Distribution and Gradient Constrained Embedding Model for Zero-Shot Learning with Fewer Seen Samples

Jing Zhang[a], YangLi-ao Geng[a], Wen Wang[a], Wenju Sun[a], Zhirong Yang[b], Qingyong Li[a,*]

[a]*Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing, 100044, China*
[b]*Norwegian Open AI Lab, Norwegian University of Science and Technology, Trondheim, 7034, Norway*

Abstract

Zero-Shot Learning (ZSL), which aims to recognize unseen classes with no training data, has made great progress in recent years. However, established ZSL methods implicitly assumed that there exist sufficient labeled samples for each seen class, which is quite idealistic in general as collecting sufficient labeled samples is a labor-intensive task and may even be naturally impractical for some low-probability events. Accordingly, we investigate how to perform ZSL with fewer seen samples. Specifically, we propose a Distribution and Gradient constrained Embedding Model (DGEM), which aims to predict the visual prototypes (means) for the given semantic vectors of seen classes. Specifically, we summarize the main challenges brought by limited seen samples as the representation bias problem and the over-fitting problem. Correspondingly, two regularizers are proposed to solve them: (1) a proto-

*Corresponding author
Email addresses: j_zhang@bjtu.edu.cn (Jing Zhang), gengyla@bjtu.edu.cn (YangLi-ao Geng), wangwen@bjtu.edu.cn (Wen Wang), SunWenJu@bjtu.edu.cn (Wenju Sun), zhirong.yang@ntnu.no (Zhirong Yang), liqy@bjtu.edu.cn (Qingyong Li)

type refinement loss that uses the relative distribution of class semantics to constrain that of the predicted visual prototypes; (2) a projection smoothing constraint that prevents the model from forming sharp decision boundaries. We validate the effectiveness of DGEM on five ZSL datasets and compare it with several representative ZSL methods. Experimental results show that DGEM outperforms the other established methods when each seen class has only 1/5 sample(s).

Keywords:

Zero-shot learning, fewer seen samples, representation bias, over-fitting

---

1. Introduction

Given the description of an unseen animal, e.g., a liger, humans can readily imagine its approximate visual appearance by combining the elements from similar classes, e.g., tiger and lion. Inspired by this cognitive competence, researchers proposed the concept of **Zero-Shot Learning** (ZSL) [1, 2], which tries to identify unseen classes without labeled samples in the training stage. Specifically, to achieve the above goal, ZSL uses the semantic descriptions of classes, which can be attribute vectors [3], word2vec [4], and textual descriptions [5], as the bridge to transfer the knowledge learned from seen domain to unseen domain. To this end, ZSL can be boiled down to a task of learning the correspondence relationship between visual features and semantic descriptions.

There are two main paradigms in the current ZSL community: embedding methods and generative methods. Embedding methods try to solve ZSL by projecting visual features and semantic vectors into a shared space, followed

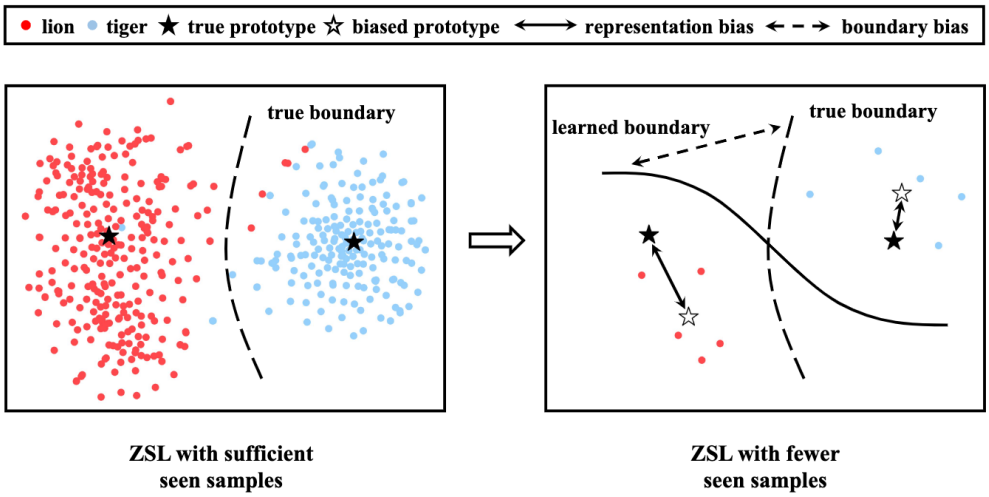


Figure 1: The diagrams of ZSL with sufficient seen samples (left) and fewer seen samples (right). We visualize the distribution of two seen classes ("lion" and "tiger") in AwA2 dataset [6] by t-SNE [7]. As can be seen, when each class only has 5 samples: (1) their visual prototypes (hollow stars) will deviate from the true expectations (solid stars); (2) the model will over fit to the available samples and thus learn an unsatisfactory boundary (solid line) which is far from the true one (dotted line).

by the nearest neighbor search algorithm to determine the final predictions. Generative methods employ generative adversarial network (GAN) or variational auto-encoder (VAE) to synthesize fake visual features for the input unseen semantic vectors, which convert ZSL into a supervised learning problem. Various ZSL methods in these two paradigms are proposed and have shown promising performance under general ZSL setting [8, 9]. However, we find that the performance of their models relies on an implicit assumption, that is, existing sufficient labeled seen samples to train their models. This assumption is quite idealistic in the practical scenarios because (1) collecting sufficient labeled samples for all seen classes is a pretty costly task; (2)

some objects or events may be naturally rare or hard to happen, such as endangered animals, extreme weather scenarios and rare diseases. In this paper, to go beyond the above limitations, we investigate how to perform ZSL with much fewer seen samples, which is more challenge than the general ZSL setting.

Figure 1 illustrates the diagrams of ZSL with sufficient seen samples (left) and fewer seen samples (right). As can be seen, the reduction of training (seen) data will bring new challenges and further increase the difficulty of ZSL. Specifically, fewer data mean less and incomplete information. If we use the visual prototype (mean of visual features) to represent each seen class in the visual space, the prototype calculated by limited samples (hollow start) has a high probability of deviating from the true expectation calculated by sufficient samples (solid star). As a result, the model will learn a biased projection between visual features and semantic descriptions, and suffer severe performance deterioration, which we called **the representation bias problem**. Besides, fewer data put higher demands on the generalization capability of models. With limited training samples, the model can easily fit them well, causing the learned boundaries to be far from the ideal boundaries, that is, **the over-fitting problem**.

To tackle the above problems, we propose a Distribution and Gradient constrained Embedding Model, referred to as DGEM. Specifically, given the semantic vectors of seen classes, we use an embedding model to predict their corresponding visual prototypes, supervised by a mean square error loss to be close to the visual means of seen classes. In addition, based on the assumption that the class distribution in the semantic space is similar to that

in the visual space, we specifically design a prototype refinement loss to alleviate the bias of visual prototypes. Furthermore, to improve the generalization ability of the model, we generate some interpolated semantic vectors by combining the real ones and then constrain their gradients to enforce the Lipschitz continuity of DGEM, which can prevent it from generating sharp decision boundaries. After training, we apply DGEM to the semantic vectors of unseen classes to predict the corresponding visual prototypes, which will be utilized as anchors to determine the final class labels of test samples. In brief, our contributions can be summarized as follows:

- For zero-shot learning with fewer seen samples, we summarize the main challenges caused by limited seen samples as the representation bias problem and the over-fitting problem, and propose a distribution and gradient constrained embedding model (DGEM) to solve them.

- For the representation bias problem, we design a prototype refinement loss that exploits the relative distribution information of classes in the semantic space to alleviate the bias of the predicted visual prototypes. For the over-fitting problem, we introduce a projection smoothing loss to enforce the Lipschitz continuity of the model by constraining the gradients with respect to some interpolated semantic vectors.

- We evaluate the performance of DGEM on five commonly used datasets in which each seen class only has 1/5 sample(s). The experimental results demonstrate that DGEM outperforms the other methods for ZSL with fewer seen samples.

2. Related work

The target of ZSL is to train a model which can generalize to unseen classes with no training data and only a semantic description [1]. The premise of achieving this target is that seen classes and unseen classes share a common semantic space. Thus the projection between visual features and semantic descriptions learned from the seen domain can be transferred to the unseen domain. Briefly, according to how to use the learned projection, the existing ZSL methods can be roughly divided into embedding methods and generative methods.

Embedding methods use the learned projection to map visual features and class semantic vectors to a shared feature space, followed by the nearest neighbor search algorithm to implement classification. For example, Romera-Paredes et al. [10] used a two linear layers network to learn the projection from the visual space to the attribute space and regularized the model with the Frobenius norm. Zhang et al. [5] argued that the above mapping direction (visual to semantic) would aggravate the hubness problem [11, 12, 13] and thus chose to learn a projection from the semantic space to the visual space by a deep network. Similarly, Skorokhodov et al. [14] also chose to learn a semantic to visual projection, but they focused on the the application of normalization techniques in ZSL. In addition, some methods try to map visual features and semantic vectors to another shared space. Representatively, Li et al. [15] and Yang et al. [16] both thought the label information of classes is helpful for the learning of the projection. Therefore, they chose the label space as the shared space to learn the relationship between visual features and semantic vectors. Besides, to exploit the common knowledge

shared by both visual and semantic features, Wu et al. [17] introduced reconstruction regularization to the projection from the visual and semantic spaces to a common latent subspace. Furthermore, to balance the information of classes in visual and semantic spaces, Liu et al. [18] proposed Isometric Propagation Network (IPN) to learn the prototypes of classes in each space and align their distribution.

Generative methods use generative models, e.g., GAN and VAE, to learn the projection between visual features and semantic vectors, and then synthesize fake visual features for unseen classes. With the synthesized unseen features, generative methods eliminate the imbalance between seen and unseen data, and thus have a promising performance under the Generalized ZSL (GZSL) setting where the test data contain both seen and unseen samples. Specifically, Xian et al. [19] first proposed to use a Wasserstein GAN to synthesize visual features conditioned on class-level semantic information. Following this idea, Li et al. [20] designed two regularizations to require the generated features to be close to the "soul" samples of real data. In addition, to generate discriminative and representative features, Liu et al. [21] defined a latent space where the features from different classes are orthogonal and used cascade GANs to generate samples. To fully investigate the bilateral connections in ZSL, Li et al. [22] chose to concatenate a conditional WGAN with a bidirectional autoencoder. Different from the above GAN-based methods, Schonfeld et al. [23] proposed to learn a shared latent space of visual features and semantic vectors by modality-specific VAEs. Li et al. [24] used two VAEs to model the latent distribution of samples in visual and semantic spaces and tried to learn modality-invariant latent representations for classes

by leveraging their mutual information and entropy. To extract the semantically related information from the visual features of samples, Chen et al. [25] designed a semantic disentangling module to learn semantic-consistent and semantic-unrelated representations from visual vectors. Specially, for ZSL with fewer seen samples, Verma et al. [26] proposed a generative framework that combines a conditional VAE with a GAN to generate high-quality samples in a meta-learning paradigm.

Various ZSL methods have been proposed and contributed to the ZSL community. However, most of the existing ZSL methods are trained using sufficient seen samples and barely consider the situation where only a tiny number of seen samples are available. Unlike these methods, in this paper, we focus on how to perform ZSL with much fewer seen samples, which is more challenging than the general ZSL setting.

3. Method

3.1. Problem definition

In general ZSL setting, the training data $\mathcal{D}_s = \{\mathcal{X}_s, \mathcal{Y}_s\} = \{\boldsymbol{x}_s^i, y_s^i\}_{i=1}^{N_s}$ consists of N_s samples from s seen classes, where $\boldsymbol{x}_s^i$ and y_s^i represent the visual feature vector and the class label of the i-th seen sample, respectively. Based on $\mathcal{D}_s$, ZSL aims to train a model which can recognize the samples of u unseen classes, that is, $\mathcal{D}_u = \{\mathcal{X}_u, \mathcal{Y}_u\} = \{\boldsymbol{x}_u^i, y_u^i\}_{i=1}^{N_u}$. The challenge of ZSL is that seen and unseen classes are non-overlapping, i.e., $\mathcal{Y}_s \cap \mathcal{Y}_u = \emptyset$. Instead, the semantic descriptions of classes denoted as $\mathcal{A} = \{\mathcal{A}_s, \mathcal{A}_u\} = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_s, \boldsymbol{a}_{s+1}, ..., \boldsymbol{a}_{s+u}\}$ are provided to bridge the gap between the seen and unseen domains. Conventional ZSL evaluates the model only with unseen

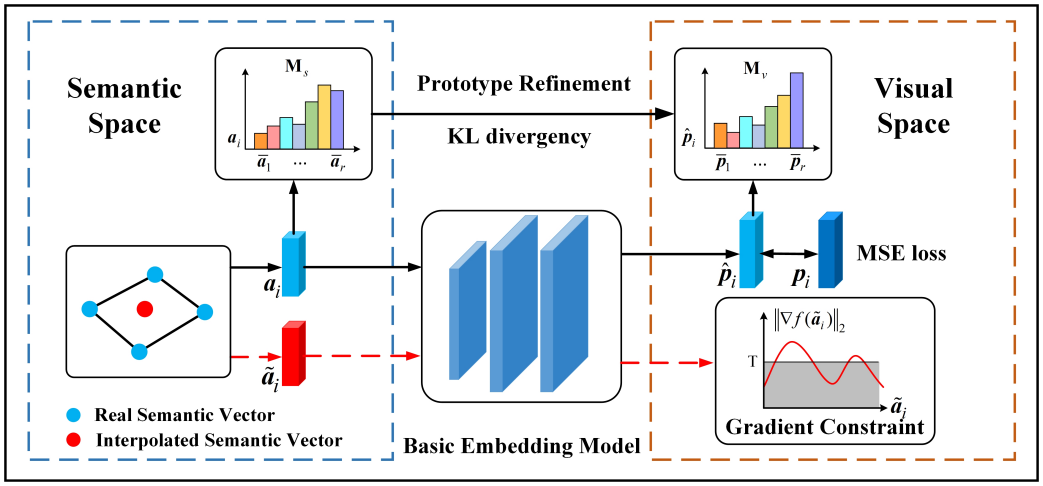


Figure 2: The framework of DGEM, which consists of a basic embedding model and two regularizers, i.e., the prototype refinement loss and the projection smoothing loss. Concretely: (1) Given the semantic vectors of seen classes ($\boldsymbol{a}_i$), we use a basic embedding model to predict their corresponding visual prototypes ($\hat{\boldsymbol{p}}_i$); (2) To alleviate the bias of visual prototypes, we select r anchor classes to calculate the relative distribution matrix in two spaces ($\mathbf{M}_s$ and $\mathbf{M}_v$) and align them; (3) To learn a smooth and robust embedding model, some interpolated semantic vectors ($\tilde{\boldsymbol{a}}_i$) will be generated and we will constrain their gradient norms ($||\nabla f(\tilde{\boldsymbol{a}}_i)||_2$) to enforce Lipschitz continuity.

samples, that is, training a classifier which can $f_{zsl} : \mathcal{X}_u \rightarrow \mathcal{Y}_u$. A more realistic and challenging setting is generalized ZSL (GZSL), in which the test samples are from both seen and unseen classes, i.e., $f_{gzsl} : \mathcal{X}_s \cup \mathcal{X}_u \rightarrow \mathcal{Y}_s \cup \mathcal{Y}_u$.

In this paper, we challenge the implicit assumption of general ZSL that each seen class has sufficient samples, and investigate how to perform ZSL with much fewer seen samples. Specifically, in our setting, each seen class now only has k samples ($k = 1/5$) and the seen data become $\mathcal{D}_s = \{\boldsymbol{x}_s^i, y_s^i\}_{i=1}^{sk}$ where $sk \ll N_s$.

3.2. Overview

To meet the challenges brought by fewer seen samples, we propose a Distribution and Gradient constrained Embedding Model (DGEM), whose framework is shown in Figure 2. To be specific, given the semantic vectors of seen classes ($\boldsymbol{a}_i$), we use a three-layer neural network as our basic embedding model to predict their corresponding visual prototypes ($\hat{\boldsymbol{p}}_i$). A mean square error (MSE) loss is applied to make the predicted visual prototypes close to the visual means of seen classes ($\boldsymbol{p}_i$). To tackle the representation bias problem and the over-fitting problem, we design a prototype refinement loss and a projection smoothing loss to solve them. The former uses the relative distribution of the semantic vectors ($\mathbf{M}_s$) to guide that of the predicted visual prototypes ($\mathbf{M}_v$). The latter enforces the Lipschitz continuity of the model by stabilizing the gradients with respect to some interpolated semantic vectors ($\tilde{\boldsymbol{a}}_i$). After training, we determine the class label of each test sample by searching its nearest visual prototype. The details of DGEM will be introduced in the following subsections.

3.3. Basic embedding model

When there is a small number of training samples, GAN-based methods will suffer severe performance deterioration as their discriminators will easily overfit to the seen data and feedback meaningless information to their generators [27, 28], as shown in Tables 2 and 3. Considering about this, in this paper, we use a three-layer fully connected network, which is simple but effective, as our basic embedding model to learn the projection from semantic vectors to visual features.

Formally, given the semantic vectors of seen classes, i.e., $\mathcal{A}_s = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_s\}$, the basic embedding model f is expected to predict their visual prototypes $\{\hat{\boldsymbol{p}}_i \triangleq f(a_i)\}_{i=1}^{s}$ that are close to the visual means of seen classes. In formula, the target of f is to minimize

$$\mathcal{L}_{mse} = \frac{1}{s}\sum_{i=1}^{s} dis(\hat{\boldsymbol{p}}_i, \boldsymbol{p}_i) = \frac{1}{s}\sum_{i=1}^{s} dis(f(\boldsymbol{a}_i), \boldsymbol{p}_i), \tag{1}$$

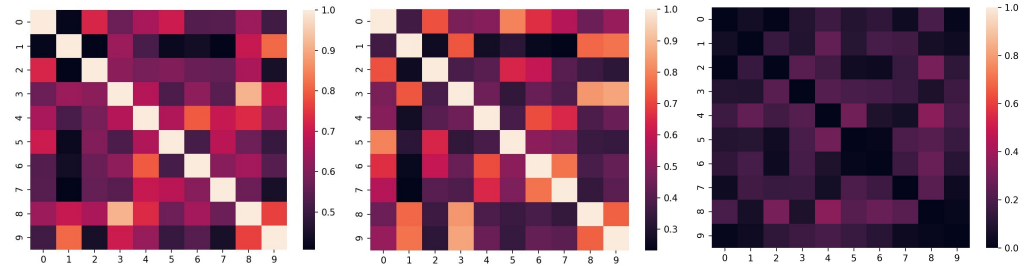
where $dis(\cdot, \cdot)$ is the Euclidean distance and $\boldsymbol{p}_i$ is the ground truth visual prototype of the i-th seen class calculated by k samples:

$$\boldsymbol{p}_i = \frac{1}{k}\sum_{y_j^s=i} \boldsymbol{x}_j^s. \tag{2}$$

The above formula indicates that the quality of the ground truth visual prototypes is quite crucial to the learning of the projection. However, as illustrated in Figure 1, when each seen class only has 1/5 samples, the calculated $\boldsymbol{p}_i$ has a high probability of deviating from the true expectation of the potential distribution. As a consequence, the model will learn a biased projection between visual features and semantic vectors, and generate untrustworthy visual prototypes, which we called *the representation bias problem*. In the following subsection, we will introduce how to use the distribution information of classes in the semantic space to solve this problem.

3.4. Prototype refinement

To reduce the impact of the representation bias problem, we propose to utilize the distribution information of classes in the semantic space to refine the predicted visual prototypes. Specifically, given the visual prototypes and semantic vectors of classes, an intuitive idea is that their relative distribution should be close to each other. In other words, the classes with high/low



(a) Visual similarity (b) Semantic similarity (c) Similarity difference

Figure 3: The similarity heatmaps of AwA2 [6] unseen classes in the visual (a) and semantic spaces (b). We also display the difference between them (c). The brighter the grid, the higher the cosine similarity between the two classes. Note that the visual similarity is calculated by using the mean vectors of each class with sufficient samples.

semantic similarity should also have high/low visual similarity. To validate this intuition, we visualize the similarity matrices of AwA2 [6] unseen classes and their difference, as shown in Figure 3. Take classes 1 (blue whale), 8 (walrus), and 9 (dolphin) as examples. We can observe that they both have high visual and semantic similarities. Furthermore, Figure 3(c) intuitively illustrates that the similarity distributions of classes in two spaces are similar. All this evidence proves that our intuition is tenable in ZSL datasets. Based on the observations, we propose a relatively slack assumption that "*the probability distribution between one class and some anchor classes in the visual domain should be consistent with that in the semantic domain*," which motivates us to align the relative distribution of the predicted visual prototypes to that of the semantic vectors in the training stage.

Specifically, we first apply k-means algorithm [29] to the semantic vectors of seen classes, i.e., $\mathcal{A}_s = \{\boldsymbol{a}_i\}_{i=1}^{s}$, to select r classes as our anchor classes

which can be denoted as

$$\mathcal{A}_r = \{\bar{\boldsymbol{a}}_i\}_{i=1}^{r} = kmeans(\mathcal{A}_s) = kmeans(\{\boldsymbol{a}_i\}_{i=1}^{s}). \tag{3}$$

After that, we calculate the semantic similarities between each seen class and all anchor classes to construct a relative semantic distribution matrix $\mathbf{M}_s = [m_{i,j}^s] \in \mathbb{R}^{s \times r}$ by

$$m_{i,j}^s = \frac{exp(sim(\boldsymbol{a}_i, \bar{\boldsymbol{a}}_j))}{\sum_l exp(sim(\boldsymbol{a}_i, \bar{\boldsymbol{a}}_l))}, \tag{4}$$

where $sim(\cdot, \cdot)$ is the cosine similarity. Similarly, with the predicted visual prototypes $\{\hat{\boldsymbol{p}}_i\}_{i=1}^{s}$, we can also obtain a relative visual distribution matrix $\mathbf{M}_v = [m_{i,j}^v] \in \mathbb{R}^{s \times r}$ by

$$m_{i,j}^v = \frac{exp(sim(\hat{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_j))}{\sum_l exp(sim(\hat{\boldsymbol{p}}_i, \bar{\boldsymbol{p}}_l))}, \tag{5}$$

where $\hat{\boldsymbol{p}}_i = f(\boldsymbol{a}_i)$ and $\bar{\boldsymbol{p}}_j = f(\bar{\boldsymbol{a}}_j)$. Finally, to make the distribution of the predicted visual prototypes be close to that of the semantic vectors, we minimize the Kullback-Leibler divergence between $\mathbf{M}_s$ and $\mathbf{M}_v$:

$$\mathcal{L}_{pr} = D_{KL}(\mathbf{M}_s || \mathbf{M}_v) = \sum_i \sum_j m_{i,j}^s log \frac{m_{i,j}^s}{m_{i,j}^v}. \tag{6}$$

By introducing the distribution information of classes in the semantic space, the distribution of the predicted visual prototypes is constrained. As a result, the quality of the predicted visual prototypes is improved, and the representation bias problem is alleviated.

3.5. Projection smoothing

In addition to the representation bias problem, another challenge in ZSL with fewer seen samples is the over-fitting of the model to the limited seen

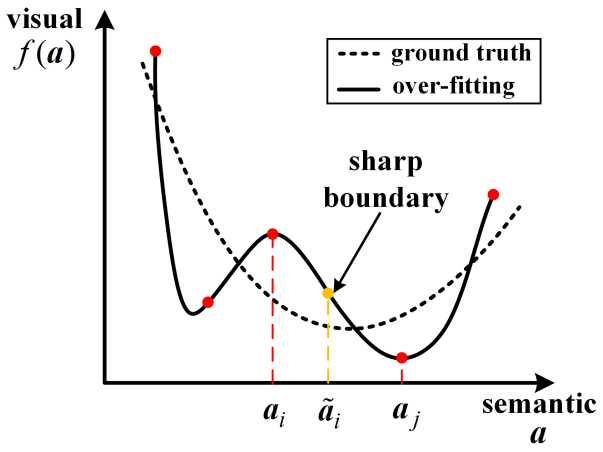


Figure 4: The diagram of over-fitting (black curve) and reasonable fitting (dashed curve). Red points are the visual prototypes of seen semantic vectors while yellow points are that of the interpolated semantic vectors. We constrain the gradients with respect to the interpolated semantic vectors to smooth the projection.

data. Concretely, as the visual prototypes of seen classes are biased and the information of unseen classes is unavailable in the training stage, the model tends to learn a semantic-visual projection that fits seen data well but can not generalize to unseen data as shown in Figure 4. In this condition, sharp boundaries exist between seen classes, and the whole projection becomes "steep". To tackle this problem, we introduce a projection smoothing loss to enforce the Lipschitz continuity [30] of the model, which limits how fast the projection can change and is particularly suitable for the situations where only a tiny number of training samples are available [31].

Specifically, the embedding model f is said to be T-Lipschitz continuous if it satisfies

$$d_v(f(\boldsymbol{a}_i), f(\boldsymbol{a}_j)) \leq T d_s(\boldsymbol{a}_i, \boldsymbol{a}_j), \tag{7}$$

where $d_v(\cdot,\cdot)$ and $d_s(\cdot,\cdot)$ denote the metrics in the visual and semantic spaces, respectively. When $d_v(\cdot,\cdot)$ and $d_s(\cdot,\cdot)$ are the euclidean distance and $\boldsymbol{a}_i$, $\boldsymbol{a}_j$ are close enough, Eq (7) can be approximately seen as requiring the gradients with respect to $\boldsymbol{a}_i$ to be less than a constant T. To meet this condition, we choose to generate some interpolated semantic vectors by combining the real ones:

$$\begin{aligned} &\tilde{\boldsymbol{a}}_i = \alpha_1 \boldsymbol{a}_i + \alpha_2 \boldsymbol{a}_{i,1} + \alpha_3 \boldsymbol{a}_{i,2} + \alpha_4 \boldsymbol{a}_{i,3}, \\ &\text{s.t.} \quad \alpha_i > 0, \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1, \end{aligned} \tag{8}$$

where $\boldsymbol{a}_i$ is the semantic vector of a random seen class and $\boldsymbol{a}_{i,1}$, $\boldsymbol{a}_{i,2}$, $\boldsymbol{a}_{i,3}$ are its three nearest neighbors. With the interpolated semantic vectors, we constrain the gradients with respect to them to be less than T:

$$\mathcal{L}_{ps} = \frac{1}{N_{inter}} \sum_i max(0, ||\nabla_{\tilde{\boldsymbol{a}}_i} f(\tilde{\boldsymbol{a}}_i)||_2 - T), \tag{9}$$

where N_{inter} is the number of the interpolated semantic vectors.

By constraining the gradients with respect to the interpolated semantic vectors, the generation of sharp boundaries can be avoided, and the projection will become more stable. As a result, the generalization capability of the model will be highly improved.

3.6. Objective and inference

In the training stage, we use the training data $\mathcal{D}_s$ and the semantic vectors of seen classes $\mathcal{A}_s$ to learn the projection from the semantic vectors to the visual prototypes. The objective of the model is formulated by integrating the introduced three loss functions:

$$\mathcal{L} = \mathcal{L}_{mse} + \alpha \mathcal{L}_{pr} + \beta \mathcal{L}_{ps}, \tag{10}$$

where α and β are weight parameters.

At test time, given the semantic vectors of unseen classes $\mathcal{A}_u = \{\boldsymbol{a}_{s+i}\}_{i=1}^{u}$, we use the trained model to predict their corresponding visual prototypes and determine the class label of each sample $\boldsymbol{x}$ by searching its nearest visual prototype:

$$\hat{y} = \arg\min_{i} dis(\boldsymbol{x}, f(\boldsymbol{a}_i)), \tag{11}$$

where $dis(\cdot, \cdot)$ denotes the euclidean distance.

3.7. Comparison with related works

After introducing the details of our method, we would like to clarify the difference between it and those methods with similar techniques.

Prototype refinement. Many methods have tried to use prototypes as the visual representations of classes in ZSL [18, 32]. However, to utilize the substantial information of classes in different spaces, they generally choose to align the distributions of all classes in visual and semantic modalities. Different from them, our target is to alleviate the visual bias caused by limited seen samples. More importantly, we align the relative distribution between all and r representative classes, which is a weaker constraint.

Lipschitz continuity. By observing Eq. (9), we can find that its form is similar to the gradient penalty term of WGAN-based methods. However, there are two main differences between our method and them: (1) We constrain the gradients with respect to the interpolated semantic vectors. In contrast, the gradient penalty term in WGAN is applied to the intermediate vectors between real and generated vectors. (2) The gradient penalty term in WGAN is used to improve the training stability of networks, while our motivation is to make the learned projection smooth and robust.

Table 1: Dataset statistics of ZSL datasets with fewer seen samples in terms of dimension of semantic (attribute) vectors, number of seen (s) and unseen (u) classes, number of training samples when $k = 1/5$, number of test seen and test unseen samples.

Dataset	Attribute	s	u	Train		Test	
				1	5	seen	unseen
aPY [3]	64	20	12	20	100	1483	7924
AwA1 [33]	85	40	10	40	200	4958	5685
AwA2 [6]	85	40	10	40	200	5882	7913
SUN [34]	102	645	72	645	3225	2580	1440
CUB [35]	312	150	50	150	750	1764	2967

4. Experiments

In this section, we will present our experimental setting and compare our method with some representative methods. Besides, we will conduct an ablation study to evaluate the effectiveness of each component and analyze the hyperparameters of our model.

4.1. Datasets

We conduct experiments on five widely used datasets, which are APascal-aYahoo (aPY) [3], Animals with Attribute 1 (AwA1) [33], Animals with Attribute 2 (AwA2) [6], SUN Attribute (SUN) [34] and Caltech-UCSD Birds 200-2011 (CUB) [35]. Based on these five public datasets, we randomly select 1/5 samples from each seen class to construct the training data for ZSL with fewer seen samples. Specially, to reduce the influence of random sampling, we repeat sampling 10 times for both $k = 1$ and $k = 5$ by setting different random seeds. In other words, there exist 10 different training sets

for each dataset, and the final performance of each method will be measured by calculating the mean and standard deviation of all the results. Table 1 records the statistics of five datasets when each seen class only has 1/5 samples. As can be seen, in this setting, the training samples of datasets are quite limited.

4.2. Experimental setting

4.2.1. Evaluation metric

For conventional ZSL task, we use the average per-class top-1 accuracy [6] of test unseen data to measure the performance of methods, which is defined as:

$$acc_u = \frac{1}{u} \sum_{y \in \mathcal{Y}_u} \frac{\#\textbf{correct predictions in } y}{\#\textbf{samples in } y}, \tag{12}$$

where u is the number of unseen classes and $\mathcal{Y}_u$ is the label set of test unseen samples.

For generalized ZSL, we use the harmonic mean of seen and unseen accuracies [6] to measure the overall performance of methods, which is defined as:

$$H = \frac{2 \times acc_s \times acc_u}{acc_s + acc_u}, \tag{13}$$

where acc_s and acc_u represent the accuracy of the model on test seen and test unseen samples, respectively.

4.2.2. Implementation and compared methods.

For the sake of fair comparison and analysis, we follow the setting of [6] to use the 2048 dimensional visual vectors extracted by ResNet-101 and attribute vectors as the visual features ($\boldsymbol{x}_i$) and the semantic vectors ($\boldsymbol{a}_i$)

of classes, respectively. The embedding model f is implemented via fully-connected layers and Rectified Linear Unit (ReLU) activation. Specifically, it contains three fully connected layers with 2048 hidden units, followed by a Leaky ReLU function and a ReLU function. In each epoch during training, we input the semantic vectors of all seen classes into the embedding model. For weight parameters, we tune α and β in $\{1e-6, ..., 1e-3\}$ and $\{1e-4, ..., 1e-3\}$, respectively.

The compared methods include representative ones published in the past few years and the recently reported state-of-the-art ones. Specifically, we compare our method with ESZSL [10], SAE [36], f-CLSWGAN [19], LisGAN [20], TF-VAEGAN [37], CE-GZSL [38], and SDGZSL [18]. Among these methods, ESZSL and SAE are embedding methods, while f-CLSWGAN, LisGAN, TF-VAEGAN, CE-GZSL and SDGZSL are generative methods. All the results of these methods are reproduced by implementing their open source codes on Github and personal pages.

4.3. Experimental results

4.3.1. Conventional ZSL

Table 2 shows the results of all methods on five datasets under the conventional ZSL setting. As can be seen, DGEM outperforms the other compared methods on most benchmarks when each class only has a very small number of samples. To be specific, under the most challenging setting that each seen class only has one sample ($k = 1$), DGEM obtains the state-of-the-art performance on all datasets, which is 3.1% higher than the second-best results on average. When each seen class has more samples ($k = 5$), the diversity of samples becomes more affluent, and the representation bias problem is alle-

Table 2: Conventional ZSL comparison on five datasets when each seen class only has 1/5 samples ($k = 1/5$). The results are shown in the form of "mean±std". The best result is in **bold**. The results of CE-GZSL when $k = 1$ are unavailable as its contrastive loss requires at least two samples per class.

k	Method	AwA1	AwA2	aPY	SUN	CUB
1	ESZSL [10]	50.8±4.0	52.7±3.7	24.8±3.3	48.9±1.3	33.7±1.9
	SAE [36]	13.0±1.1	14.9±2.2	11.0±1.8	36.4±0.7	11.2±1.8
	f-CLSWGAN [19]	34.7±1.6	39.1±2.6	22.6±1.2	49.4±0.4	30.4±1.1
	LisGAN [20]	42.3±3.2	38.9±1.9	28.9±2.4	51.6±0.6	16.0±0.6
	TF-VAEGAN [37]	43.2±2.9	44.8±2.0	26.3±2.7	49.4±0.6	31.9±0.1
	CE-GZSL [38]	-	-	-	-	-
	SDGZSL [25]	41.2±2.8	43.8±2.9	27.3±2.8	50.3±0.9	35.8±1.3
	DGEM(ours)	**53.3±4.0**	**58.3±5.0**	**30.1±2.4**	**54.9±0.5**	**36.8±1.5**
5	ESZSL [10]	57.3±2.6	57.2±1.9	34.9±3.2	53.2±1.1	42.1±0.7
	SAE [36]	19.8±3.3	19.7±3.9	14.0±2.2	42.4±0.8	19.8±1.8
	f-CLSWGAN [19]	52.1±2.4	55.8±1.4	33.6±2.2	56.1±0.5	47.2±0.8
	LisGAN [20]	49.9±1.2	48.8±2.4	35.0±1.7	56.9±0.5	42.3±0.9
	TF-VAEGAN [37]	47.7±2.5	46.7±1.0	32.4±1.0	**60.5±0.8**	45.2±2.8
	CE-GZSL [38]	55.3±2.8	59.3±1.8	34.7±2.2	58.2±2.5	32.2±1.9
	SDGZSL [25]	52.1±2.9	54.3±2.0	36.3±2.4	57.4±0.9	**51.4±0.7**
	DGEM(ours)	**63.9±2.1**	**67.3±2.0**	**37.4±2.0**	59.2±1.0	45.9±1.1

viated. However, we can find that DGEM still achieves the best performance on AwA1, AwA2 and aPY datasets, which proves its robustness. On SUN and CUB datasets, DGEM achieves comparable results with the generative methods (f-CLSWGAN, LisGAN, TF-VAEGAN, CE-GZSL, and SDGZSL). The reason is that SUN and CUB datasets have much more classes, and thus their training samples are relatively sufficient, as shown in Table 1. As a result, the influence of limited seen samples becomes small, and the effect of designed $\mathcal{L}_{pr}$ and $\mathcal{L}_{ps}$ is reduced.

Table 3: Generalized ZSL comparison on five datasets when each seen class only has 1 or 5 samples ($k = 1/5$). "S" and "U" represent the accuracy of the model on seen and unseen data respectively. "H" is the harmonic mean of "S" and "U". The best result is in **bold**. Due to the limited space, we only show the mean of results in this table. Similar, the results of CE-GZSL when $k = 1$ are unavailable as its contrastive loss requires at least two samples per class.

k	Method	AwA1			AwA2			aPY			SUN			CUB		
		S	**U**	**H**	**S**	**U**	**H**	**S**	**U**	**H**	**S**	**U**	**H**	**S**	**U**	**H**
1	ESZSL [10]	56.4	5.0	9.1	61.8	6.4	11.4	54.0	2.2	4.1	23.7	10.2	14.2	28.9	9.1	13.8
	SAE [36]	0.5	10.9	1.0	0.8	10.7	1.5	0.5	8.9	0.8	19.8	8.5	11.8	4.9	10.1	6.6
	f-CLSWGAN [19]	0.1	10.7	0.1	0.1	6.4	0.1	2.1	9.5	3.4	5.4	44.3	9.6	2.1	17.4	3.6
	LisGAN [20]	6.4	8.4	7.2	2.7	19.6	4.7	5.1	17.3	7.7	6.8	42.8	11.8	4.3	14.5	6.5
	TF-VAEGAN [37]	7.7	12.1	9.4	11.1	12.9	11.7	8.8	10.3	9.3	6.7	23.1	10.3	6.6	8.0	7.0
	CE-GZSL [38]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	SDGZSL [25]	10.1	1.7	2.5	10.2	2.3	3.5	9.0	0.9	1.2	35.3	8.7	14.0	17.3	5.4	4.0
	DGEM(ours)	28.9	36.6	**32.0**	34.8	41.4	**37.7**	27.8	23.6	**25.5**	13.8	34.1	**19.7**	12.8	23.4	**16.5**
5	ESZSL [10]	74.9	6.1	11.3	77.5	7.1	12.9	66.5	1.9	3.3	33.8	11.4	17.0	45.3	13.0	20.2
	SAE [36]	18.1	17.5	17.5	18.0	16.2	16.4	9.8	12.6	10.6	36.6	10.5	16.3	25.1	14.7	18.5
	f-CLSWGAN [19]	9.4	10.2	9.6	13.2	8.4	10.2	26.2	22.4	24.1	36.1	45.9	40.4	16.4	45.1	13.8
	LisGAN [20]	43.8	32.6	37.3	43.0	33.9	37.6	36.6	28.4	31.0	32.0	54.0	40.1	26.5	32.6	29.0
	TF-VAEGAN [37]	41.5	34.4	37.5	46.1	37.8	41.4	38.5	26.0	30.8	34.8	49.3	40.8	25.2	34.1	28.2
	CE-GZSL [38]	19.0	30.7	22.7	25.4	33.2	27.9	28.0	26.8	26.5	33.7	37.1	35.1	11.1	22.8	14.6
	SDGZSL [25]	6.5	20.4	9.8	7.1	21.8	10.7	30.6	7.4	11.7	45.4	39.0	**41.9**	45.1	28.1	**34.6**
	DGEM(ours)	63.3	46.8	**53.7**	69.8	43.4	**53.4**	59.5	29.9	**39.8**	42.4	35.4	38.5	36.0	30.2	32.8

In summary, the reduction of training samples will significantly affect the performance of general ZSL methods. In contrast, DGEM is less dependent on the number of training samples and has a stronger generalization capability.

4.3.2. Generalized ZSL

To further evaluate the effectiveness and robustness of DGEM, we conduct generalized ZSL experiments on five datasets, and the results are shown in

Table 3.

Compared with ZSL, GZSL is more realistic and challenging, as its test samples come from both seen and unseen classes. In this challenging setting, we can observe that DGEM achieves the best performance on all datasets when $k = 1$ and 3 of 5 datasets when $k = 5$, which further proves that DGEM can effectively meet the challenges brought by limited seen samples. In contrast, the performance of the generative methods (f-CLSWGAN, LisGAN, TF-VAEGAN, CE-GZSL, and SDGZSL) has a severe deterioration, which is mainly caused by the following reasons: (1) Limited seen samples can not provide sufficient information to their discriminators, causing that the feedback from the discriminators to the generators becomes meaningless and the training starts to diverge [27]. (2) The imbalance between real seen samples and synthesized unseen samples affects the performance of the finally trained softmax classifier.

In general, DGEM is superior to the other methods in the GZSL setting. It could be a promising method to address GZSL with fewer seen samples.

4.4. Ablation study

To dive deeper into the role of each component in DGEM, we conduct ablation experiments on AwA2, aPY, and CUB datasets. The results are shown in Table 4.

The results show that both the prototype refinement loss ($\mathcal{L}_{pr}$) and the projection smoothing loss ($\mathcal{L}_{ps}$) contribute to the final performance of DGEM. Concretely, the introduction of $\mathcal{L}_{pr}$ improves the performance of the basic embedding model ($\mathcal{L}_{mse}$) by 1.7% and 0.7% on average for ZSL and GZSL, respectively. These results prove that the semantic distribution information

Table 4: Ablation experiments on AwA2, aPY and CUB datasets under ZSL and GZSL settings when each seen class only has one sample ($k = 1$).

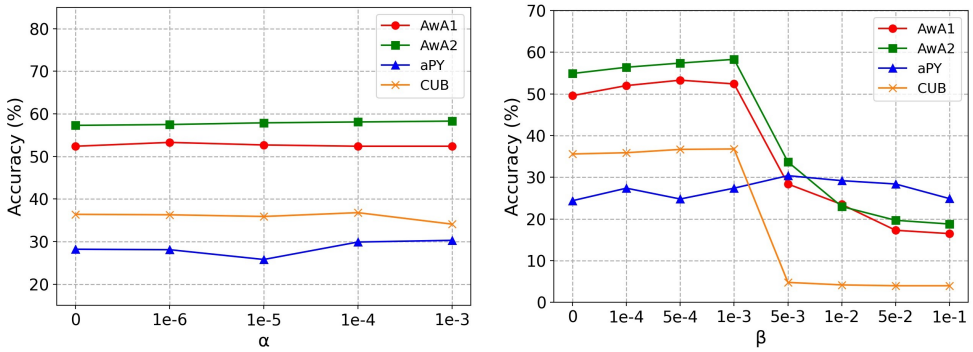
$\mathcal{L}_{mse}$	$\mathcal{L}_{pr}$	$\mathcal{L}_{gs}$	AwA2		aPY		CUB	
			ZSL	GZSL	ZSL	GZSL	ZSL	GZSL
✓			53.2	35.7	22.6	19.5	34.5	15.1
✓	✓		54.9	36.1	24.4	20.8	35.6	15.5
✓		✓	57.3	37.2	28.2	24.3	36.4	16.0
✓	✓	✓	58.3	37.7	30.1	25.5	36.9	16.5

can play a positive role in alleviating the bias of the visual prototypes. Another point worth noting is the effect of $\mathcal{L}_{ps}$. After applying $\mathcal{L}_{ps}$, we can observe that the accuracy of the basic embedding model is greatly improved by 3.9% and 2.4% on ZSL and GZSL tasks, respectively. On the whole, $\mathcal{L}_{ps}$ plays a more prominent role than $\mathcal{L}_{pr}$. The reason is that the degree of the representation bias is related to the quality of available samples, while the over-fitting problem always exists and is especially serious when each class only has one sample. In conclusion, the proposed $\mathcal{L}_{pr}$ and $\mathcal{L}_{ps}$ can effectively mitigate the impact of limited seen samples and help the embedding model gain better performance under ZSL and GZSL settings.

In Table 5, we further display the detailed ZSL results of four variants on the AwA2 dataset with different training sets. We can observe that the results of the model with different training sets have a large variance which means that the quality of the one sample greatly affects the performance of the learned projection. For those tasks whose training samples have poor quality (e.g., tasks 2, 4, 5, and 7), our $\mathcal{L}_{pr}$ and $\mathcal{L}_{ps}$ can effectively reduce the negative impact of biased samples and help the embedding model achieve

Table 5: Detailed ablation results on AwA2 dataset under ZSL setting. As mentioned in Section 4.1, we generate 10 different training sets for each dataset. Here is the results of the model with each training set. Note that each seen class only has one sample ($k = 1$).

$\mathcal{L}_{mse}$	$\mathcal{L}_{pr}$	$\mathcal{L}_{gs}$	1	2	3	4	5	6	7	8	9	10	Avg±Std
✓			64.1	48.3	58.8	43.6	47.2	59.9	48.6	53.0	53.8	54.9	53.2±6.1
✓	✓		64.9	51.8	61.1	45.7	48.3	60.8	52.2	54.9	54.1	55.5	54.9±6.0
✓		✓	65.7	54.6	62.3	48.9	53.1	61.8	56.7	58.0	53.8	58.0	57.3±4.7
✓	✓	✓	65.8	56.2	63.9	48.9	53.4	63.9	58.4	58.4	54.6	59.0	58.3±5.0



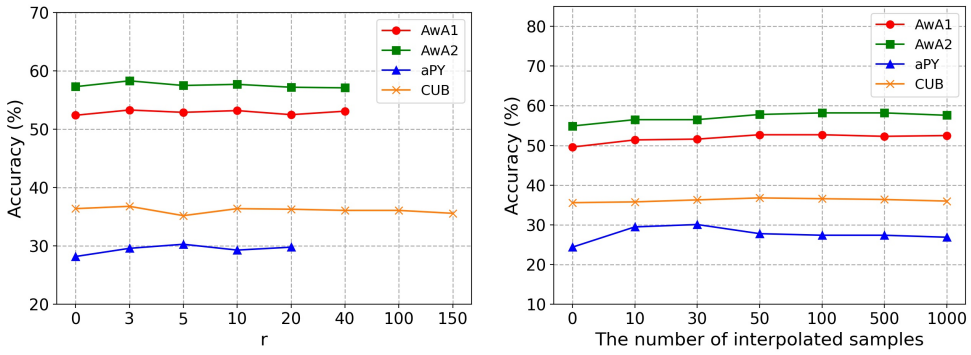
(a) Weight of $\mathcal{L}_{pr}$ (b) Weight of $\mathcal{L}_{gs}$

Figure 5: ZSL performance of DGEM on AwA1, AwA2, aPY and CUB datasets with respect to α and β.

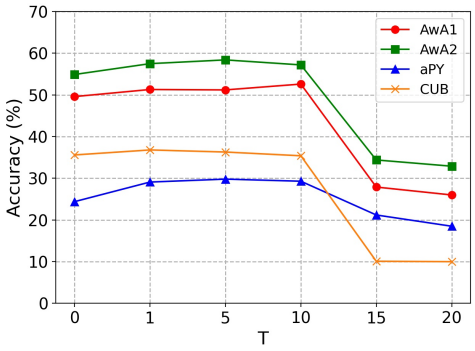
better performance.

4.5. Parameter analysis

In this section, we discuss the effect of each hyperparameter in our method and analyze their influence on the performance of DGEM.



(a) The number of anchor classes (b) The number of interpolated samples



(c) Lipschitz constant

Figure 6: Parameter sensitivity of the proposed DGEM on AwA1, AwA2, aPY and CUB datasets. When changing the value of one parameter, the other parameters are fixed.

4.5.1. Weight parameters

In our method, we use α and β to control the weights of $\mathcal{L}_{pr}$ and $\mathcal{L}_{ps}$, respectively. To analyze the sensitivity of the model concerning α and β, we choose AwA1, AwA2, aPY, and CUB as example datasets to conduct experiments. The results are shown in Figure 5. In general, the performance of DGEM is not sensitive to α, and setting it to $1e-4$ is the best choice for most datasets. As for β, its influence on the final accuracy is relatively large,

but we can empirically set it to $1e-3$, which can help the model achieve peak performance in most scenarios.

4.5.2. The number of anchor classes

Figure 6(a) shows the accuracy of DGEM with different numbers of anchor classes. Note that $r = 0$ means $\mathcal{L}_{pr}$ has no effect, and the maximum value of r in five datasets is equal to the number of their training classes. As can be seen, when $r = 3$, DGEM can achieve the best performance on most benchmarks. In addition, we can find that using the global distribution information ($r = s$) is not as good as using the relative distribution information ($r = 3$). The reason is that setting r to the number of training classes will push the distribution of the predicted visual prototypes to be identical to that of the semantic vectors, which is too strict.

4.5.3. The number of interpolated samples

The purpose of $\mathcal{L}_{gs}$ is to eliminate the sharp boundaries between seen classes. Only using the real seen classes is not enough to achieve this goal. Thus we choose to generate some interpolated semantic vectors by combining the real ones, as detailed in Section 3.5. Figure 6(b) shows the results of DGEM with different N_{inter}. As can be seen, generating 30 interpolated semantic vectors is the best choice for aPY datasets, and setting N_{inter} to 100 is recommended for the other three datasets.

4.5.4. Lipschitz constant

In Eq. (9), Lipschitz constant T is used as an upper boundary to limit the gradients concerning the interpolated semantic vectors. When T is large, sharp boundaries will exist between seen classes. When T is small, the

Table 6: Training and inference time of different methods on AwA2 dataset when each seen class only has k samples ($k = 1/5$). "epoch" is the number of iterations required for different models to converge. The unit is seconds.

Method	$k = 1$		$k = 5$		Inference
	epoch	time	epoch	time	
ESZSL [10]	-	5.8	-	6.2	0.3
SAE [36]	-	0.7	-	0.8	0.01
f-CLSWGAN [19]	500	98.4	500	300.3	1.2
LisGAN [20]	500	166.8	500	255.9	4.7
TF-VAEGAN [37]	500	308.8	500	1850.6	5.5
CE-GZSL [38]	-	-	200	604.4	8.3
SDGZSL [25]	700	58.5	700	264.5	2.2
DGEM (ours)	300	10.6	300	17.7	0.01

learned projection will become steady. To investigate its influence, we conduct experiments to evaluate the performance of DGEM with different T, and the results are shown in Figure 6(c). We can observe that DGEM can achieve the best performance on AwA1, AwA2, aPY, and CUB datasets when T is set to 5, 10, 5, and 1, respectively. For different datasets, we conjecture that the optimal value of T is related to the distribution and spatial relationship of samples.

4.6. Computation cost

Table 6 shows the computation costs of different methods on AwA2 dataset. Note that SAE is based on MATLAB and all the other methods are implemented by python and PyTorch. Not surprisingly, there is a huge computation cost gap between embedding and generative methods. Even

there are few training samples, generative methods still need much time to train their generative adversarial models. Unlike these methods, the main component of DGEM is a multi-layer perceptron. Thus it can achieve better performance with a much smaller computation cost. As for the inference stage, generative methods need to synthesize sufficient visual features and then train a softmax classifier, which will cost much time. Compared with them, our methods can quickly obtain the predictions as our classification method is a nearest neighbor search algorithm.

5. Conclusion

In this paper, we investigate how to perform zero-shot learning (ZSL) with fewer seen samples. We summarize the main challenges brought by limited seen samples as the representation bias problem and the over-fitting problem, and we propose a distribution and gradient constrained embedding model (DGEM) to solve them. Specifically, DGEM learns a projection from the semantic vectors of classes to the corresponding visual prototypes. To alleviate the bias of the visual prototypes calculated by a few samples, we design a prototype refinement loss to align the relative distribution of class prototypes in the semantic and visual spaces. For the over-fitting problem, we introduce a projection smoothing loss to enforce the Lipschitz continuity of the model by constraining the gradients with respect to some interpolated semantic vectors. To evaluate the performance of DGEM, we choose five commonly used datasets as our benchmarks, and the results show that DGEM outperforms the other methods when each seen class only has 1/5 samples. As DGEM now mainly focuses on the problems caused by limited seen sam-

ples, its performance on fine-grained datasets may be not competitive. To this end, we would like to introduce self-supervised learning techniques to enhance its capability of learning discriminative features in the future.

References

[1] H. Larochelle, D. Erhan, Y. Bengio, Zero-data learning of new tasks, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Vol. 2, 2008, pp. 646–651.

[2] X. Li, M. Fang, D. Feng, H. Li, J. Wu, Learning unseen visual prototypes for zero-shot classification, Knowledge-Based Systems 160 (2018) 176–187.

[3] A. Farhadi, I. Endres, D. Hoiem, D. Forsyth, Describing objects by their attributes, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2009, pp. 1778–1785.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, Vol. 2, 2013, pp. 3111–3119.

[5] L. Zhang, T. Xiang, S. Gong, Learning a deep embedding model for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017, pp. 2021–2030.

[6] Y. Xian, C. H. Lampert, B. Schiele, Z. Akata, Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly, IEEE

Transactions on Pattern Analysis and Machine Intelligence 41 (9) (2019) 2251–2265. doi:10.1109/TPAMI.2018.2857768.

[7] L. van der Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (86) (2008) 2579–2605.

[8] H. Wu, Y. Yan, S. Chen, X. Huang, Q. Wu, M. K. Ng, Joint visual and semantic optimization for zero-shot learning, Knowledge-Based Systems 215 (2021) 106773.

[9] B. Liu, Q. Dong, Z. Hu, Semantic-diversity transfer network for generalized zero-shot learning via inner disagreement based ood detector, Knowledge-Based Systems 229 (2021) 107337.

[10] B. Romera-Paredes, P. Torr, An embarrassingly simple approach to zero-shot learning, in: International Conference on Machine Learning, 2015, pp. 2152–2161.

[11] M. Radovanovic, A. Nanopoulos, M. Ivanovic, Hubs in space: Popular nearest neighbors in high-dimensional data, Journal of Machine Learning Research 11 (sept) (2010) 2487–2531.

[12] N. Tomasev, M. Radovanovic, D. Mladenic, M. Ivanovic, The role of hubness in clustering high-dimensional data, IEEE Transactions on Knowledge and Data Engineering 26 (3) (2014) 739–751. doi:10.1109/TKDE.2013.25.

[13] A. Lazaridou, G. Dinu, M. Baroni, Hubness and pollution: Delving into cross-space mapping for zero-shot learning, in: Proceedings of the 53rd

Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 270–280.

[14] I. Skorokhodov, M. Elhoseiny, Class normalization for (continual)? generalized zero-shot learning, in: International Conference on Learning Representations, 2021.

[15] J. Li, X. Lan, Y. Long, Y. Liu, X. Chen, L. Shao, N. Zheng, A joint label space for generalized zero-shot classification, IEEE Transactions on Image Processing 29 (2020) 5817–5831.

[16] Y. Liu, X. Gao, Q. Gao, J. Han, L. Shao, Label-activating framework for zero-shot learning, Neural Networks 121 (2020) 1–9.

[17] H. Wu, Y. Yan, S. Chen, X. Huang, Q. Wu, M. K. Ng, Joint visual and semantic optimization for zero-shot learning, Knowledge-Based Systems 215 (2021) 106773.

[18] L. Liu, T. Zhou, G. Long, J. Jiang, X. Dong, C. Zhang, Isometric propagation network for generalized zero-shot learning, in: International Conference on Learning Representations, 2021.

[19] Y. Xian, T. Lorenz, B. Schiele, Z. Akata, Feature generating networks for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 5542–5551.

[20] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, Z. Huang, Leveraging the invariant side of generative zero-shot learning, in: Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7402–7411.

[21] J. Liu, L. Fu, H. Zhang, Q. Ye, W. Yang, L. Liu, Learning discriminative and representative feature with cascade gan for generalized zero-shot learning, Knowledge-Based Systems 236 (2022) 107780.

[22] J. Li, M. Jing, K. Lu, L. Zhu, H. T. Shen, Investigating the bilateral connections in generative zero-shot learning, IEEE Transactions on Cybernetics (2021).

[23] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, Z. Akata, Generalized zero-and few-shot learning via aligned variational autoencoders, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8247–8255.

[24] J. Li, M. Jing, L. Zhu, Z. Ding, K. Lu, Y. Yang, Learning modality-invariant latent representations for generalized zero-shot learning, in: Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 1348–1356.

[25] Z. Chen, Y. Luo, R. Qiu, S. Wang, Z. Huang, J. Li, Z. Zhang, Semantics disentangling for generalized zero-shot learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 8712–8720.

[26] V. K. Verma, A. Mishra, A. Pandey, H. A. Murthy, P. Rai, Towards zero-shot learning with fewer seen class examples, in: Proceedings of

the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2241–2251.

[27] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, Training generative adversarial networks with limited data, in: Advances in Neural Information Processing Systems, Vol. 33, 2020, pp. 12104–12114.

[28] R. Webster, J. Rabin, L. Simon, F. Jurie, Detecting overfitting of deep generative networks via latent recovery, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11273–11282.

[29] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, 1967, pp. 281–297.

[30] L. Armijo, Minimization of functions having lipschitz continuous first partial derivatives, Pacific Journal of mathematics 16 (1) (1966) 1–3.

[31] H. Gouk, E. Frank, B. Pfahringer, M. J. Cree, Regularisation of neural networks by enforcing lipschitz continuity, Machine Learning 110 (2) (2021) 393–416.

[32] Y. Wang, H. Zhang, Z. Zhang, Y. Long, L. Shao, Learning discriminative domain-invariant prototypes for generalized zero shot learning, Knowledge-Based Systems (2020) 105796.

[33] C. H. Lampert, H. Nickisch, S. Harmeling, Attribute-based classification for zero-shot visual object categorization, IEEE Transactions

on Pattern Analysis and Machine Intelligence 36 (3) (2014) 453–465. doi:10.1109/TPAMI.2013.140.

[34] G. Patterson, J. Hays, Sun attribute database: Discovering, annotating, and recognizing scene attributes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2012, pp. 2751–2758.

[35] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200-2011 dataset (2011).

[36] E. Kodirov, T. Xiang, S. Gong, Semantic autoencoder for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017, pp. 3174–3183.

[37] S. Narayan, A. Gupta, F. S. Khan, C. G. Snoek, L. Shao, Latent embedding feedback and discriminative features for zero-shot classification, in: European Conference on Computer Vision, 2020, pp. 479–495.

[38] Z. Han, Z. Fu, S. Chen, J. Yang, Contrastive embedding for generalized zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2371–2381.