Research Article

A Distinguishable Pseudo-Feature Synthesis Method for Generalized Zero-Shot Learning

Yunpeng Jia 1, Xiufen Ye 1, Yusong Liu 1, Huiming Xing 1, and Shuxiang Guo 1,2

1 College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin, Heilongjiang 150001, China
2 Faculty of Engineering, Kagawa University, Takamatsu, Kagawa 7608521, Japan

Correspondence should be addressed to Xiufen Ye; yexiufen@hrbeu.edu.cn

Received 11 June 2022; Revised 13 October 2022; Accepted 3 November 2022; Published 29 November 2022

Abstract

Generalized zero-shot learning (GZSL) aims to classify seen classes and unseen classes that are disjoint simultaneously. Hybrid approaches based on pseudo-feature synthesis are currently the most popular among GZSL methods. However, they suffer from problems of negative transfer and low-quality class discriminability, causing poor classification accuracy. To address them, we propose a novel GZSL method of distinguishable pseudo-feature synthesis (DPFS). The DPFS model can provide high-quality distinguishable characteristics for both seen and unseen classes. Firstly, the model is pretrained by a distance prediction loss to avoid overfitting. Then, the model only selects attributes of similar seen classes and makes sparse representations based on attributes for unseen classes, thereby overcoming negative transfer. After the model synthesizes pseudo-features for unseen classes, it disposes of the pseudo-feature outliers to improve the class discriminability. The pseudo-features are fed into a classifier of the model together with features of seen classes for GZSL classification. Experimental results on four benchmark datasets verify that the proposed DPFS has GZSL classification performance better than that in existing methods.

1. Introduction

Target classification and recognition have been dramatically improved with the development of deep learning technologies. Traditional deep learning methods rely heavily on large-scale labelled training datasets such as ImageNet [1]. However, some are infeasible in extreme cases without labelled samples of some classes [2]. To address it, zero-shot learning (ZSL), which imitates the process of human recognition, has been proposed to link seen classes (available in training datasets) and unseen ones (not available in training datasets) using auxiliary information (e.g., attributes [3] and word vectors [4]). Conventional ZSL methods only consider the recognition of unseen classes but neglect that of seen classes. It leads to the failure of simultaneous recognition of them [5]. Subsequently, generalized zero-shot learning (GZSL) [6] has been found to address it.

Most previous GZSL works are mainly divided into mapping-based approaches [7, 8] and hybrid approaches. The former learns a visual-semantic projection model trained with labelled samples. However, they are prone to overfitting due to limitation of labelled sample numbers and domain shift between disjointed seen classes and unseen classes [9], failing in unseen class classification. The latter, including generating-based approaches [10] and synthesis-based ones, has been proposed to alleviate overfitting. Generating-based approaches (e.g., generative adversarial networks (GANs) [11] and variational auto-encoders (VAEs) [12]) generate pseudo-features for unseen classes with prior semantic knowledge. However, they suffer from mode collapse [13] because it is challenging to train hybrid models. Unlike them, synthesis-based approaches [14–16] synthesize pseudo-features for unseen classes by using semantic information and seen class features. However, they suffer from negative transfer [17] and low-quality class discriminability [18].

In this paper, we propose a novel two-stage method of distinguishable pseudo-feature synthesis (DPFS) for GZSL tasks, as shown in Figure 1. Here, the embedding network and the preclassifier are jointly pretrained to extract...
distinguishable features for seen classes and simultaneously predict prototypes for unseen ones in stage 1. It ensures that the features of seen classes are well-kept and avoids overfitting effectively. Next, distinguishable pseudo-features of unseen classes are synthesized through the attribute projection module (APM) and the pseudo-feature synthesis module (PFSM) in stage 2. Here, for each unseen class, APM builds a sparse representation based on attributes to output a base vector. It only uses attributes of the base classes (i.e., the similar seen classes), thereby overcoming negative transfer. Furthermore, PFSM creates feature representations and synthesizes the pseudo-features by using the base class features, the base vectors and the unseen class attributes. The outliers of pseudo-features are disposed of to get distinguishable pseudo-features.

Our major contributions are summarized as follows:

(1) We proposed a novel generalized zero-shot learning (GZSL) method of distinguishable pseudo-feature synthesis (DPFS). The proposed method can further improve GZSL classification performance compared with other state-of-the-art methods.
(2) We pretrained our model by a well-designed distance prediction loss while predicting prototypes for unseen classes, thereby avoiding overfitting.

(3) We only selected attributes of similar seen classes when making sparse representations based on attributes for unseen classes, thereby overcoming negative transfer effectively.

(4) We screened the outliers of synthesized pseudo-features and disposed of them to further improve class discriminability.

2. Related Works

Mapping-based approaches can be traced back to early ZSL tasks [2–4, 9]. They learn a mapping function between visual features and semantic features by supervised learning. So, it is important to construct a feature-semantic loss function that can be used to train mapping model [19]. But early methods are prone to overfitting in GZSL tasks [7]. CPL [8] learned visual prototype representations for unseen classes to solve the problem. To obtain discriminative prototype, DVBE [20] used second-order graphical statistics, DCC [21] learned the relationship between embedded features and visual features, and HSVA [22] used hierarchical two-step adaptive alignment of visual and semantic feature manifolds.

However, the prototype representation is constrained and adaptive alignment of visual and semantic feature manifolds. Visual features, and HSVA [22] used hierarchical two-step adaptive alignment of visual and semantic feature manifolds. However, the prototype representation is constrained and adaptive alignment of visual and semantic feature manifolds.

Generating-based approaches [23, 24], which utilize GANs and VAEs, have been widely applied to produce information about unseen classes and improve the prototype representation for GZSL tasks. They generate pseudo-features for unseen classes under the prior condition of semantic knowledge and random noise. LDMS [25], Inf-FG [26], and FREE [27] improved the generating strategy from aspects of discrimination loss, consistency descriptors, and feature refining. Besides, GCF [28] presented counterfactual-faithful generation to solve recognition rate imbalance between both seen classes and unseen ones. Although the strategies of generating-based methods are added to our proposed method, the use of simplex semantic information and the training difficulty [16] of GANs cause mode collapse.

Synthesis-based approaches [24, 29] integrate features and semantics of seen classes to enhance the feature diversity. SPF [15] designed a synthesis rule to guide feature embedding, TCN [14] exploited class similarities to build knowledge transfer from seen to unseen classes. To deal with the domain shift, LIUF [16] synthesized domain invariant features by minimizing the maximum mean discrepancy distance of seen classes of distance. However, it would lead to negative transfer by mixing irrelevant class information. Different from the above mentioned, we only select the similar seen classes, instead of all seen classes, to finish knowledge transfer, thereby avoiding negative transfer caused by the mixing of irrelevant information. Then, we utilize distinguishable features extracted from the pretrained embedding network to apply to the pseudo-feature synthesis. Besides, we use a preclassifier to dispose of the outliers of synthesized components, thereby improving class discriminability. Unlike the method [24] of using synthesized elements from other domains, we only utilize the similar seen classes from this domain to overcome the unavailability of data from other domains.

3. Proposed Method

GZSL is more challenging than ZSL, which recognizes samples only from unseen classes, because GZSL needs to recognize samples from seen classes and unseen classes. Therefore, we propose the DPFS method to improve the theoretical basis of GZSL further and boost the classification performance. DPFS can synthesize distinguishable pseudo-features for unseen classes, and then use the pseudo-features to finish GZSL classification together with features of seen classes. In this chapter, we first define notions and definitions of GZSL, then outline the proposed method, including base class selection, distinguishable feature extraction, attribute projection, and distinguishable pseudo-feature synthesis. Finally, we provide the process of our training algorithm.

3.1. Mathematical Formulation. In GZSL tasks, suppose we have $S$ seen classes $y^S$ and $U$ unseen classes $y^U$, $y^S \cap y^U = \emptyset$. We give training dataset $\Delta^S = \{y^S(x_i, y_j) \in \Xi \times y^S|_{i=1}^{n_s}\}$ where $n_s$ is the sample number, $\Xi$ is visual space, $x_i$ is a visual feature, and $y_j$ is the class index of $x_i$. The mapping function of the embedding network is denoted as $\varphi: \Xi \rightarrow \varsigma$ where $\varsigma$ is latent space. The weight parameters of the embedding network, the preclassifier and the classifier are $\theta^\text{en}_1, \theta^\text{cls}_1$, and $\theta^\text{cls}_2$, respectively. $A^S = [a^S_1, \ldots, a^S_n]$ and $A^U = [a^U_1, \ldots, a^U_n]$ are class-attribute matrices of seen classes and unseen classes, respectively. $s$ and $u$ are indexes of seen classes and unseen classes, $s \in y^S$ and $u \in y^U$, respectively.

GZSL methods learn a function $f_{GZSL}: \Xi \rightarrow y^S \cup y^U$ with training dataset $\Delta^S$, and class-attribute matrices $A^S$ and $A^U$ to classify disjoint seen classes and unseen ones at the same time. After the training, both seen and unseen classes from testing datasets will be predicted by $f_{GZSL}$.

3.2. Base Class Selection. For each unseen class, we only select the top $K$ seen classes similar to the unseen classes to overcome negative transfer. Attributes of all base classes of unseen class $u$ are with the closest distance to the attribute of the unseen class, which are as follows:

$$B_u = \left\{1 - \frac{a^U_u \cdot a^S_s}{\|a^U_u\| \|a^S_s\|} | s \in y^S \right\},$$  \hspace{1cm} (1)

$$y_u^B = \left\{\text{arg}_k \left\{1 - \frac{a^U_u \cdot a^S_s}{\|a^U_u\| \|a^S_s\|} \in \text{topk}(B_u) \right\} \right\},$$  \hspace{1cm} (2)
where $\text{top}^k(\bullet)$ is an operator that sorts elements from small to large and selects indices of the top $K$ elements. $y^B_u$ stones indices of the top $K$ base classes, which are the first to the $K$ th seen classes most similar to unseen class $u$.

### 3.3 Distinguishable Feature Extraction

In stage 1, we pre-train the embedding network and the preclassifier. It makes the embedding network extract distinguishable features for seen classes to build a relationship between classes and semantics, as shown in Figure 1. The attributes obtained by cognitive scientists [30] are the most commonly used semantic knowledge, and they are based on the high-level description of target objects specified by human beings [2].

We introduce the constraint of feature-attribute distance by imitating meta-learning [31], and build prototype representations, as shown in Figure 2. The customary way to construct the meta-learning task is called as $K$-way-$N$-shot [32], where $N$ labelled samples in each of the $K$ classes are provided in each iteration of the model training.

We randomly sample one unseen class and $K$ seen classes per iteration. And, we set support set $\Sigma = \{(x_i, y_i)|y_i \in \Psi^S\}_{i=1}^{N \times K}$ and query set $\Theta = \{(x_i, y_i)|y_i \in \Psi^U\}_{i=N \times K+1}^{N \times K+Q \times K}$. The visual features from $\Sigma$ produce prototypes for seen classes through the embedding network are as follows:

$$ c_{y_i} = \frac{\sum_{i=1}^{N} E_{\theta_m}(x_i)}{N}, $$

where $x_i$ is a visual feature from seen class $s$ and $N$ is the class number. Then, a feature-attribute distance (FAD) loss is constructed as follows:

$$ \Lambda_{\text{FAD}} = \sum_{(x_i, y_i) \in \Theta} \left( \|E_{\theta_m}(x_i) - c_{y_i}\|_2^2 + \|c_{y_i} - a_{y_i}\|_2^2 \right). $$

Different from the meta-representation [33] restrained by the distance minimization of intraclass features, we act on the feature-attribute distance constraint to structure the meta-representation associating common characteristics between different attributes. After the constraint, features in latent space are pulled near their prototypes to ensure that the similar attracts and the dissimilarity repels each other. The prototype and the attribute from the same class are close to each other. Therefore, the features of seen classes in latent space can be regarded as the distinguishable features extracted from the embedding network.

To keep the embedding network from overfitting, the prototypes are predicted by features of their base classes. A component from the base class is denoted as follows:

$$ v_k = E_{\theta_m}(\text{choice}(b_k)). $$
where choice (∗) is a choice operator, specifically choice (b_k) means randomly choosing a visual feature of the k th similar base class from \( \Psi^u \). A predicted prototype is denoted as follows:

\[
\tilde{c}_u = \frac{\sum_{h \in \Psi^u} u^{\Psi^u} v_h + a_u^U}{K + 1}.
\]

For each iteration, we build a prototype query set \( \Theta^U = \{ (\tilde{c}_u, u) \} \). Then, a preclassification loss \( \Lambda_{PC} \) operating to pretrain the preclassifier is donated as follows:

\[
\Lambda_{PC} = \sum_{(x, y) \in \Theta} \log \left( P_{\theta_{PC}}(y \mid E_{\Theta}(x)) \right) - \sum_{(\tilde{c}_u, u) \in \Theta^U} \log \left( P_{\theta_{PC}}(u \mid \tilde{c}_u) \right),
\]

where \( P(\cdot|\cdot) \) is a SoftMax function for the preclassification. Then, \( \Lambda_{FAD} \) and \( \Lambda_{PC} \) are summed to form distance prediction loss \( \Lambda_{DP} \) as follows:

\[
\Lambda_{DP} = \Lambda_{FAD} + \Lambda_{PC}.
\]

We use the distance prediction loss to jointly pretrain the embedding network and the preclassifier. After that, seen classes will be classified, and unseen classes will be predicted preliminarily. It prevents trade-off failure between seen and unseen classes. Besides, features of seen classes will be extracted, and then used for unseen pseudo-feature synthesis.

\subsection{3.4. Attribute Projection.}

Inspired by sparse coding, we make a sparse representation for each unseen class. We select attributes only from the base classes unlike the methods [14, 16] using all seen classes, to build attribute projections from seen to unseen classes. For unseen class \( u \), the matrix of its attribute projection is denoted as follows:

\[
M_u = \begin{bmatrix} a_{h_1}^S & \cdots & a_{h_K}^S \end{bmatrix},
\]

where \( a_{h_1}^S, a_{h_K}^S \in \Psi^u \). The attribute projection can represent the unseen class information by using sparse representation vector set \( \{ m_u \}_{u=1}^{U} \). The objective function of the attribute projection is as follows:

\[
m_u = \arg \min_{m_u} \| a_u^U - M_u m_u \|_2^2 + \beta_1 \| m_u \|_1 + \beta_2 \| m_u \|_2^2,
\]

where \( \beta_1 \) and \( \beta_2 \) are two regulation coefficients, \( \beta_1, \beta_2 > 0 \). The mixed regularizations of \( L1 \)-norm and \( L2 \)-norm have the advantages of sparsity and trade-off between deviation and variance [34]. Both \( \beta_1 \) and \( \beta_2 \) are set as 0.4 with appropriate generality. The objective function is optimized by the optimal local condition of Karush-Kuhn-Tucker [35] where \( m_u \) are non-negative. We normalize \( m_u \) by using the following equation:

\[
m_u = \frac{m_u}{\| m_u \|}.
\]
Then, we treat \( m_u \) as the base vector. The attribute projection provides a vital item for the pseudo-feature synthesis, as shown in Figure 3.

3.5. Distinguishable Pseudo-Feature Synthesis. For unseen class \( u \), we randomly choose a feature from each of its base classes to construct an embedding matrix \([ v_1 \cdots v_K]\). The base vectors are utilized for weighting the chosen features that are embedded into the attribute projection, as shown in Figure 3(a). Then, a feature representation is formulated as follows:

\[
\overline{v} = (1 - \gamma) [v_1 \cdots v_K] m_u + \gamma a_u^U,
\]

(12)

where \( \gamma \) is a weighting coefficient (\( \gamma \in [0, 1] \)). However, the feature representation only integrated with features of the base classes may be scattered and produce outliers of candidate pseudo-features, as shown in Figure 3(b). Therefore, attribute information is integrated into the feature representation to synthesize candidate pseudo-features, as shown in Figure 3(c).

To dispose of the outliers, we screen them by the following equation:

\[
f(\overline{v}) = \begin{cases} 1, & \text{otherwise}, \\
0, & \max_{s \in \Psi} P_{\theta_{\text{pre}}} (s|\overline{v}) \geq \tau,
\end{cases}
\]

(13)

where \( \tau \) is a creditability threshold (\( \tau \in [0, 1] \)). The preclassifier acts as an operator of the outlier disposing. It screens and reserves the credible pseudo-features satisfying \( f(\overline{v}) = 1 \) to get distinguishable pseudo-features of unseen classes, as shown in Figure 3(d). After the operations of the attribute projection and the pseudo-feature synthesis, the synthesized features integrated with the information of the similar base classes and unseen classes have separability characteristics.

3.6. Train and Inference. We conduct the DPFS model training. Algorithm 1 shows the pseudo-code of the DPFS training algorithm. The algorithm mainly includes two-cycle structures because DPFS is a two-stage method. Firstly, the sequence structure from lines 1 to 2 performs the attribute projection to get the base vector for each unseen class. Next, the first cycle from lines 3 to 9 performs the embedding module pretraining to extract distinguishable features of seen classes. Then, the second cycle from lines 10 to 15 performs the classifier training for GZSL tasks. In each iteration of the classifier training, we randomly select a certain number of the whole samples from training samples and synthesized pseudo-feature samples, where the number of the selected whole samples is \( N_w \). Here, the proportion of the pseudo-feature samples in the whole samples is set as \( \eta \). After each iteration, the classifier is adopted for evaluation.

4. Experimental Results

4.1. Datasets. The DPFS model is evaluated on four widely datasets as evaluating benchmarks, i.e., Animals with Attributes 2 (AWA2 [6]), aPascal & Yahoo (aPY [36]), Caltech UCSD Birds 200 (CUB [37]), and SUN Attribute (SUN [38]). AWA2 and aPY are coarse-grained datasets and aPY includes a higher proportion of unseen classes than AWA2. CUB and SUN are fine-grained datasets, especially SUN, with more whole classes and fewer training samples per class than CUB. Table 1 summarizes the statistics of the four evaluating benchmarks.

4.2. Implementation Details. We conduct ResNet-101 [39] as a backbone based on a convolutional neural network. Visual features are extracted from the output of the final avg-pooling layer after the backbone is pretrained on ImageNet [1]. Figure 4 shows the network structures of the DPFS model including the embedding network, the preclassifier and the classifier. The embedding network is composed of three fully connected (FC) layers, and the back of each layer is connected to a ReLU activation function for nonlinear activation. Both the preclassifier and the classifier have the same modules. Their modules are composed of two FC layers and the output dimensions equal the total number of all classes. For the four benchmarks, the middle layer dimension of the classifier is 512 for AWA2 and aPY, and 1024 for CUB and SUN, respectively.

Our model is coded in PyTorch and runs on GeForce RTX 2080 Ti. It is trained by an adaptive moment estimation (Adam) [40] optimizer. During the embedding module pretraining, sample numbers of each class in both the support set and the query set, \( N \) and \( Q \), are set as 4 for AWA2, aPY, and CUB, and 2 for SUN, respectively. The learning rate of our model is \( 10^{-4} \). During the classifier training, the number of the whole selected samples, \( N_w \), is set as 1000. The classifier is trained with a learning rate of \( 10^{-4} \) and the embedding module is fine-tuned with a learning rate of \( 10^{-6} \). Besides, four additional hyper-parameters, the proportion of pseudo-feature samples \( \eta \), creditability threshold \( \tau \), number of base classes \( K \), and weighting coefficient \( \gamma \) will be discussed later in the hyper-parameter sensitivity chapter. Samples from training datasets are used to train our model by supervised learning. And samples from the testing datasets are used to evaluate GZSL classification performance of our model.

The accuracies of average seen classes (\( As \)) and average unseen classes (\( Au \)) are computed based on the universal evaluation protocols [6].

\[
As = \frac{1}{|\Psi|} \sum_{y \in \Psi} \frac{\# \text{correct pre d i c t i o n s in } y}{\# \text{samples in } y},
\]

(14)

\[
Au = \frac{1}{|\Psi^U|} \sum_{y \in \Psi^U} \frac{\# \text{correct pre d i c t i o n s in } y}{\# \text{samples in } y}.
\]

(15)

We evaluate the simultaneous classification accuracy of both seen and unseen classes by computing harmonic mean \( H \) as follows:

\[
H = \frac{2 \times As \times Au}{As + Au},
\]

(16)

\( H \) is regarded as the most crucial criterion to measure the GZSL classification performance.
4.3. Hyper-Parameter Sensitivity. There are four hyper-parameters including the proportion of pseudo-feature samples $\eta$, creditability threshold $\tau$, number of base classes $K$, and weighting coefficient $c$. We discuss the sensitivity of the hyper-parameters because proper hyper-parameters give our model extra reliability and robustness.

Proportion $\eta$ controls the frequencies of obtaining information from seen classes and unseen ones. Higher $\eta$ provides the classifier with more opportunities to learn the characteristics of unseen classes. Figure 5 shows GZSL classification performance under different $\eta$ on the four benchmarks. We set $\eta$ within the range from 0.7 to 0.97 and select the proper $\eta$ value according to the optimal GZSL performance.

As will decrease slowly while $Au$ and $H$ will increase until reaching a peak along with the increase of $\eta$ in most cases. This result reveals that DPFS can provide more balanced GZSL performance by adjusting $\eta$. The decreasing ratio of $As$ will increase after $Au$ and $H$ reach the peak. It indicates a proper selection of $\eta$ is necessary to solidify seen class classification. When $H$ reaches the peak, $\eta$ is different on the four benchmarks. The value depends on the
granularity of training samples. In general, the value on the benchmarks with a few training samples (such as SUN) should be lower than that on the benchmarks with multitraining samples (such as AWA2), and the value on the benchmarks with a higher proportion of unseen classes (such as aPY and CUB) should be higher. Therefore, we set $\eta = 0.85$ for AWA2, $\eta = 0.94$ for aPY, $\eta = 0.91$ for CUB, and $\eta = 0.76$ for SUN.

Creditability threshold $\tau$ controls the effect of the outlier disposing. Figure 6 shows the performance under different $\tau$ on the four benchmarks. We set $\tau$ within the range of 0.7 to 0.95. This result reveals that $As$ will decrease and $Au$ will increase along with the increase of $\tau$ in most cases. Meanwhile, $H$ will increase until reaching a peak. When the range of $\tau$ is 0.8 to 0.9, $H$ will reach the peak, and the classification accuracy will be the best. It indicates proper $\tau$ can prevent the outliers from interfering with seen class classification while maintaining unseen class classification. Therefore, we set $\tau = 0.85$ on all the four benchmarks.

Numbers of base classes and weighting coefficient, $K$ and $y$, concurrently control the pseudo-feature synthesis simultaneously. Figure 7 shows the $H$ heatmap results of the
performance under different $K$ and $c$ values on the four benchmarks. The range of $K$ is set from 3 to 9 for AWA2 and CUB, from 6 to 12 for aPY, and from 2 to 8 for SUN, respectively. The range of $c$ is set from 0 to 0.4.

The result reveals that $N$ has a more significant impact than $c$ on $H$. $H$ will increase first and then reduce along with the increase of $N$. It indicates that an appropriate integration with the similar seen classes will achieve outstanding classification accuracy, but an over-integration will degrade the classification accuracy because it mixes information of irrelevant classes. According to the performance on the four benchmarks, $N$ making $H$ reach peak depends on the granularity of training samples. In general, $N$ on the benchmarks with a few training samples (such as CUB and SUN) should be lower than that on the benchmarks with multitraining samples (such as AWA2), and $N$ on the benchmarks with the higher proportion of unseen classes (such as aPY) should be higher. So, we set $N = 5$ for AWA2, $N = 9$ for aPY, $N = 6$ for CUB, and $N = 3$ for SUN.

The result also reveals that when $N$ is fixed, $H$ will also increase first and then reduce along with the increase of $c$ in most cases. It indicates that weighting a certain proportion of attributes will improve the classification accuracy and the proper introduction of attribute information can raise the performance of our model. Therefore, we set $c = 0.2$ for AWA2, $c = 0.3$ for aPY, $c = 0.1$ for CUB, and $c = 0.35$ for SUN.

Figure 6: Accuracy of classification under different $\tau$ on (a) AWA2, (b) aPY, (c) CUB, and (d) SUN.
results show that DPFS gains the best performance on AWA2, aPY, CUB, and SUN, and achieves the second performance on CUB. Compared with the mapping-based approaches, DPFS is superior to DCC by 5.5% on aPY, and DVBE by 4.8%, 2%, and 5.4% on AWA2, CUB, and SUN, respectively. Compared with the generating-based approaches, DPFS is superior to FREE by 4.7% on AWA2, LDMS by 4.9% on aPY, and GCF by 3.9% on SUN, respectively. And compared with the synthesis-based approaches, DPFS is superior to LIUF by 1.6%, 1.1%, 6%, and 3.2% on AWA2, aPY, CUB, and SUN, respectively. DPFS significantly improves Au and avoids overfitting.

Figure 7: Four H heatmaps under different N and γ on (a) AWA2, (b) aPY, (c) CUB, and (d) SUN. Each row and each column denote the results of different N and γ, respectively. In each heatmap, brighter colour represents greater H.

Table 1: Statistics of the four benchmark datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of classes</th>
<th>Attribute</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seen</td>
<td>Unseen</td>
<td>Total</td>
</tr>
<tr>
<td>AWA2</td>
<td>40</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>APY</td>
<td>20</td>
<td>12</td>
<td>32</td>
</tr>
<tr>
<td>CUB</td>
<td>150</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>SUN</td>
<td>645</td>
<td>72</td>
<td>717</td>
</tr>
</tbody>
</table>

4.4. Performance Results. Table 2 shows GZSL classification performance results compared with existing state-of-the-art approaches and the proposed DPFS. The existing approaches contain the mapping-based, the generating-based, and the synthesis-based approaches, which are marked with †, ‡, and §, respectively. Among these, the results show that DPFS gains the best performance on AWA2, CUB, and SUN, and achieves the second performance on CUB.
DPFS is superior to most mapping-based approaches in the aspects of $A_u$ and $H$, especially on SUN. It indicates that DPFS has a more vital learning ability on the benchmarks with a few training samples. And DPFS shows significant improvement of $A_pl$, $A_u$, and $H$, especially compared to generating-based approaches on aPY. It explains that DPFS makes full use of the feature information of seen classes and the attribute information, thereby solving the difficulty of classifying the higher proportion of unseen classes and avoiding mode collapse.

DPFS also experiments on the four benchmarks for conventional ZSL tasks, where only the synthesized pseudo-features are fed into the classifier. Table 3 shows ZSL classification performance results. We observe that DPFS outperforms existing methods on AWA2, aPY, and SUN, which can also verify that the synthesized pseudo-features have distinguishable characteristics.

We further demonstrate the advantage of DPFS over SPF and LIUF. We imitate SPF and LIUF, replacing the strategy of our pseudo-feature synthesis with the strategy synthesis of SPF and LIUF to form the reference methods, D-SPF and D-LIUF, respectively. Meanwhile, the stages of the embedding module pretraining and classifier training of D-SPF and D-LIUF are the same as those of DPFS. Table 4 shows the comparison results among D-SPF, D-LIUF, and DPFS. DPFS gains prominent advantages over D-SPF because the optimized attribute projection can embed and project features of seen class into features of unseen class more accurately, to improve class discriminability.

4.5. Ablation Results. We conducted ablative experiments to illustrate the influence of different tactics in DPFS. The tactics contain the embedding module pretraining (mpt), the outlier disposing (odi) in equation (13), and the preclassification loss (pc) in equation (7). Table 5 shows the results of ablation experiments. Four ablated methods, PFS, DPFS-1, DPFS-2, and DPFS-3 are all validated. PFS is to remove all the tactics. DPFS-1, which pretrains the model only by the feature-attribute distance loss in equation (4), is to add the mpt tactic. DPFS-2 is to add both the mpt and odi tactics. And DPFS-3, which pretrains the model by the distance prediction loss in equation (8), is to add both the tactics of mpt and pc.

It is important to add the mpt tactic for extracting some common characteristics between seen classes and unseen ones because it improves prototype representations and eliminates the domain shift. Therefore, DPFS-1 performs obvious progress compared with PFS. PFS-1 is superior to PFS by 8.6% on AWA2, 8.3% on aPY, 9.3% on CUB, and 9.3% on SUN. On this foundation, DPFS-2 adopts the odi tactic to eliminate the outliers of candidate pseudo-features. It boosts the performance on parts of benchmarks. PFS-2 is superior to PFS-1 by 0.9% on AWA2, and 0.4% on aPY.
respectively. DPFS-3 adopts the pc tactic to predict prototypes for unseen classes before the classifier training, thus improving the classification performance. PFS-3 is superior to PFS-1 by 2.9% on AWA2, 6.6% on aPY, 1.5% on CUB, and 2.4% on SUN, respectively. DPFS can cohere all the features in the same class and therefore avoid outlier interference. Thus, DPFS adopting the three auxiliary tactics at the same time makes the best progress in $H$ on the four benchmarks. And, DPFS is superior to PFS-3 by 1.7% on AWA2, 2% on aPY, 1.7% on CUB, and 2.6% on SUN.

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

We visualize features from the embedding module by t-SNE [41] to further show the tactic effect on the AWA2 benchmark for GZSL tasks. Figure 8 shows the visualization results. We find that DPFS can improve the distinguishability of unseen classes. Meanwhile, it can also maintain the distinguishability of seen classes according to the comparison results between Figures 8(a), 8(c) and 8(b), and 8(d). Considering that existing methods [18, 26] do not visualize all features of both seen and unseen classes, we visualize all the output features of testing samples from PFS and DPFS in

Table 4: Quantitative comparisons among D-SPF, D-LIUF, and DPFS.

<table>
<thead>
<tr>
<th>Method</th>
<th>AWA2</th>
<th>aPY</th>
<th>CUB</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-SPF</td>
<td>81.3</td>
<td>55.4</td>
<td>65.9</td>
<td>32.9</td>
</tr>
<tr>
<td>D-LIUF</td>
<td>86.1</td>
<td>60.4</td>
<td>71.0</td>
<td>36.2</td>
</tr>
<tr>
<td>DPFS</td>
<td>87.3</td>
<td>61.0</td>
<td>71.8</td>
<td>38.6</td>
</tr>
</tbody>
</table>

Table 5: Ablation results on DPFS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mpt</th>
<th>Odi</th>
<th>Pc</th>
<th>AWA2</th>
<th>aPY</th>
<th>CUB</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFS</td>
<td></td>
<td></td>
<td></td>
<td>58.6</td>
<td>35.8</td>
<td>46.0</td>
<td>31.8</td>
</tr>
<tr>
<td>DPFS-1</td>
<td>√</td>
<td></td>
<td></td>
<td>67.2(+8.6)</td>
<td>44.1(+8.3)</td>
<td>55.3(+9.3)</td>
<td>41.1(+9.3)</td>
</tr>
<tr>
<td>DPFS-2</td>
<td>√</td>
<td>√</td>
<td></td>
<td>68.1(+9.5)</td>
<td>44.5(+8.7)</td>
<td>55.2(+9.2)</td>
<td>40.8(+9.0)</td>
</tr>
<tr>
<td>DPFS-3</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>70.1(+11.5)</td>
<td>50.7(+14.9)</td>
<td>56.8(+10.8)</td>
<td>43.5(+11.7)</td>
</tr>
<tr>
<td>DPFS</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>71.8(+13.2)</td>
<td>52.7(+16.9)</td>
<td>58.5(+12.5)</td>
<td>46.1(+14.3)</td>
</tr>
</tbody>
</table>

Figure 8: T-SNE visualization of features in 40 seen classes (a, b), 10 unseen classes (c, d) and all 50 classes (e, f) on AWA2 by PFS (a, c, e) and DPFS (b, d, f) in GZSL tasks. Different colours denote different classes. It is obvious that DPFS can provide more separable classes.
Figures 8(e) and 8(f), respectively. It is obvious that the classes characterized by the output features from DPFS is more separable than those characterized by the output features from the PFS. DPFS eliminates the confusion between classes and improves feature distinguishability, thus achieving a better multiclass classification accuracy. Both seen and unseen classes satisfy the characteristics of intra-class gather and interclass separability. Therefore, DPFS can effectively eliminate the domain shift.

5. Discussion

Based on the results above, our model was trained and evaluated on four benchmark datasets. Our method selected the optimal hyper-parameters for different benchmarks to achieve the best GZSL classification performance compared with most existing methods. Especially on the benchmarks with a few training samples or with a higher proportion of unseen classes, DPFS gained the superior performance because it can use the information of features and attributes appropriately and avoid mode collapse. Compared with existing synthesis-based models similar to DPFS, DPFS can eliminate the introduction of irrelevant classes and suppress negative transfer. It can also synthesize candidate pseudo-features and dispose of the outliers to improve class discriminability.

Furthermore, our model was also trained and evaluated for ZSL tasks and outperformed competing ZSL methods on most benchmarks. Besides, we conducted the ablation experiments of DPFS and further explained the performance gain of each tactic. Distinguishable features can be extracted and the GZSL performance can be improved with the embedding module pretraining tactic. On this basis, adding the preclassification tactic can predict prototypes for unseen classes before the classifier training, thereby improving the performance and avoiding overfitting. The tactic of the outlier disposing can further enhance the performance. These are the foundation that outperforms the competing GZSL methods. The visualization results have demonstrated that DPFS has the distinguishability characteristics of both seen and unseen classes.

6. Conclusion

This paper proposed a novel distinguishable pseudo-feature synthesis (DPFS) method for GZSL tasks. It included the procedures of base class selection, distinguishable feature extraction, attribute projection, feature representations, and outlier disposing. These procedures can realize the initialization, the connection, and the weight updating of the DPFS model. Therefore, the model can synthesize distinguishable pseudo-features with attributes of unseen classes and features of similar seen classes. Experimental results showed that DPFS achieved the GZSL classification performance better than existing methods. It indicated DPFS significantly improved class discriminability and restrained negative transfer, and DPFS also effectively eliminated the domain shift and the confusion between classes. In the future, we will synthesize more distinguishable features of unseen classes by integrating more auxiliary information, such as statistical features and knowledge graphs, to extend our method into other applications.

Data Availability

The dataset AWA2 can be downloaded from https://cmli.st.ac.at/Awa2/ or https://academictorrents.com/details/1490ae815141cdb50a32b81ef78b1eaf6b38b03. The other three datasets, aPY, CUB, and SUN can also be downloaded from https://vision.cs.uiuc.edu/attributes/, http://www.vision.caltech.edu/datasets/cub_200_2011/, and https://www.cnblogs.com/GarfieldEr007/p/5438417.html, respectively.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 42276187 and 41876100) and the Fundamental Research Funds for the Central Universities (Grant no. 3072022FSC0401).

References


