u$ as the generator input to synthesize sample features $\hat{x}^u_{syn}$ for unseen classes. Then, we concatenate sample features with their $\tilde{z}^{s/u}$ for semantic enhancement during ZSL classification.

generative models (*e.g.*, VAE and GAN) to generate the unseen visual features for data augmentation (Arora et al., 2018; Xian et al., 2018; Shen et al., 2020; Hong et al., 2022), ZSL is then converted to a supervised classification task further. As such, the generative ZSL methods have shown significant performance and have become very popular recently. However, there are lots of challenges that need to be tackled in generative ZSL. To improve the optimization process of ZSL methods, (Skorokhodov & Elhoseiny, 2021) introduces class normalization for the ZSL task. (Çetin et al., 2022) introduces closed-form sample probing for generative ZSL. (Chou et al., 2021) designs a generative scheme to simultaneously generate virtual class labels and their visual features. To enhance the visual feature in generative ZSL methods, (Felix et al., 2018), (Narayan et al., 2020) and (Chen et al., 2021a) refine the holistic visual features by learning a visual→semantic projection, which encourages conditional generator to synthesize visual features with more semantic information. Orthogonal to these methods, we propose a novel dynamic semantic prototype learning method to tackle the visual-semantic domain shift problem and advance generative ZSL further.

Visual-Semantic Domain Shift and Projection Domain Shift. ZSL learns a projection between visual and semantic feature space on seen classes, and then the projection is transferred to the unseen classes for classification. However, data samples of the seen classes (source domain) and unseen classes (target domain) are disjoint, unrelated for some classes, and their distributions can be different, resulting in a large domain shift (Fu et al., 2015; Wan et al., 2019; Pourpanah et al., 2022). This challenge is the well-known problem of projection domain shift. To tackle this challenge, inductive-based methods incorporate additional constraints or information from the seen classes (Zhang et al., 2019; Jia et al., 2020; Wang et al., 2021). Besides that, several transductive-based methods have been developed to alleviate the projection domain shift problem (Fu et al., 2015; Xu et al., 2017; Wan et al., 2019). We should note that the visual-semantic domain shift is different from the projection domain shift. Visual-semantic domain shift denotes that the two domains (visual and semantic domains) share the same categories and only the joint distribution of input and output differs between the training and test stages. In contrast, the projection domain shift can be directly observed in terms of the projection shift, rather than the feature distribution shift (Pourpanah et al., 2022). As shown in Fig. 1, we show that visual-semantic domain shift is a bottleneck challenge in generative ZSL and is untouched in the literature.

3. Dynamic Semantic Prototype Evolvement

Fig. 2 shows the framework of our DSP during training and inference. We first revisit ZSL setting and illustrate the spirit of prototype evolvement. Then, we show the detailed framework design in the subsections.

ZSL setting: In ZSL, we have C^s seen classes and the total seen class data $\mathcal{D}^s = \{(x_i^s, y_i^s)\}$, where $x_i^s \in \mathcal{X}$ denotes the i-th sample feature (in 2048-dim extracted from a CNN backbone, e.g., ResNet101 (He et al., 2016)), and $y_i^s \in \mathcal{Y}^s$ is its class label. The $\mathcal{D}^s$ is split into a training set $\mathcal{D}_{tr}^s$ and a testing set $\mathcal{D}_{te}^s$ following (Xian et al., 2019a). On the other hand, we have C^u unseen classes with $\mathcal{D}_{te}^u = \{(x_i^u, y_i^u)\}$ data, where $x_i^u \in \mathcal{X}$ are the sample features of unseen classes, and $y_i^u \in \mathcal{Y}^u$ are its class label for evaluation. The semantic prototypes are represented by vectors. Each vector corresponds to one class. The total class number is $c \in \mathcal{C}^s \cup \mathcal{C}^u$. So there are c prototype vectors, and each vector is in the $|A|$ dimension where each dimension is a semantic attribute. Formally, the total semantic prototype vectors are denoted as $z^c = [z^c(1), \ldots, z^c(A)]^\top \in \mathbb{R}^{|A|}$. In the conventional ZSL setting (CZSL), we learn a classifier for unseen classes (i.e., $f_{\text{CZSL}} : \mathcal{X} \rightarrow \mathcal{Y}^U$). Differently in generalized ZSL (GZSL), we learn a classifier for both seen and unseen classes (i.e., $f_{\text{GZSL}} : \mathcal{X} \rightarrow \mathcal{Y}^U \cup \mathcal{Y}^S$).

Prototype evolvement spirit: In the training phase of DSP as shown in Fig. 2, we conduct prototype evolvement in VOPE that is supervised by V2SM. In the k-th step of evolvement, generator G synthesizes sample features $\hat{x}$ conditioned on the semantic prototype z_k. These features are sent to V2SM to produce a prototype $\hat{z}$, which supervises the VOPE output $\tilde{z}_{k+1}$. The z_k and $\tilde{z}_{k+1}$ are then linearly fused as z_{k+1} to finish the k-th step prototype evolvement, where semantic sample features $\hat{x}$ are integrated via V2SM mapping to align with the VOPE output $\tilde{z}_{k+1}$. In sum, a semantic prototype helps the generator to synthesize semantic sample features, and these features are mapped via V2SM to supervise VOPE output for prototype evolvement in VOPE. This evolvement mitigates the gap between sample features and semantic prototypes, and aligns the empirically predefined prototype to the real one that is unavailable in practice.

3.1. Generative ZSL Model

Generative ZSL design is to train a conditional generator G: $\mathcal{Z} \times \mathcal{O} \rightarrow \hat{\mathcal{X}}$. This generator takes semantic prototypes and Gaussian noise as inputs to synthesize class-specific sample features $\hat{x}$. We train this generator G together with V2SM and VOPE. Then, we use G to synthesize a large number of sample features for unseen classes based on semantic prototypes z^u. These synthesized sample features will be utilized to train a standard classifier (e.g., softmax). This classifier then predicts the testing data from unseen classes for ZSL performance evaluations.

3.2. Visual→Semantic Mapping Network (V2SM)

We design V2SM to map sample features to the semantic prototype. The sample features with semantics, synthesized by generator G conditioned on semantic prototypes, will be mapped via V2SM for prototype supervision. V2SM is a multilayer perceptron (MLP) with a residual block. The detailed network structure is in Appendix A. V2SM maps real or synthesized sample features (x or $\hat{x}$ in 2048-dim) to the semantic prototypes ($\hat{z}_{real}$ or $\hat{z}_{syn}$ in $|A|$-dim). The mapping can be written as:

$$\hat{z}_{real} = \text{V2SM}(x) \quad \text{or} \quad \hat{z}_{syn} = \text{V2SM}(\hat{x}) \tag{1}$$

where $\hat{z}_{real} \cup \hat{z}_{syn} = \hat{z}$, and $\hat{z} \in \mathbb{R}^{|A|}$. The conditional generator G (i.e., semantic→visual mapping) and V2SM conduct a semantic cycle network, i.e., $z \rightarrow G(o, z) = \hat{x}$ and $\hat{x} \rightarrow \text{V2SM}(\hat{x}) = \hat{z}$, where o is a random Gaussian noise vector with $|A|$-dim. As such, V2SM encourages G to synthesize sample features with semantics. We propose the semantic cycle-consistency loss to improve V2SM mapping, which can be written as:

$$\mathcal{L}_{Scyc} = \mathbb{E}\left[\|\hat{z}_{real} - z_k\|_1\right] + \mathbb{E}\left[\|\hat{z}_{syn} - z_k\|_1\right], \tag{2}$$

where $\mathbb{E}(\cdot)$ is the average value computed by using all the training samples, this operator is also utilized in the following equations, z_k is the dynamic semantic prototype from VOPE at the k-th step, which will be illustrated in Sec. 3.3. This loss is different from TF-VAEGAN (Narayan et al., 2020) and FREE (Chen et al., 2021a) where the predefined semantic prototypes are utilized as semantic supervision. Since the dynamic semantic prototype is closer to the real prototype, it supervises V2SM and G more accurately and thus further improves TF-VAEGAN and FREE.

3.3. Visual-Oriented Semantic Prototype Evolving Network (VOPE)

We propose VOPE to refine semantic prototypes under the supervision of $\hat{z}$. After refinement, the dynamic semantic prototype is closer to the real prototype, and the visual-semantic domain shift is alleviated as shown in Fig. 1(c). As the predefined semantic prototypes are empirically annotated, we refine them progressively via $\hat{z}$ that is mapped from sample features with semantics. The structure of VOPE is designed as an MLP network with a residual block. This block acts as a routing gate controlled by channel attention. The channel attention evolves the element value of attributes in semantic prototypes. The detailed network structure is in Appendix B. VOPE maps the semantic prototype z_k to another semantic prototype $\tilde{z}_{k+1}$ at the k-th step of evolvement. The mapping can be written as:

$$\tilde{z}_{k+1} = \text{VOPE}(z_k). \tag{3}$$

Note that initially (i.e., $k = 0$), z_k is the predefined semantic prototype that is empirically annotated. To enable $\tilde{z}_{k+1}$ evolved under the guidance from $\hat{z}$ that is mapped from semantic sample features, we propose a visual→semantic loss as follows:

$$\mathcal{L}_{V2S} = \mathbb{E}\left[1 - \text{Cosine}(\hat{z}, \tilde{z}_{k+1})\right] \tag{4}$$

where we use the cosine similarity as a weak constraint to conduct the semantic transfer from the visual feature domain to the prototype domain. Meanwhile, we propose the semantic reconstruction loss as follows:

$$\mathcal{L}_{S2S} = \mathbb{E}\left[\|\tilde{z}_{k+1} - z_k\|_1\right] \tag{5}$$

where $\tilde{z}_{k+1}$ is the VOPE output at the $(k+1)$-th step. The final dynamic semantic prototype at the $(k+1)$-th can be computed as:

$$z_{k+1} = \alpha \cdot z_k + (1 - \alpha) \cdot \tilde{z}_{k+1} \tag{6}$$

where α is a scale value and is set as 0.9 for a smooth moving average update. As a result, z_k is dynamically evolved and fed back to VOPE as input to start the $(k+1)$-th step.

3.4. Training and Inference

DSP objective loss function. We train V2SM and VOPE jointly with the generator G. The total loss function can be written as follows:

$$\mathcal{L}_{total} = \mathcal{L}_G + \lambda_{Scyc}\mathcal{L}_{Scyc} + \lambda_{V2S}\mathcal{L}_{V2S} + \lambda_{S2S}\mathcal{L}_{S2S} \tag{7}$$

where $\mathcal{L}_G$ is the loss of generative model G, λ_{Scyc}, λ_{V2S}, λ_{S2S} are the scale values controlling the influence of each loss term. We set λ_{Scyc} equal to λ_{S2S} as they both weigh semantic reconstruction loss. In our DSP, by using this loss function, we are effective in training various conditional generators G (i.e., CLSWGAN (Xian et al., 2018), f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020) and FREE (Chen et al., 2021a)). After training G, we are effective in synthesizing sample features that are closely related to unseen class prototypes for the classifier training. This classifier then makes well predictions to achieve superior performance on the ZSL benchmarks. In the following, we illustrate the inference phase where our DSP synthesizes features for training the classifier.

Sample features synthesis for unseen classes. After DSP training, we use VOPE to update the predefined semantic prototypes of unseen classes via Eq. 3 and Eq. 6. The prototype update can be written as follows:

$$\ddot{z}^u = \alpha z^u + (1 - \alpha)\text{VOPE}(z^u) \tag{8}$$

where $\ddot{z}^u$ is the updated semantic prototype of unseen classes. We use $\ddot{z}^u$ as the condition in G to synthesize sample features of unseen classes. The sample feature synthesis can be written as follows:

$$\hat{x}^u_{syn} = G(o, \ddot{z}^u), \tag{9}$$

where o is a random Gaussian noise vector in $|A|$-dim.

Sample features enhancement for all classes. We enhance sample features by concatenating them and the semantic prototypes. The enhanced features are more semantically related to the classes and alleviate the cross-dataset bias. For seen classes, we concatenate training data as $\tilde{x}^s_{tr} = [x^s_{tr}, \tilde{z}^s]$, and testing data as $\tilde{x}^s_{te} = [x^s_{te}, \tilde{z}^s]$, respectively For unseen classes, we concatenate training data as $\tilde{x}^u_{syn} = [\hat{x}^u_{syn}, \tilde{z}^u]$, and testing data as $\tilde{x}^u_{te} = [x^u_{te}, \tilde{z}^u]$, respectively. The x_{tr} is the real visual sample feature in the training set $\mathcal{D}^s_{tr}$, x_{te} is the real visual sample feature in the test sets $\mathcal{D}^s_{te}$ and $\mathcal{D}^u_{te}$. The $\tilde{z}^s \cup \tilde{z}^u = \tilde{z}$. These concatenated features, with semantic prototype conditions, will be utilized for the classifier training and prediction.

ZSL classifier training and prediction. The enhanced sample features $\tilde{x}^s_{tr}$ and $\tilde{x}^u_{syn}$ are utilized for the ZSL classifier training (e.g., softmax). We denote the classifier as f. For conventional ZSL, we can train the classifier as $f_{\text{CZSL}} : \tilde{\mathcal{X}} \rightarrow \mathcal{Y}^u$. For GZSL, we train the classifier as $f_{GZSL} : \tilde{\mathcal{X}} \rightarrow \mathcal{Y}^s \cup \mathcal{Y}^u$. After classifier training, we use $\tilde{x}^s_{te}$ and $\tilde{x}^u_{te}$ to evaluate the classifier predictions. Note that our DSP is inductive as there are no real visual sample features of unseen class for ZSL classifier training.

4. Experiments

In this section, we illustrate the experimental configurations, compare our DSP integrations with existing ZSL methods, and conduct ablation studies.

4.1. Experimental Configurations

Benchmark datasets. Our experiments are on four standard ZSL datasets including CUB (Welinder et al., 2010), SUN (Patterson & Hays, 2012), FLO (Nilsback & Zisserman, 2008)) and AWA2 (Xian et al., 2019a). In CUB, there are 11,788 images and 200 bird classes, where 150 are seen and 50 are unseen. The attributes (i.e., prototype dimension) are 312. In SUN, the image number, scene class number, seen/unseen class numbers, and attributes are 14,340, 717, 645/72, and 102, respectively. In FLO, these numbers are 8,189, 102, 82/20, and 1024, respectively. In AWA2, these numbers are 37,322, 50, 40/10, and 85, respectively. These configurations are the same for all the methods.

Implementation details. We extract image visual features in 2048-dim from the top-layer pooling units of a CNN ResNet-101 (He et al., 2016) encoder. The encoder is pretrained on the ImageNet dataset. In the ablation study, we

Table 1. State-of-the-art comparisons on CUB, SUN, AWA2, and FLO under GZSL settings. Embedding-based methods are categorized as ♣, and generative methods are categorized as ♠. The best and second-best results are marked in **Red** and **Blue**, respectively.

	Methods	Venue	CUB			SUN			AWA2			FLO		
			U	S	H	U	S	H	U	S	H	U	S	H
♣	SGMA (Zhu et al., 2019)	NeurIPS'19	36.7	71.3	48.5	–	–	–	37.6	87.1	52.5	–	–	–
	AREN (Xie et al., 2019)	CVPR'19	38.9	**78.7**	52.1	19.0	38.8	25.5	15.6	**92.9**	26.7	–	–	–
	CRnet (Zhang & Shi, 2019)	ICML'19	45.5	56.8	50.5	34.1	36.5	35.3	52.6	78.8	63.1	–	–	–
	APN (Xu et al., 2020)	NeurIPS'20	65.3	69.3	67.2	41.9	34.0	37.6	56.5	78.0	65.5	–	–	–
	DAZLE (Huynh & Elhamifar, 2020a)	CVPR'20	56.7	59.6	58.1	**52.3**	24.3	33.2	60.3	75.7	67.1	–	–	–
	CN (Skorokhodov & Elhoseiny, 2021)	ICLR'21	49.9	50.7	50.3	44.7	**41.6**	43.1	60.2	77.1	67.6	–	–	–
	TransZero (Chen et al., 2022a)	AAAI'22	**69.3**	68.3	**68.8**	52.6	33.4	40.8	61.3	82.3	70.2	–	–	–
	MSDN (Chen et al., 2022c)	CVPR'22	**68.7**	67.5	**68.1**	52.2	34.2	41.3	62.0	74.5	67.7	–	–	–
	I2DFormer (Naeem et al., 2022)	NeurIPS'22	35.3	57.6	43.8	–	–	–	**66.8**	76.8	71.5	35.8	**91.9**	51.5
♠	f-VAEGAN (Xian et al., 2019b)	CVPR'18	48.7	58.0	52.9	45.1	38.0	41.3	57.6	70.6	63.5	56.8	74.9	64.6
	TF-VAEGAN (Narayan et al., 2020)	CVPR'19	53.7	61.9	57.5	48.5	37.2	42.1	58.7	76.1	66.3	62.5	84.1	71.7
	LsrGAN (Vyas et al., 2020)	ECCV'20	48.1	59.1	53.0	44.8	37.7	40.9	54.6	74.6	63.0	–	–	–
	AGZSL (Chou et al., 2021)	ICLR'21	48.3	58.9	53.1	29.9	40.2	34.3	**65.1**	78.9	71.3	–	–	–
	FREE (Chen et al., 2021a)	ICCV'21	54.9	60.8	57.7	47.4	37.2	41.7	60.4	75.4	67.1	**67.4**	84.5	**75.0**
	GCM-CF (Yue et al., 2021)	CVPR'21	61.0	59.7	60.3	47.9	37.8	42.2	60.4	75.1	67.0	–	–	–
	HSVA (Chen et al., 2021b)	NeurIPS'21	52.7	58.3	55.3	48.6	39.0	**43.3**	59.3	76.6	66.8	–	–	–
	ICCE (Kong)	CVPR'22	67.3	65.5	66.4	–	–	–	65.3	82.3	**72.8**	66.1	86.5	74.9
	FREE+ESZSL (Çetin et al., 2022)	ICLR'22	51.6	60.4	55.7	48.2	36.5	41.5	51.3	78.0	61.8	65.6	82.2	72.9
	TF-VAEGAN+ESZSL (Çetin et al., 2022)	ICLR'22	51.1	63.3	56.6	44.0	39.7	41.7	55.2	74.7	63.5	63.5	83.2	72.1
	f-VAEGAN (Xian et al., 2019b) + **DSP**	Ours	62.5	**73.1**	67.4	**57.7**	**41.3**	**48.1**	63.7	**88.8**	**74.2**	**66.2**	**86.9**	**75.2**

use f-VAEGAN as a baseline and validate the effectiveness of our DSP. The training and validation splits are from (Xian et al., 2019a). We synthesize 150, 800, and 3400 sample features for each unseen class during classifier training for SUN, CUB, and AWA2 datasets, respectively. The λ_{Scyc} and λ_{S2S} are set with the same value, which are 0.1, 0.01, and 0.001 for CUB, SUN, and AWA2, respectively. The λ_{V2S} is 0.6, 0.6, and 1.0 for CUB, SUN, and AWA2, respectively. The settings on other baselines are in Appendix C.

Evaluation protocols. During testing (i.e., ZSL classification), we follow the evaluation protocols in (Xian et al., 2019a). In the CZSL setting, we calculate the top-1 accuracy of the unseen class, which is denoted as $\boldsymbol{acc}$. In the GZSL setting, we measure the top-1 accuracy on seen and unseen classes, denoted as $\boldsymbol{S}$ and $\boldsymbol{U}$, respectively. We also compute the harmonic mean of $\boldsymbol{S}$ and $\boldsymbol{U}$ to evaluate the GZSL performance. The harmonic mean can be computed as $\boldsymbol{H} = (2 \times \boldsymbol{S} \times \boldsymbol{U})/(\boldsymbol{S} + \boldsymbol{U})$.

4.2. Comparisons with State-of-the-art Methods

We integrate our DSP on f-VAEGAN, and compare it with state-of-the-art methods. Table 1 shows evaluation results. The comparing methods include non-generative methods (e.g., SGMA (Zhu et al., 2019), AREN (Xie et al., 2019), CRnet (Zhang & Shi, 2019), APN (Xu et al., 2020), DAZLE (Huynh & Elhamifar, 2020a), CN (Skorokhodov & Elhoseiny, 2021), TransZero (Chen et al., 2022a), MSDN (Chen et al., 2022c), and I2DFormer (Naeem et al., 2022)) and generative methods (e.g., f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020), AGZSL (Chou et al., 2021), GCM-CF (Chou et al., 2021), HSVA (Chen et al., 2021b), ICCE (Kong), and ES-ZSL (Çetin et al., 2022)). Compared to the generative methods, our f-VAEGAN+DSP setting achieves the best results on all datasets, (i.e., SUN ($\boldsymbol{H} = 48.1$), AWA2 ($\boldsymbol{H} = 74.2$) and FLO ($\boldsymbol{H} = 75.2$)). For instance, our f-VAEGAN+DSP outperforms the latest generative method (i.e., TF-VAEGAN+ESZSL (Çetin et al., 2022)) by a large margin, resulting in the improvements of harmonic mean by 10.8%, 6.4%, and 10.7% on CUB, SUN and AWA2, respectively. Moreover, our DSP effectively improves generative ZSL methods to achieve similar performance to the embedding-based methods. These results consistently demonstrate the superiority and great potential of our DSP in generative ZSL.

Table 2. Comparison with generative ZSL methods on the CUB, SUN, and AWA2 datasets under CZSL setting.

Methods	CUB	SUN	AWA2
	acc	acc	acc
CLSWGAN (Xian et al., 2018)	57.3	60.8	68.2
f-VAEGAN (Xian et al., 2019b)	61.0	64.7	71.1
CADA-VAE (Schönfeld et al., 2019)	59.8	61.7	63.0
Composer (Narayan et al., 2020)	**69.4**	62.6	71.5
FREE (Chen et al., 2021a)	64.8	65.0	68.9
HSVA (Chen et al., 2021b)	62.8	63.8	70.6
f-VAEGAN + DSP	62.8	**68.6**	**71.6**

Besides evaluating under the GZSL setting, we also compare our f-VAEGAN+DSP with the generative methods under the conventional ZSL setting (CZSL). Table 2 shows the evaluation results. Our f-VAEGAN+DSP achieves the accuracies of 62.8, 68.6, and 71.6 on CUB, SUN, and AWA2, respectively. These results are competitive compared to the generative ZSL methods under the CZSL setting. Table 2 indicates that our DSP is effective in mitigating the semantic-visual domain shift problem for generative ZSL methods in the CZSL setting.

Table 3. Ablation study on loss terms, smooth evolvement, and feature enhancement of our DSP. The baseline is f-VAEGAN.

Configurations	CUB			SUN		
	U	S	H	U	S	H
baseline	48.7	58.0	52.9	45.1	38.0	41.3
baseline+DSP (w/o $\mathcal{L}_{Scyc}$)	60.0	63.9	61.9	57.4	38.4	46.0
baseline+DSP (w/o $\mathcal{L}_{S2S}$)	61.9	69.6	65.5	58.1	37.2	45.4
baseline+DSP (w/o $\mathcal{L}_{V2S}$)	58.8	61.0	59.9	55.1	34.8	42.7
baseline+DSP (w/o smooth evolvement)	54.4	55.3	54.8	50.3	36.2	42.1
baseline+DSP (w/o enhancement)	52.4	53.9	53.1	54.2	35.0	42.5
baseline+DSP (full)	62.5	73.1	67.4	57.7	41.3	48.1

Table 4. Evaluation of DSP with multiple popular generative ZSL models on three benchmark datasets. Each row pair shows the effect of adding DSP to a particular generative ZSL model.

Generative ZSL Methods	CUB			SUN			AWA2		
	U	S	H	U	S	H	U	S	H
CLSWGAN (Xian et al., 2018)	43.7	57.7	49.7	42.6	36.6	39.4	57.9	61.4	59.6
CLSWGAN (Xian et al., 2018)+**DSP**	**51.4**	**63.8**	**56.9**$^{\uparrow 7.2}$	**48.3**	**43.0**	**45.5**$^{\uparrow 6.1}$	**60.0**	**86.0**	**70.7**$^{\uparrow 11.1}$
f-VAEGAN (Xian et al., 2019b)	48.7	58.0	52.9	45.1	38.0	41.3	57.6	70.6	63.5
f-VAEGAN (Xian et al., 2019b)+**DSP**	**62.5**	**73.1**	**67.4**$^{\uparrow 14.5}$	**57.7**	**41.3**	**48.1**$^{\uparrow 6.8}$	**63.7**	**88.8**	**74.2**$^{\uparrow 10.7}$
TF-VAEGAN (Narayan et al., 2020)	53.7	61.9	57.5	48.5	37.2	42.1	58.7	76.1	66.3
TF-VAEGAN (Narayan et al., 2020)+**DSP**	**58.7**	**67.4**	**62.8**$^{\uparrow 5.3}$	**60.3**	**45.3**	**51.7**$^{\uparrow 9.6}$	**65.6**	**87.1**	**74.8**$^{\uparrow 8.5}$
FREE (Chen et al., 2021a)	54.9	60.8	57.7	47.4	37.2	41.7	60.4	75.4	67.1
FREE (Chen et al., 2021a)+**DSP**	**60.9**	**68.7**	**64.6**$^{\uparrow 6.9}$	**60.3**	**44.1**	**51.0**$^{\uparrow 9.3}$	**65.3**	**89.2**	**75.4**$^{\uparrow 8.3}$

4.3. Ablation Study

We train our DSP via loss terms $\mathcal{L}_{Scyc}$, $\mathcal{L}_{V2S}$, and $\mathcal{L}_{S2S}$. The prototype evolves in a smooth manner, as shown in Eq. 6. Besides, feature enhancement via prototype concatenation is utilized for inference. We validate the effectiveness of loss terms, smooth evolvement, and feature enhancement by using the f-VAEGAN as the baseline. Table 3 shows the ablation results. When DSP is without $\mathcal{L}_{Scyc}$ during training, the performance degrades as shown in the second row (i.e., w/o $\mathcal{L}_{Scyc}$). This is because the generator lacks supervision from semantic cycle consistency. On the other hand, the results do not drop significantly without $\mathcal{L}_{S2S}$. However, if DSP does not contain $\mathcal{L}_{V2S}$, the performance drops significantly (i.e., the harmonic mean drops by 7.5% and 5.4% on CUB and SUN, respectively). These results show that the supervision from V2SM outputs (i.e., prototypes mapped from sample features) is effective for prototype evolvement in VOPE. The prototypes are gradually aligned to the real prototypes by matching the visual sample features via loss terms, which mitigates the visual-semantic domain shift problem in generative ZSL.

Besides loss terms, the dynamic prototype evolvement via smooth averaging is validated in our study. Without smooth evolvement (i.e., moving average in Eq. 6), the performance of our DSP decreases significantly (i.e., the harmonic mean drops by 12.6% and 6.0% on CUB and SUN, respectively, compared to the full DSP on the last row). In the inference phase, feature enhancement via prototype concatenation improves the DSP performance. The results in Table 3 show that our DSP method is effective in mitigating the visual-semantic domain shift problem and achieves favorable performance.

Generative ZSL methods with DSP. Our DSP is a general prototype evolvement method for generative ZSL. We experimentally analyze whether our DSP improves existing generative ZSL methods. We introduce four prevalent generative ZSL methods, including CLSWGAN (Xian et al., 2018), f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020), and FREE (Chen et al., 2021a). Their official implementations are utilized for producing results. We note that TF-VAEGAN and FREE contain the semantic decoder. So we only integrate VOPE with $\mathcal{L}_{V2S}$ and $\mathcal{L}_{S2S}$ for these two generative ZSL methods during the training phase. The detailed settings of our DSP integrated on these generative ZSL methods are in Sec. C. Table 4 shows the evaluation results. Our DSP consistently improves these generative ZSL methods on all benchmark datasets by a large margin. The average performance gains of harmonic mean are 8.5%, 8.0%, and 9.7% on CUB, SUN, and AWA2, respectively. These results show that visual-semantic domain shift is an important but unnoticed problem in existing generative ZSL methods. Our DSP evolving method is effective in improving the performance of existing generative ZSL methods.

Qualitative evaluation. We conduct a qualitative evaluation of the performance improvement of our DSP integrated into the baseline f-VAEGAN. The t-SNE visualization (Maaten & Hinton, 2008) of the real and synthesized sample fea-

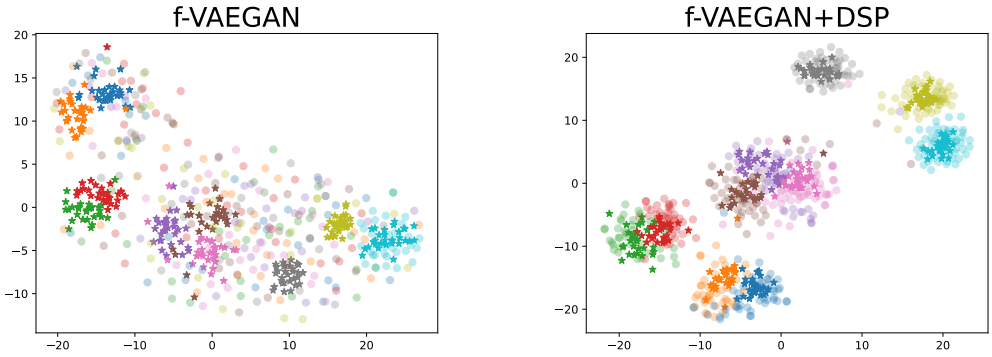
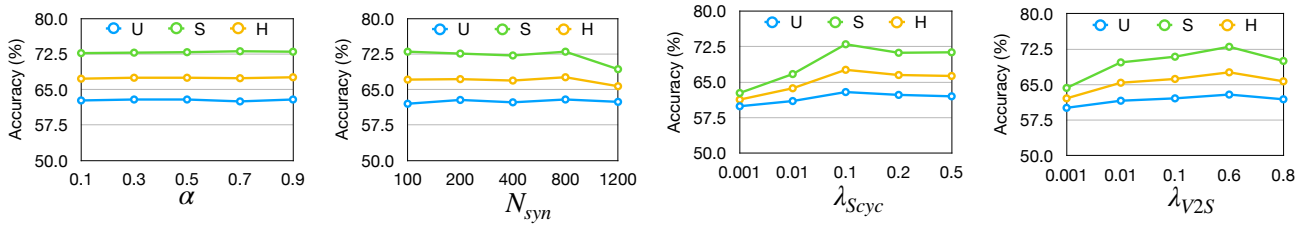


Figure 3. Qualitative evaluation with t-SNE visualization. The sample features from f-VAEGAN are shown on the left, and from f-VAEGAN with our DSP integration are shown on the right. We use 10 colors to denote randomly selected 10 classes from CUB. The ○ and ⋆ are denoted as the real and synthesized sample features, respectively. The synthesized sample features and the real features distribute differently on the left while distributing similarly on the right. (Best Viewed in Color)



(a) Varying α effect (b) Varying N_{syn} effect (c) Varying λ_{Scyc} effect (d) Varying λ_{V2S} effect

Figure 4. Hyper-parameter analysis. We show the GZSL performance variations on CUB by adjusting the value of α in (a), the value of N_{syn} in (b), the value of loss weight λ_{Scyc} in (c), and the value of loss weight λ_{V2S} in (d). (Best Viewed in Color)

tures is shown in Fig. 3. We randomly select 10 classes from CUB, and visualize these two types of sample features of f-VAEGAN on the left. Then, we visualize the same features of f-VAEGAN+DSP on the right. The left figure shows that sample features synthesized by f-VAEGAN and the real features distribute very differently, which brings the visual-semantic domain shift problem. In contrast, our DSP evolves the predefined semantic prototypes to match the real sample features. The prototype evolvement enables the generator to synthesize sample features close to the real features, as shown on the right figure. The visualization shows that our DSP helps the generative ZSL method to synthesize sample features close to the real sample features, which benefits classifier training and improves ZSL performance.

Hyper-parameter analysis. We analyze the effects of different hyper-parameters of our DSP under the CUB dataset. These hyper-parameters include the smooth fusion weight α in Eq. 6, the number of synthesized samples for each unseen class N_{syn}, and the loss weight λ_{Scyc} and λ_{V2S}. We set $\lambda_{Scyc} = \lambda_{S2S}$ as both of them are the semantic reconstruction loss, and we take one loss (i.e., λ_{Scyc}) for analysis here. Fig. 4 shows the ZSL performance of using different hyper-parameters. In (a), the results show that DSP is robust to varying values of α and achieves good performance when α is relatively large (i.e., $\alpha = 0.9$). This is because of the smooth prototype evolvement that does not change significantly for each step. In (b), our DSP is shown robust to N_{syn} when it is not set in a large number. The N_{syn} can be set as 800 to balance between the data amount and the ZSL performance. In (c) and (d), we conclude that we set λ_{Scyc} as 0.1, and set λ_{V2S} as 0.6 to achieve good performance. Moreover, we find that the value of λ_{V2S} should be larger than those of λ_{Scyc} and λ_{S2S}. This indicates that using V2SM output (i.e., prototypes mapped via semantic sample features) to supervise VOPE output is the crucial design to match the evolved prototypes and sample features. Under this supervision, the evolvement of semantic prototypes gradually matches the sample features synthesized by the generator G. Overall, Fig. 4 shows that our DSP is robust to overcome hyper-parameter variations.

5. Concluding Remarks

In this work, we analyze that the predefined semantic prototype introduces the visual-semantic domain shift problem in generative ZSL. We then propose a dynamic semantic prototype evolving method to mutually refine the prototypes and sample features. After evolvement, the generator syn-

thesizes sample features close to the real ones and benefits ZSL classifier training. Our DSP can be integrated into a series of generative ZSL methods for performance improvement. Experiments on the benchmark datasets indicate our effectiveness, which shows that evolving semantic prototype explores one promising virgin field in generative ZSL.

Acknowledgements

The work was supported in part by the NSF-Convergence Accelerator Track-D award #2134901, by the NSFC (62172177), by the National Key R&D Program (2022YFC3301004,2022YFC3301704), by the National Institutes of Health (NIH) under Contract R01HL159805, by grants from Apple Inc., KDDI Research, Quris AI, and IBT, and by generous gifts from Amazon, Microsoft Research, and Salesforce.

References

Akata, Z., Reed, S., Walter, D., Lee, H., and Schiele, B. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, pp. 2927–2936, 2015.

Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38: 1425–1438, 2016.

Arora, G., Verma, V., Mishra, A., and Rai, P. Generalized zero-shot learning via synthesized examples. In *CVPR*, pp. 4281–4289, 2018.

Chen, S., Wang, W., Xia, B., Peng, Q., You, X., Zheng, F., and Shao, L. Free: Feature refinement for generalized zero-shot learning. In *ICCV*, 2021a.

Chen, S., Xie, G.-S., Yang Liu, Y., Peng, Q., Sun, B., Li, H., You, X., and Shao, L. Hsva: Hierarchical semantic-visual adaptation for zero-shot learning. In *NeurIPS*, 2021b.

Chen, S., Hong, Z., Liu, Y., Xie, G.-S., Sun, B., Li, H., Peng, Q., Lu, K., and You, X. Transzero: Attribute-guided transformer for zero-shot learning. In *AAAI*, 2022a.

Chen, S., Hong, Z., Xie, G., Peng, Q., You, X., Ding, W., and Shao, L. Gndan: Graph navigated dual attention network for zero-shot learning. *IEEE transactions on neural networks and learning systems*, 2022b.

Chen, S., Hong, Z., Xie, G.-S., Yang, W., Peng, Q., Wang, K., Zhao, J., and You, X. Msdn: Mutually semantic distillation network for zero-shot learning. In *CVPR*, 2022c.

Chen, S., Hong, Z.-Q., Xie, G., Zhao, J., Li, H., You, X., Yan, S., and Shao, L. Transzero++: Cross attribute-guided transformer for zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 2022d.

Chou, Y.-Y., Lin, H.-T., and Liu, T.-L. Adaptive and generative zero-shot learning. In *ICLR*, 2021.

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

Fu, Y., Hospedales, T. M., Xiang, T., and Gong, S. Transductive multi-view zero-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:2332–2345, 2015.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Hong, Z., Chen, S., Xie, G., Yang, W., Zhao, J., Shao, Y., Peng, Q., and You, X. Semantic compression embedding for generative zero-shot learning. In *IJCAI*, 2022.

Huynh, D. and Elhamifar, E. Fine-grained generalized zero-shot learning via dense attribute-based attention. In *CVPR*, pp. 4482–4492, 2020a.

Huynh, D. T. and Elhamifar, E. Compositional zero-shot learning via fine-grained dense feature composition. In *NeurIPS*, 2020b.

Jia, Z., Zhang, Z., Wang, L., Shan, C., and Tan, T. Deep unbiased embedding transfer for zero-shot learning. *IEEE Transactions on Image Processing*, 29:1958–1971, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR*, 2014.

Kong, X. En-compactness: Self-distillation embedding & contrastive generation for generalized zero-shot learning.

Maaten, L. V. D. and Hinton, G. E. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Naeem, M. F., Xian, Y., Gool, L. V., and Tombari, F. I2dformer: Learning image to document attention for zero-shot image classification. In *NeurIPS*, 2022.

Narayan, S., Gupta, A., Khan, F., Snoek, C. G. M., and Shao, L. Latent embedding feedback and discriminative features for zero-shot classification. In *ECCV*, 2020.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *ICVGIP*, pp. 722–729, 2008.

Patterson, G. and Hays, J. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, pp. 2751–2758, 2012.

Pourpanah, F., Abdar, M., Luo, Y., Zhou, X., Wang, R., Lim, C. P., and Wang, X. A review of generalized zero-shot learning methods. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

Schönfeld, E., Ebrahimi, S., Sinha, S., Darrell, T., and Akata, Z. Generalized zero- and few-shot learning via aligned variational autoencoders. In *CVPR*, pp. 8239–8247, 2019.

Shen, Y., Qin, J., and Huang, L. Invertible zero-shot recognition flows. In *ECCV*, 2020.

Skorokhodov, I. and Elhoseiny, M. Class normalization for (continual)? generalized zero-shot learning. In *ICLR*, 2021.

Vyas, M. R., Venkateswara, H., and Panchanathan, S. Leveraging seen and unseen semantic relationships for generative zero-shot learning. In *ECCV*, 2020.

Wan, Z., Chen, D., Li, Y., Yan, X., Zhang, J., Yu, Y., and Liao, J. Transductive zero-shot learning with visual structure constraint. In *NeurIPS*, 2019.

Wang, C., Min, S., Chen, X., Sun, X., and Li, H. Dual progressive prototype network for generalized zero-shot learning. In *NeurIPS*, 2021.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S. J., and Perona, P. Caltech-ucsd birds 200. *Technical Report CNS-TR-2010-001, Caltech,*, 2010.

Xian, Y., Lorenz, T., Schiele, B., and Akata, Z. Feature generating networks for zero-shot learning. In *CVPR*, pp. 5542–5551, 2018.

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2251–2265, 2019a.

Xian, Y., Sharma, S., Schiele, B., and Akata, Z. F-vaegan-d2: A feature generating framework for any-shot learning. In *CVPR*, pp. 10267–10276, 2019b.

Xie, G.-S., Liu, L., Jin, X., Zhu, F., Zhang, Z., Qin, J., Yao, Y., and Shao, L. Attentive region embedding network for zero-shot learning. In *CVPR*, pp. 9376–9385, 2019.

Xu, W., Xian, Y., Wang, J., Schiele, B., and Akata, Z. Attribute prototype network for zero-shot learning. In *NeurIPS*, 2020.

Xu, X., Shen, F., Yang, Y., Zhang, D., Shen, H. T., and Song, J. Matrix tri-factorization with manifold regularizations for zero-shot learning. In *CVPR*, pp. 2007–2016, 2017.

Yan, C., Chang, X., Li, Z., Ge, Z., Guan, W., Zhu, L., and Zheng, Q. Zeronas: Differentiable generative adversarial networks search for zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Yu, Y., Ji, Z., Fu, Y., Guo, J., Pang, Y., and Zhang, Z. Stacked semantics-guided attention model for fine-grained zero-shot learning. In *NeurIPS*, 2018.

Yue, Z., Wang, T., Zhang, H., Sun, Q., and Hua, X. Counterfactual zero-shot and open-set visual recognition. In *CVPR*, 2021.

Zhang, F. and Shi, G. Co-representation network for generalized zero-shot learning. In *ICML*, 2019.

Zhang, H., Long, Y., Guan, Y., and Shao, L. Triple verification network for generalized zero-shot learning. *IEEE Transactions on Image Processing*, 28:506–517, 2019.

Zhu, Y., Xie, J., Tang, Z., Peng, X., and Elgammal, A. Semantic-guided multi-attention localization for zero-shot learning. In *NeurIPS*, 2019.

Çetin, S., Baran, O. B., and Cinbis, R. G. Closed-form sample probing for learning generative models in zero-shot learning. In *ICLR*, 2022.

APPENDIX

Appendix organization:

- Appendix A: Network details of V2SM.
- Appendix B: Network details of VOPE.
- Appendix C: The hyper-parameter settings of our DSP entailed on four baselines.

A. Nework Details of V2SM

As shown in Fig. 5, the network of V2SM is an MLP with a residual block, which avoids too much information loss from 2048-dim to $|A|$-dim.

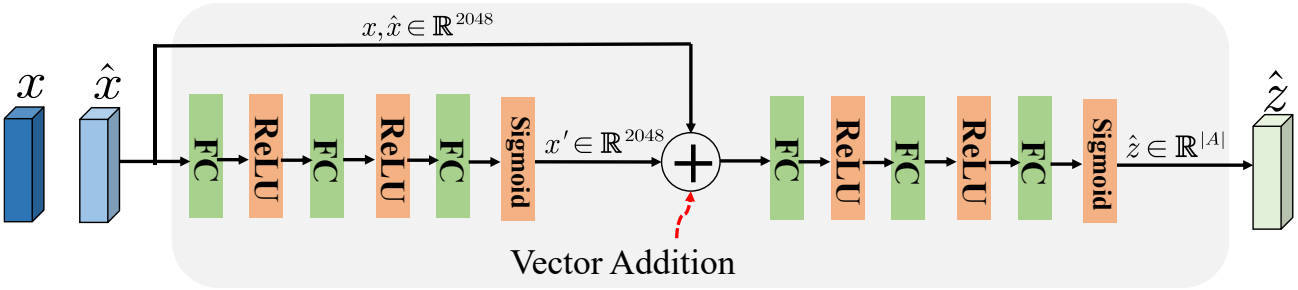


Figure 5. Network details of V2SM.

B. Nework Details of VOPE

As shown in Fig. 6, the network of VOPE is an MLP with a residual block (fusing with Hardamad Product), which preserves enough semantic information of the predefined semantic prototype and enables VOPE to evolve progressively.

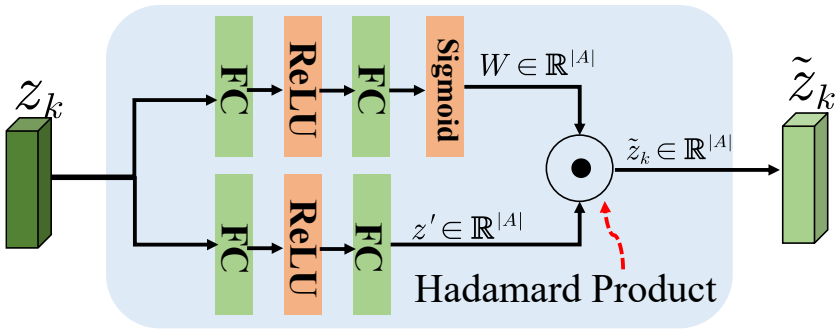


Figure 6. Network details of VOPE.

C. The Hyper-Parameter Settings of Our DSP Entailed on Various Baselines

We present the hyper-parameter settings of our DSP entailed on various baselines (*i.e.*, CLSWGAN (Xian et al., 2018), f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020) and FREE (Chen et al., 2021a)) on CUB, SUN and AWA2. Including the synthesizing number of per unseen classes N_{syn}, the loss weights λ_{Scyc}, λ_{V2S} and combination coefficient α in Eq. 6. We empirically observe that our DSP is robust and easy to train when it is entailed on various generative models. Based on these hyper-parameter settings, our DSP achieves significant performance gains over the various popular generative models on all datasets. For instance, the average performance gains of harmonic mean are 8.5%, 8.0% and 9.7% on CUB, SUN and AWA2, respectively. Please refer to Table 4.

Table 5. The hyper-parameter settings of our DSP entailed on various baselines (*i.e.*, CLSWGAN (Xian et al., 2018), f-VAEGAN (Xian et al., 2019b), TF-VAEGAN (Narayan et al., 2020) and FREE (Chen et al., 2021a)) on CUB, SUN and AWA2. Including the synthesizing number of per unseen classes N_{syn}, the loss weights λ_{Scyc}, λ_{V2S} and combination coefficient α in Eq. 6.

Methods	**CUB**				**SUN**				**AWA2**			
	N_{syn}	λ_{Scyc}	λ_{V2S}	α	N_{syn}	λ_{Scyc}	λ_{V2S}	α	N_{syn}	λ_{Scyc}	λ_{V2S}	α
clswGAN (Xian et al., 2018) + **DSP**	300	0.15	1.0	0.9	300	0.005	1.0	0.9	3400	0.1	1.0	0.9
f-VAEGAN (Xian et al., 2019b)+**DSP**	800	0.1	0.6	0.9	150	0.01	1.0	0.9	3400	0.001	0.6	0.9
TF-VAEGAN (Narayan et al., 2020)+ **DSP**	400	0.01	1.0	0.9	500	0.05	1.5	0.9	5300	0.09	1.4	0.9
FREE (Chen et al., 2021a) + **DSP**	600	0.1	0.6	0.9	150	0.01	1.0	0.9	4000	0.001	2.0	0.9