Graph Knows Unknowns: Reformulate Zero-Shot Learning as Sample-Level Graph Recognition

Jingcai Guo[1,2], Song Guo[1,2], Qihua Zhou[1], Ziming Liu[1], Xiaocheng Lu[1,3], Fushuo Huo[1]

[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China
[2]The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen, China
[3]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
{jc-jingcai.guo, song.guo}@polyu.edu.hk, {qi-hua.zhou, ziming.liu, xiaocheng.lu, fushuo.huo}@connect.polyu.hk

Abstract

Zero-shot learning (ZSL) is an extreme case of transfer learning that aims to recognize samples (e.g., images) of unseen classes relying on a train-set covering only seen classes and a set of auxiliary knowledge (e.g., semantic descriptors). Existing methods usually resort to constructing a *visual-to-semantics* mapping based on features extracted from each whole sample. However, since the visual and semantic spaces are inherently independent and may exist in different manifolds, these methods may easily suffer from the *domain bias* problem due to the knowledge transfer from seen to unseen classes. Unlike existing works, this paper investigates the fine-grained ZSL from a novel perspective of *sample-level graph*. Specifically, we decompose an input into several fine-grained elements and construct a graph structure per sample to measure and utilize *element-granularity relations* within each sample. Taking advantage of recently developed graph neural networks (GNNs), we formulate the ZSL problem to a *graph-to-semantics* mapping task, which can better exploit element-semantics correlation and local sub-structural information in samples. Experimental results on the widely used benchmark datasets demonstrate that the proposed method can mitigate the domain bias problem and achieve competitive performance against other representative methods.

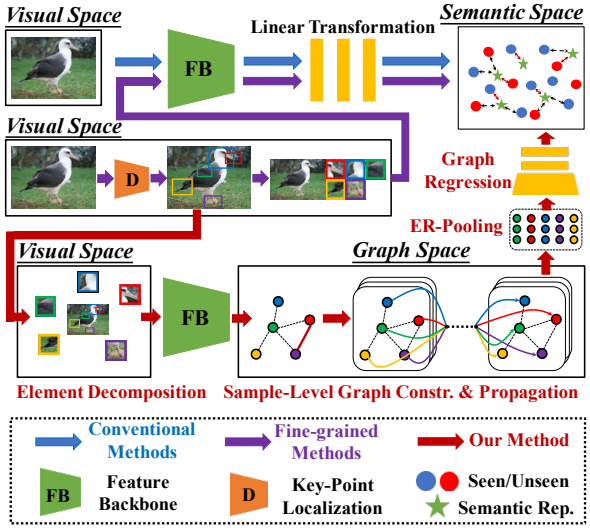


Figure 1: A comparison of our method (red stream) with conventional ZSL methods (blue stream) and fine-grained ZSL methods (purple stream, zoom in for better view).

Introduction

Recent years have seen a rise of interest in zero-shot learning (ZSL) which imitates the human ability to recognize unseen classes without observing real samples (Kodirov, Xiang, and Gong 2017; Yu et al. 2018; Guo and Guo 2019; Zhu et al. 2019; Chen et al. 2021c; Liu et al. 2022; Kim, Shim, and Shim 2022; Su et al. 2022; Khan et al. 2023). Specifically, ZSL takes utilization of seen classes with labeled samples and auxiliary knowledge between seen and unseen classes to achieve recognition. This knowledge, e.g., semantic descriptions that exist in a high-dimensional feature space, can represent meaningful high-level and per-class information about samples. In ZSL, the common practice is to map an unseen class sample from its original feature space, e.g., visual space, to the semantic space by reusing a mapping function trained on seen classes. With such mapped semantic features (representation), we can then search for the most closely related (e.g., by similarity metrics) description whose corresponding class is assigned to the sample. In practice, the ZLS can be further restricted to the classic ZSL and generalized ZSL (GZSL), where the former only considers the recognition of unseen classes during inference, while the latter also classifies samples from seen classes.

In ZSL, one non-negligible issue is that the visual and semantic spaces are inherently independent and may exist in entirely different manifolds. This issue brings about the domain bias problem (Fu et al. 2015) when generalizing the learned knowledge from seen to unseen classes, i.e., the mapped semantic features of unseen class samples may be biased to seen classes. Recently, several works have been proposed to mitigate the domain bias problem and obtained some promising results. For example, some methods consider doing the alignment between visual-semantic spaces when constructing the mapping function (Zhang and Saligrama 2015; Schonfeld et al. 2019; Guo and Guo 2020). Differently, some methods try to synthesize unseen class samples or features based on generative models, e.g., GANs

or VAEs, and involve them in the training process (Huang et al. 2019; Zhao et al. 2022; Feng et al. 2022). Some other methods apply the encoder-decoder structure to maintain the robustness of the mapping function (Kodirov, Xiang, and Gong 2017; Yu et al. 2018). Most recently, several fine-grained methods (Xie et al. 2019; Huynh and Elhamifar 2020) concatenate the features of the whole sample with local regions to enhance the representations, while the relation among local regions is not addressed.

As far as we know, existing methods usually train the mapping functions with either whole sample features or concatenated fine-grained features. Such a training scheme may ignore the subtle element-semantics correlation and local sub-structural information in samples from different classes. Concretely, our motivations are two-fold: ❶ Taking the seagull as an example (Figure 1), it may have some special elements such as *beak*, *nape*, *feet*, etc. These elements are more accurate visual characteristics of bird species and can usually correspond '*one-to-one*' with the semantic description like *yellow-beak*, *white-nape*, *gray-tail*, etc. Such element-semantics correlation can potentially facilitate the recognition. ❷ Certain connectivity exists among these elements which establish unique *topological structures* such as the relative position/distance of each fine-grained element. Moreover, these elements may have some *mutual influences*, e.g., the *beak-feet* pair usually has stronger connection and co-occurrence. Thus, a properly adopted *propagation* can potentially enhance the representation of each other.

In this paper, we suggest that the above intrinsic properties are big pluses for the more accurate and robust visual-semantics mapping, and for the first time, we propose a novel *sample-level graph-based ZSL framework* with improved performance. Specifically, we first decompose an input into several fine-grained elements, i.e., via key-point localization and cropping, and then convert it into a sample-level graph considering its *topological structure* and *mutual influences* between elements. Regarding the graph variables, we use the nodes embedded with visual features to present each element of the sample, and use the linking edges to present whether a relation exists between two elements. To determine the edges, we further design a *pseudo-link and propagate* verification to identify the mutual influence among elements. Afterward, we build upon the graph neural networks (GNNs) to extract and fuse the local sub-structural information among elements residing in each sample-level graph, and further formulate the ZSL problem to a *graph-to-semantics mapping* task for better preservation of the one-to-one element-semantics correlation.

In summary, our contributions are three-fold:

- We first utilize the graph structure to model the samples in ZSL, and explore the element-semantics correlation and local sub-structure information to construct more accurate and robust ZSL mapping.
- We reformulate the ZSL to a graph-to-semantics mapping task and convert the recognition into the sample-level graph classification as an alternative for ZSL.
- Experimental results on ZSL and GZSL tasks demonstrate that the proposed method can outperform other representative methods and verify its effectiveness.

Related Work

Zero-Shot Learning

Most ZSL methods are implemented by mapping the visual features to semantic space spanned by class descriptions and then performing nearest neighbor search (Akata et al. 2015; Schonfeld et al. 2019). In contrast, some methods propose to map the semantic features into visual space and point out that using semantic space as shared latent space may reduce the variance of features (Zhang, Xiang, and Gong 2017). Different from the direct mappings, a few branches of them try to learn a metric network or compatibility function that takes paired visual and semantic features as inputs and calculates their similarities (Sung et al. 2018). [*Summary*]: our mapping function is similar to the first one, while we convert the visual input into the sample-level graph and construct a graph-to-semantics mapping to achieve the ZSL recognition.

Compared with the above conventional methods, our model is more likely related to the fine-grained ZSL and semantics-level graph-based ZSL. Specifically, the fine-grained ZSL tries to make use of the concatenation of global and local features (Ji et al. 2018; Xie et al. 2019; Huynh and Elhamifar 2020), or learn dictionaries through joint training with samples, attributes, and labels (Chen, Cao, and Ji 2019), to enhance the representation. Recently, another branch of fine-grained method (Xie et al. 2020) extends the graph embedding into the global and local concatenation to further enhance the representation, which is very similar to (Ji et al. 2018; Zhu et al. 2019). Differently, the semantics-level graph-based ZSL utilizes the WordNet (Miller 1995) to link each per-class semantic description and then model the class-wise correlation as a global semantics-level graph, which can better capture the dependencies among classes (Wang, Ye, and Gupta 2018; Kampffmeyer et al. 2019). [*Summary*]: our method differs in two aspects. First, we only use local features rather than the concatenation with also global features. Second and more importantly, we model the inputs as sample-level graphs and convert the ZSL problem into a graph-to-semantics mapping task. Thus, our method is complementary to the semantics-level graph-based ZSL.

GNNs for Graph Recognition

The GNNs are popular graph techniques in deep learning that attracts increasing attention most recently. In practice, the GNNs can be trained in a supervised or unsupervised manner to handle multiple tasks such as node classification, edge prediction, graph embedding, and graph classification. Specifically, the graph classification aims at classifying an entire graph structure into different classes (Zhang et al. 2018; Ying et al. 2018), which has been widely used in some real-world applications like community recognition, documents categorization, social network analysis, drug discovery, and so on. In this paper, we extend the graph classification to ZSL domain, where we replace the learning targets with class semantic descriptions to form a graph-level regression process.

Methodology

Problem Definition

Given a train-set of seen class $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, where x_i is the input sample with class label y_i belonging to m seen classes $C = \{c_1, c_2, \cdots, c_m\}$. The goal of ZSL is to construct a model for a set of unseen classes $C' = \{c'_1, c'_2, \cdots, c'_v\}$ ($C \cap C' = \phi$), of which no sample is available. During inference, given unseen class sample x', the model is expected to predict its class $c(x') \in C'$. To this end, some auxiliary knowledge, e.g., the semantic descriptions denoted as $s = (a_1, a_2, \cdots, a_n) \in \mathbb{R}^n$, is required to bridge the gaps between seen/unseen classes. The train-set can then be further specified as $\mathcal{D} = \{x_i, y_i, s_i\}_{i=1}^{N}$, and each seen/unseen class c_i / $c_i{}'$ is endowed with a semantic prototype p_{c_i} / $p_{c_i{}'} \in \mathbb{R}^n$. Thus, for each seen class sample we have its semantic features $s_i \in P = \{p_{c_1}, p_{c_2}, \cdots, p_{c_m}\}$, while for test unseen class sample x', we need to obtain its semantic features $s' \in \mathbb{R}^n$, and set the class label by searching for the most closely related semantic prototype within $P' = \{p_{c_1{}'}, p_{c_2{}'}, \cdots, p_{c_v{}'}\}$ for ZSL or within $P' \cup P$ for GZSL. In summary, the training can be described as:

$$\arg\min_{\mathbf{W}} \frac{1}{N} \cdot \sum_{i=1}^{N} \mathcal{L}\left(f\left(\phi\left(x_i\right); \mathbf{W}\right), s_i\right) + \varphi\left(\mathbf{W}\right), \quad (1)$$

where $\mathcal{L}(\cdot)$ is the loss function and $\varphi(\cdot)$ denotes the regularization term. The $f(\cdot; \mathbf{W})$ is a mapping with trainable parameter $\mathbf{W}$ that maps samples from visual space to semantic space. $\phi(\cdot)$ denotes a feature extractor, e.g., a CNNs backbone. During inference, given a test sample x_{test}, the recognition can be described as:

$$\arg\max_{j} \; Sim\left(f\left(\phi\left(x_{test}\right); \mathbf{W}\right), P'^{(j)}\right), \quad (2)$$

$$\arg\max_{j} \; Sim\left(f\left(\phi\left(x_{test}\right); \mathbf{W}\right), \{P' \cup P\}^{(j)}\right), \quad (3)$$

where $Sim(\cdot)$ is a similarity metric. Eq. (2) is used for the ZSL task where the similarity search is limited in unseen classes, and Eq. (3) is for the GZSL task where the search can generalize to novel samples from seen classes as well.

ZSL as Sample-Level Graph Recognition

Given train-set $\mathcal{D} = \{x_i, y_i, s_i\}_{i=1}^{N}$, our method has three steps to convert the ZSL task to a graph-to-semantics mapping problem: 1) elements decomposition, 2) sample-level graph construction, and 3) sample-level graph recognition. The elements decomposition obtains several fine-grained elements of a sample, which can then be presented by a well-designed graph structure. We feed these per-sample graphs into our modified Element-Rank-Aware (ERA) GNNs that sequentially pass through graph convolutional layers, our modified element ranking pooling layers, and regression layers, to reach their semantic descriptions.

Element-Rank-Aware GNNs Our modified ERA-GNNs consist of three consecutive operations. First, the graph convolution layers are the same as standard graph convolutions that responsible for extracting high-level topological-wise element representations and mutual influences of samples. Then, we design the element ranking pooling (ER-Pooling) layers, which are sensitive to element-wise importance, to downsample and fuse high-level features to more representative graphical features. Last, the regression layers are convolutional and dense layers that link to semantic descriptions.

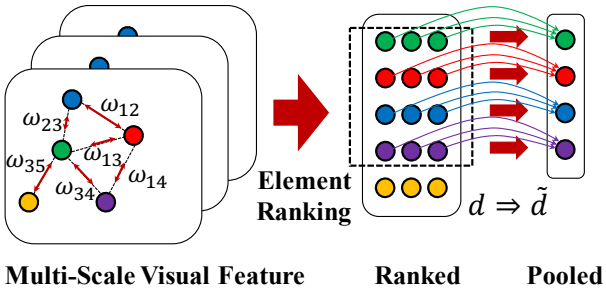


Figure 2: Element Ranking Pooling

Graph Convolutions - Suppose a sample-level graph $(\mathbf{A}, \mathbf{F})$ has been given, where $\mathbf{A} \in \{[0,1]\}^{n \times n}$ is the adjacency matrix, in which the non-zero values denote the correlation (edge) existence and weight between two elements in a sample. $\mathbf{F} \in \mathbb{R}^{n \times d}$ is the element feature matrix representing that the sample-level graph has n elements and each of them contains d-D features. The graph convolution operation in our method can be described as:

$$\mathbf{H} = \delta\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{F}\mathbf{\Theta}\right), \quad (4)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\mathbf{I}$ is the identity matrix denotes a self-loop in each element of the sample-level graph. $\tilde{\mathbf{D}} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the diagonal degree matrix, and $\mathbf{\Theta} \in \mathbb{R}^{d \times d'}$ is the trainable parameter matrix. $\delta(\cdot)$ is a nonlinear activation, e.g., ReLU. The graph convolutions can be decoupled into four processes. A linear transformation $\mathbf{F}\Theta$ is first performed which maps the element features from d to m channels into the next layer. Then, $\tilde{\mathbf{A}}\mathbf{F}\Theta$ propagates element information to neighboring elements. Next, $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ normalizes each row in the obtained feature matrix $\mathbf{H}$. The last nonlinear activation $\delta(\cdot)$ performs point-wise activation and outputs the graph convolution results. To further extract the deep high-level multi-scale features, we can stack multiple graph convolution layers, e.g., k layers, as:

$$\mathbf{H}^{(k)} = \delta\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k-1)}\mathbf{\Theta}^{(k-1)}\right), \quad (5)$$

where the obtain k sequential features $\left\{\mathbf{H}^{(i)}\right\}_{lr=1}^{k}$ can be further downsampled and regressed by our element ranking pooling layers and regression layers, respectively.

Element Ranking Pooling - Conventional graph poolings usually downsample the graph in a hard way due to its non-Euclidean structure (Gilmer et al. 2017; Ying et al. 2018), i.e., the reduced graph nodes are usually not organized and the calculation is usually mean/maximum value-based. Such a pooling strategy fits classic graph applications well since the structural information provides the major contribution to

their recognition. However, in our sample-level graph-based ZSL, the visual information is also important since the nodes explicitly correspond to key elements of a visual sample. Thus, it is crucial to properly organize the reduced elements to fit the downstream ZSL recognition.

Intuitively, the elements within a sample can have different visual importance. For example, the *back* and *beck* may contribute more than other key elements to recognize the seagull class. Moreover, we note that different layers can also contain rich multi-scale visual features, which are usually ignored in conventional graph pooling strategies. As demonstrated in Figure 2, we design the Element Ranking Pooling to integrate such visual information. Specifically, we use the element degrees and weights w.r.t. the linking edges to calculate the element importance score:

$$Score_i = \frac{1}{g} \cdot \sum_{j=1}^{g} \omega_{ij} \cdot RI_i, \tag{6}$$

where ω_{ij} denotes the edge-weight between element i and j in $\mathbf{A}$, and RI_i is the relative importance of element i w.r.t. the recognition of whole sample. Note ω_{ij} and RI_i can be obtained during the sample-level construction in Sect. *Sample-Level Graph Construction*. The parameter g is the degree of element i, which can be easily obtained by $g = \text{Sum}(\mathbf{A})_i$. Next, we rank all elements according to their scores and select the highest p elements to calculate its linear transformation, i.e., denoted as $\text{LT}(\cdot)$, into $\tilde{d}$-D element features:

$$\mathbf{pE} = \text{LT}(\mathbf{rE}, p; \Phi), \tag{7}$$

where $\mathbf{rE} \in \mathbb{R}^{n \times \sum \forall d'}$ is the ranked element matrices and Φ denotes the transformation weight. The pooled $\mathbf{pE} \in \mathbb{R}^{p \times \tilde{d}}$ (suppose the final output element are $\tilde{d}$-D features) is then further fed into the regression layers to reach the semantic descriptions.

Element Decomposition Key-point localization is usually applied to predict a set of semantic key-points for objects (Huang, Gong, and Tao 2017; Sarlin et al. 2019; Guo and Farrell 2019). For example, a bird can have several standard key-points reflecting its appearance and subtle characteristics. Taking the CUB-Birds[1] dataset (Wah et al. 2011) as an example, a bird image can be detected with 15 key-points, of which each key-point is located at a specific element, e.g., forehead, beak, leg, tail, etc. These key-points can be used to align the objects and reveal their subtle differences which help to recognize different fine-grained classes, e.g., bird species. In our method, we follow the method of using a 2-dimensional probability distribution heat map of the object to localize the key-points. Specifically, a ResNet-34 (He et al. 2016) with the classification layer removed, is used as the encoder. Then, by stacking three blocks consisting of one upsampling (bilinear interpolation) layer, one convolution layer, one batch normalization layer, and one ReLU layer each, and a final convolution layer and upsampling layer to output the key-points location tensor. The elements decomposition can localize several key-points of a sample, and we can then use a cropping operation to decompose the sample (e.g., width W and height H) into several fine-grained elements based on these key-points as:

$$\begin{cases} x_p = x_k - w/2 \\ y_p = y_k - h/2 \end{cases}, \tag{8}$$

where (x_k, y_k) is the 2-d coordinate of one key-point, w and h are width and height of the cropped element. (x_p, y_p) is the 2-d coordinate of the top left corner of the cropped element, where x_p and y_p are potentially set to $(W - w)$ and $(H - h)$ respectively, or set to 0, to retain the cropped element within the sample size range. These elements can be regarded as the nodes of a sample-level graph, and each of them is further fed into a visual feature extractor to obtain the element features.

Sample-Level Graph Construction Given the elements, we design a verification-based *pseudo-link and propagate* method to identify the mutual influence among them, and further determine their linking edges. Notably, in our method, the edges are set based on their mutual influences, thus two linked elements are not necessarily spatially adjacent. To this end, we first assume that every two-element pair is initially established with a relation that can perform the node propagation as a graph, and then to *verify* whether or not such relation can be satisfied under certain measurement. Given several elements, we specify two elements, e.g., i and j, are initially connected by a *pseudo-link*. We denote f_i and f_j as element features and thus a *simulated one-pass propagation* can be achieved by:

$$f_i = f_i \upsilon_i + f_j \upsilon_{ij}, \tag{9}$$

$$f_j = f_j \upsilon_j + f_i \upsilon_{ji}, \tag{10}$$

where υ_{ij} / υ_{ji} is the weight of i w.r.t. j and j w.r.t. i, respectively. υ_i / υ_i are used to simulate the self-loop in graph. Regardless of the orders and weights, we can simply use the mean features of these two elements, i.e., $\frac{1}{2}(f_i + f_j)$, to approximatively indicate whether the propagation is *positive or not* for the establishment of edge between elements i and j. Notably, such a simulated propagation is similar to two elements performing graph convolution with the same weight of $1/2$ and containing a self-loop.

To verify the positivity, we train a mini-classifier based on the *whole feature representation* of training samples (seen classes only), and then to calculate the classification confidence, i.e., accuracy denoted as $Con(\cdot)$, for each two-element pair of single features f_i and f_j, and propagated features $\frac{1}{2}(f_i + f_j)$, respectively. If both conditions, i.e.,

$$Con((f_i + f_j)/2) > Con(f_i) + \varepsilon, \tag{11}$$

$$Con((f_i + f_j)/2) > Con(f_j) + \varepsilon, \tag{12}$$

satisfied, then we say that the propagation is verified as *positive* and the edge is confirmed. Here, ε is a small positive constant that controls the threshold.

Afterward, for all confirmed edges, we define the edge-weight between elements i and j as:

$$\omega_{ij} = \frac{Con((f_i + f_j)/2) - (Con(f_i) + Con(f_j))/2}{\max(Con(f_i), Con(f_j))}, \tag{13}$$

[1] http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

and for each single element, we define its relative importance w.r.t. the recognition of the whole sample as:

$$RI_i = \frac{Con(f_i)}{\sum_{l=1}^{e} Con(f_l)}, \tag{14}$$

where ω_{ij} and RI_i can reflect the strength of relation i and j and the contribution degree of element i to the recognition, respectively.

Training ZSL on Sample-Level Graphs Finally, the train-set can be presented as $\mathcal{D} = \{(\mathbf{A}_i, \mathbf{F}_i), y_i, s_i\}_{i=1}^{N}$. We feed these labeled seen class sample-level graphs into our ERA-GNNs, i.e., denoted as $G(\cdot; \mathbf{W}_\mathcal{G})$, in which we stack k graph convolution layers to obtain k sequential features as:

$$\left\{\mathbf{H}^{(i)}\right\}_{lr=1}^{k} = G(\mathbf{A}_i, \mathbf{F}_i; \mathbf{W}_\mathcal{G}), \tag{15}$$

where each $\mathbf{H}^{(i)} \in \mathbb{R}^{n \times c_i}$ is the propagated feature matrix of lr-th layer. Each row stores an element and each column represents a feature channel. Afterward, we use the designed Element Ranking Pooling to downsample the multi-scale graphs, and obtain the ranked and pooled graph representation **pE**. For simplicity, we denote the whole pooling process as $ERP(\cdot; \Phi)$. Last, the regression layers are convolutional and dense layers, i.e., denoted as $Rgs(\cdot; \mathbf{W}_\mathcal{R})$, that link to semantic descriptions. Thus, the training of ZSL can be formalized as:

$$\begin{aligned}\underset{\mathbf{W}_\mathcal{G}, \mathbf{W}_\mathcal{R}, \Phi}{\arg\min} \frac{1}{N} \cdot \sum_{i=1}^{N} \mathcal{L}\left(Rgs\left(ERP\left(G\left(\mathbf{A}_i, \mathbf{F}_i; \mathbf{W}_\mathcal{G}\right); \Phi\right); \mathbf{W}_\mathcal{R}\right), s_i\right) \\ + \varphi\left(\mathbf{W}_\mathcal{G}\right) + \gamma\left(\mathbf{W}_\mathcal{R}\right).\end{aligned} \tag{16}$$

where $\varphi(\cdot)$ and $\gamma(\cdot)$ are L2-norms that can add penalties as model complexity increases and thus avoid overfitting.

Experiments

Experimental Setup

Dataset and Evaluation Metrics Following (Ji et al. 2018; Elhoseiny et al. 2017), we evaluate our method on two widely used fine-grained datasets including CUB-Birds (Wah et al. 2011) and NABirds (Van Horn et al. 2015). Specifically, CUB-Birds consist of bird images covering 200 classes with 11,788 images. Each image is annotated with key-point location and attribute labels. For ZSL, 150 classes of bird images act as seen classes for training, and the remaining 50 classes are unseen classes. Each of their prototypes is represented by a 312-dimensional semantic attribute description which can present meaningful class-level information. The NABirds is a larger dataset containing 1,011 total classes with 48,562 images. While for ZSL, some gender-specific classes are further merged, resulting in 404 final classes. Among them, 323 classes are seen classes and the remaining 81 are unseen classes. Similarly, each image is also annotated with the required key-point location, while differently, the semantic descriptions of each class is a collected article from Wikipedia.

Two different settings are considered in our experiments including 1) classic ZSL and 2) generalized ZSL (GZSL). For ZSL, all test samples belong to unseen classes, i.e., the model only searches for the class prototypes on unseen classes P' (Eq. (2)). While for GZSL (Xian, Schiele, and Akata 2017), the search can also generalize to novel samples from seen classes, i.e., the model searches the class prototypes on both seen and unseen classes $\{P' \cup P\}$ (Eq. (3)).

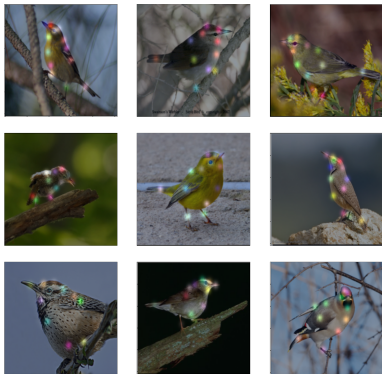


Figure 3: Localization results on ZSL setting: some examples (better viewed in color).

Key-point	PCK	Key-point	PCK
Back	92.4	Beak	97.2
Belly	91.3	Breast	92.8
Crown	98.4	Forehead	98.0
Left-eye	98.3	Left-leg	71.3
Left-wing	89.5	Nape	97.7
Right-eye	98.7	Right-leg	74.7
Right-wing	92.3	Tail	85.5
Throat	98.0	Overall	91.6

Table 1: Localization results on ZSL setting.

Implementation Our method is implemented by Pytorch and trained with NVIDIA RTX 3090 GPU. The GNNs consist of four graph convolution layers and the designed element ranking pooling layers. The regression layers are simple convolutional and dense layers that directly link to semantic descriptions. As to the element decomposition, we follow the general settings of PAIRS (Guo and Farrell 2019) to construct the key-point localization network. Specifically, a ResNet-34 with the classification layer removed is acted as an encoder. Three blocks consisting of one upsampling layer, one convolution layer, one batch normalization layer, and one ReLU layer each, and a final convolution and upsampling layers are stacked to decode the key-points location. The cropped element size w and h are both set to 56 for CUB-Birds and 224 for NABirds by an empirical investigation from the data statistics. As to the detection, we identify 15 elements for CUB-Birds and 6 elements for NABirds due to their availability. In the experiments, we also report the results when directly using the key-point annotations to construct the sample-level graph. Two results are denoted as Ours (D), i.e., detection-based model, and Ours (A), i.e., annotation-based model, respectively. When constructing the sample-level graph, we control the threshold ε

Method	F	Fine-grained	ACC (**CUB-Birds**)	ACC (**NABirds**)
ESZSL (Romera-Paredes and Torr 2015)	$\mathcal{G}$	×	48.7	24.3
JLSE (Zhang and Saligrama 2016)	$\mathcal{V}$	×	42.1	-
SYNC (Changpinyo et al. 2016)	$\mathcal{G}$	×	54.4	28.9
SAE (Kodirov, Xiang, and Gong 2017)	$\mathcal{G}$	×	61.4	-
RelationNet (Sung et al. 2018)	$\mathcal{G}$	×	62.0	-
S^2GA (Ji et al. 2018)	$\mathcal{V}$	✓	75.3	39.4
Chen *et al.* (Chen, Cao, and Ji 2019)	$\mathcal{G}$	✓	58.3	33.8
GAL (Yu and Lee 2019)	$\mathcal{G}$	×	62.5	-
Zhu *et al.* (Zhu et al. 2019)	$\mathcal{G}$	✓	70.5	35.7
AREN (Xie et al. 2019)	$\mathcal{R}$	✓	70.7	-
AMS-SFE (Guo and Guo 2020)	$\mathcal{G}$	×	70.1	-
MPGAN (Chen et al. 2020)	$\mathcal{V}$	✓	48.2	27.2
APNet (Liu et al. 2020)	$\mathcal{R}$	×	57.7	-
RGEN (Xie et al. 2020)	$\mathcal{G}$	✓	<u>76.1</u>	<u>41.4</u>
Keshari *et al.* (Keshari, Singh, and Vatsa 2020)	$\mathcal{R}$	×	60.8	-
DAZLE (Huynh and Elhamifar 2020)	$\mathcal{R}$	✓	64.1	35.5
LsrGAN (Vyas, Venkateswara, and Panchanathan 2020)	$\mathcal{R}$	×	60.3	-
Xu *et al.* (Xu et al. 2020)	$\mathcal{R}$	×	65.7	-
HSVA (Chen et al. 2021b)	$\mathcal{R}$	×	65.7	-
VGSE (Xu et al. 2022)	$\mathcal{R}$	×	35.0	-
TDCSS (Feng et al. 2022)	$\mathcal{R}$	×	61.1	-
Ours (D)	$\mathcal{G}$	✓	**76.9**	**42.8**
Ours (A)	$\mathcal{G}$	✓	**<u>78.7</u>**	**<u>44.2</u>**

Table 2: Comparison results of ZSL (accuracy %). 'F'-features: GoogleNet ($\mathcal{G}$), VGGNet ($\mathcal{V}$), ResNet ($\mathcal{R}$). The best result is marked in 'underlined bold', the second in 'bold', and the third in 'underlined'.

to retain $\cong 50$ and $\cong 20$ edges among the sample-level graph for CUB-Birds and NABirds, respectively. As to the visual features, we use GoogleNet (Szegedy et al. 2015) to extract a 1024-dimensional feature vector for each element.

Localization Results on ZSL Setting

We report the key-point localization results on CUB-Birds based on the ZSL data setting where only seen classes are used during training, and to detect key-points of samples from unseen classes. The PCK (percentage of correct key-points) score is used to measure the performance, i.e., a predicted key-point (p) is correct if it's within a small neighborhood of the ground truth (g):

$$\|p - g\| \leq c * \max(h_b, w_b), \tag{17}$$

where (h_b, w_b) is the longer side of the bounding box and c is a constant factor. The results are shown in Table 1 and Figure 3. It can be observed that most detected key-points are reasonable and accurate enough to be utilized as the base points of sample elements.

Comparison on ZSL Setting

To demonstrate the effectiveness of our proposed method, we first compare it with existing representative methods in the ZSL setting. We selected 21 competitors based on the following criteria: 1) published in the most recent years; 2) cover a wide range of models; 3) they clearly represent the state-of-the-art; and 4) all of them are under the standard splits (Xian, Schiele, and Akata 2017). We compute and report the multi-way classification accuracy as in previous works. The comparison results with the selected representative competitors are shown in Table 2. It can be seen from the results that our method outperforms all competitors with great advantages on both datasets. Taking the more significant CUB-Birds as an example, the prediction accuracy of our method achieves 76.9% and 78.7% for detection- and annotation-based models, respectively. Moreover, we can also observe that the performance of fine-grained-based methods is overally better than the average result of other competitors. Specifically, comparing with S^2GA (Ji et al. 2018), Chen *et al.* (Chen, Cao, and Ji 2019), Zhu *et al.* (Zhu et al. 2019), AREN (Xie et al. 2019), MPGAN (Chen et al. 2020), RGEN (Xie et al. 2020), and DAZLE (Huynh and Elhamifar 2020) which also fall into the fine-grained ZSL models, our method improves the prediction accuracy by a large margin as 3.4%, 20.4%, 8.2%, 8.0%, 30.5%, 2.6%, and 14.6%, respectively, which fully demonstrate the effectiveness of our method.

Comparison on GZSL Setting

In Table 3, we compare our method with 23 competitors on GZSL setting (Xian, Schiele, and Akata 2017). For the generalized ZSL, we compute the average per-class prediction accuracy on test images from unseen classes (U) and seen classes (S), respectively, and report the Harmonic Mean calculated by $H = (2 \times U \times S) / (U + S)$, which can quantify the aggregate performance across both seen and unseen classes. It can be seen from the results that, although most of these competitors cannot retain the same level of performance on both seen and unseen classes, our method can achieve the most balanced prediction accuracy. For example, ESZSL (Romera-Paredes and Torr 2015), SYNC (Changpinyo et al. 2016) and SAE (Kodirov, Xiang, and Gong 2017) have a very large margin, i.e., 51.2%, 59.4% and 50.1%, between their accuracy of seen and unseen classes in CUB-Birds, which result in poor performance on Harmonic Mean. In contrast, our method can obtain both comparative results on unseen classes and seen classes as 52.3% / 71.1% and 38.8% / 54.6%, for CUB-Birds and NABirds, respec-

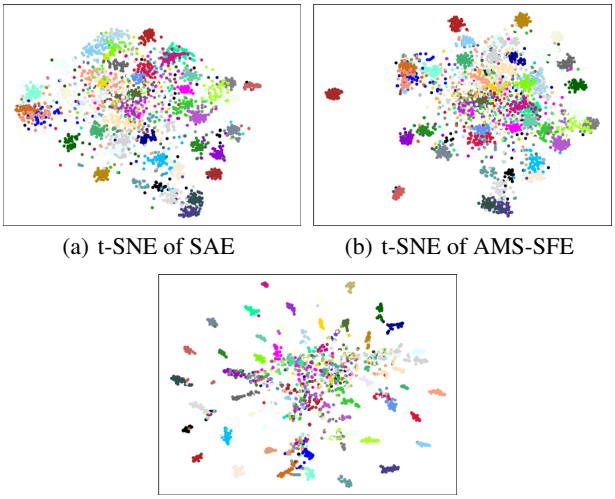
Method	F	Fine-grained	CUB-Birds			NABirds		
			U	S	HM	U	S	HM
ESZSL (Romera-Paredes and Torr 2015)	$\mathcal{V}$	×	12.6	63.8	21.0	13.5	44.2	20.7
SYNC (Changpinyo et al. 2016)	$\mathcal{G}$	×	11.5	70.9	19.8	16.3	49.5	24.5
SAE (Kodirov, Xiang, and Gong 2017)	$\mathcal{G}$	×	7.8	57.9	29.2	-	-	-
RelationNet (Sung et al. 2018)	$\mathcal{G}$	×	38.1	61.1	47.0	-	-	-
f-CLSWGAN (Xian et al. 2018)	$\mathcal{R}$	×	43.7	57.7	49.7	-	-	-
SE-GZSL (Kumar Verma et al. 2018)	$\mathcal{R}$	×	41.5	53.3	46.7	-	-	-
SP-AEN (Chen et al. 2018)	$\mathcal{R}$	×	34.7	70.6	46.6	-	-	-
Zhu *et al.* (Zhu et al. 2019)	$\mathcal{V}$	✓	36.7	71.3	48.5	28.6	53.4	37.2
SGAL (Yu and Lee 2019)	$\mathcal{G}$	×	40.9	55.3	47.0	-	-	-
DASCN (Ni, Zhang, and Xie 2019)	$\mathcal{R}$	×	45.9	59.0	51.6	-	-	-
AREN (Xie et al. 2019)	$\mathcal{R}$	✓	38.9	78.7	52.1	31.1	53.5	39.3
APNet (Liu et al. 2020)	$\mathcal{R}$	×	55.9	48.1	51.7	-	-	-
Keshari *et al.* (Keshari, Singh, and Vatsa 2020)	$\mathcal{R}$	×	44.8	59.9	51.3	-	-	-
DAZLE (Huynh and Elhamifar 2020)	$\mathcal{R}$	✓	65.3	42.0	51.1	39.7	44.5	42.0
LsrGAN (Vyas, Venkateswara, and Panchanathan 2020)	$\mathcal{R}$	×	47.7	57.0	51.9	-	-	-
FREE (Chen et al. 2021a)	$\mathcal{R}$	×	55.7	59.9	57.7	-	-	-
BZSL+Attributes (Badirli et al. 2021)	$\mathcal{R}$	✓	31.5	50.6	38.8	26.4	33.2	29.4
BZSL+Word Vectors (Badirli et al. 2021)	$\mathcal{R}$	✓	22.4	45.0	29.9	25.0	30.8	27.6
HSVA (Chen et al. 2021b)	$\mathcal{R}$	×	52.7	58.3	55.3	-	-	-
VGSE (Xu et al. 2022)	$\mathcal{R}$	×	24.1	45.7	31.5	-	-	-
TDCSS (Feng et al. 2022)	$\mathcal{R}$	×	44.2	62.8	51.9	-	-	-
SE-GZSL (Kim, Shim, and Shim 2022)	$\mathcal{R}$	×	60.3	53.1	56.4	-	-	-
CE-GZSL+SDFA^2 (Zhao et al. 2022)	$\mathcal{R}$	×	59.2	59.6	54.0	-	-	-
Ours (D)	$\mathcal{G}$	✓	51.2	68.4	**58.6**	37.1	51.2	**43.0**
Ours (A)	$\mathcal{G}$	✓	52.3	71.1	**60.3**	38.8	54.6	**45.4**

Table 3: Comparison results of GZSL (accuracy %). 'F'-features: GoogleNet ($\mathcal{G}$), VGGNet ($\mathcal{V}$), ResNet ($\mathcal{R}$). The best result is marked in 'underlined bold', the second in 'bold', and the third in 'underlined'.

tively, and thus result in the best result of Harmonic Mean as 60.3% and 45.5%, respectively. Our method outperforms all competitors for the most balanced prediction accuracy which can better fit a more realistic application scenario.

Mapping Robustness

We further conduct the evaluation of mapping robustness on our method on CUB-Birds. Given the trained model, we map unseen class samples from visual to semantic space. With these obtained semantic features, we apply t-SNE (Van der Maaten and Hinton 2008) to visualize them in a 2D map. We show the visualization results on two SOTAs, i.e., SAE (Kodirov, Xiang, and Gong 2017) and AMS-SFE (Guo and Guo 2020), and our method under the ZSL setting in Figure 4(a), Figure 4(b) and Figure 4(c), respectively. It can be seen from our method that only a small portion of unseen class samples are shifted in the semantic space. Moreover, the obtained semantic features are more continuous and aggregated. These merits demonstrate that our method significantly mitigates the domain bias problem.



(a) t-SNE of SAE (b) t-SNE of AMS-SFE

(c) t-SNE of Ours

Figure 4: Visualization results of mapping robustness.

Conclusion

This paper proposed a novel fine-grained ZSL framework based on the sample-level graph to address the challenging domain bias problem. Our method decomposes samples into fine-grained elements presented as graph structures, in which nodes and edges are different elements and relations among them. Taking advantage of GNNs, we reformulate the ZSL problem to a graph-to-semantics mapping task which can better exploit element-semantics correlation and local sub-structure information in samples. Experimental results verified the effectiveness of our method.

Acknowledgments

This research was supported by funding from the National Natural Science Foundation of China (No. 62102327, 61872310), PolyU Internal Fund (No. P0043932), Hong Kong RGC General Research Fund (No. 152211/23E, 152244/21E, 152203/20E, 152221/19E) and Research Impact Fund (No. R5060-19), Key-Area Research and Development Program of Guangdong Province (No. 2021B0101400003), and Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109142008673).

References

Akata, Z.; Reed, S.; Walter, D.; Lee, H.; and Schiele, B. 2015. Evaluation of output embeddings for fine-grained image classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2927–2936.

Badirli, S.; Akata, Z.; Mohler, G.; Picard, C.; and Dundar, M. 2021. Fine-grained zero-shot learning with dna as side information. In *Advances in Neural Information Processing Systems*, 19352–19362.

Changpinyo, S.; Chao, W.-L.; Gong, B.; and Sha, F. 2016. Synthesized classifiers for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5327–5336.

Chen, H.; Cao, L.; and Ji, R. 2019. Learning similarity-specific dictionary for zero-shot fine-grained recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3697–3701.

Chen, L.; Zhang, H.; Xiao, J.; Liu, W.; and Chang, S.-F. 2018. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1043–1052.

Chen, S.; Wang, W.; Xia, B.; Peng, Q.; You, X.; Zheng, F.; and Shao, L. 2021a. Free: Feature refinement for generalized zero-shot learning. In *IEEE/CVF International Conference on Computer Vision*, 122–131.

Chen, S.; Xie, G.; Liu, Y.; Peng, Q.; Sun, B.; Li, H.; You, X.; and Shao, L. 2021b. Hsva: Hierarchical semantic-visual adaptation for zero-shot learning. In *Advances in Neural Information Processing Systems*, 16622–16634.

Chen, Z.; Luo, Y.; Qiu, R.; Wang, S.; Huang, Z.; Li, J.; and Zhang, Z. 2021c. Semantics Disentangling for Generalized Zero-Shot Learning. In *IEEE/CVF International Conference on Computer Vision*, 8712–8720.

Chen, Z.; Wang, S.; Li, J.; and Huang, Z. 2020. Rethinking Generative Zero-Shot Learning: An Ensemble Learning Perspective for Recognising Visual Patches. In *28th ACM International Conference on Multimedia*, 3413–3421.

Elhoseiny, M.; Zhu, Y.; Zhang, H.; and Elgammal, A. 2017. Link the head to the" beak": Zero shot learning from noisy text description at part precision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5640–5649.

Feng, Y.; Huang, X.; Yang, P.; Yu, J.; and Sang, J. 2022. Non-generative Generalized Zero-shot Learning via Task-correlated Disentanglement and Controllable Samples Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9346–9355.

Fu, Y.; Hospedales, T. M.; Xiang, T.; and Gong, S. 2015. Transductive multi-view zero-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11): 2332–2345.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 1263–1272.

Guo, J.; and Guo, S. 2019. Adaptive adjustment with semantic feature space for zero-shot recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3287–3291.

Guo, J.; and Guo, S. 2020. A novel perspective to zero-shot learning: Towards an alignment of manifold structures via semantic feature expansion. *IEEE Transactions on Multimedia*, 23: 524–537.

Guo, P.; and Farrell, R. 2019. Aligned to the Object, not to the Image: A Unified Pose-aligned Representation for Fine-grained Recognition. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 1876–1885.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 770–778.

Huang, H.; Wang, C.; Yu, P. S.; and Wang, C.-D. 2019. Generative dual adversarial network for generalized zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 801–810.

Huang, S.; Gong, M.; and Tao, D. 2017. A coarse-fine network for keypoint localization. In *IEEE International Conference on Computer Vision*, 3028–3037.

Huynh, D.; and Elhamifar, E. 2020. Fine-grained generalized zero-shot learning via dense attribute-based attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4483–4493.

Ji, Z.; Fu, Y.; Guo, J.; Pang, Y.; Zhang, Z. M.; et al. 2018. Stacked semantics-guided attention model for fine-grained zero-shot learning. In *Advances in Neural Information Processing Systems*, 5995–6004.

Kampffmeyer, M.; Chen, Y.; Liang, X.; Wang, H.; Zhang, Y.; and Xing, E. P. 2019. Rethinking knowledge graph propagation for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11487–11496.

Keshari, R.; Singh, R.; and Vatsa, M. 2020. Generalized zero-shot learning via over-complete distribution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13300–13308.

Khan, M. G. Z. A.; Naeem, M. F.; Van Gool, L.; Pagani, A.; Stricker, D.; and Afzal, M. Z. 2023. Learning Attention Propagation for Compositional Zero-Shot Learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 3828–3837.

Kim, J.; Shim, K.; and Shim, B. 2022. Semantic feature extraction for generalized zero-shot learning. In *AAAI Conference on Artificial Intelligence*, volume 36, 1166–1173.

Kodirov, E.; Xiang, T.; and Gong, S. 2017. Semantic autoencoder for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3174–3183.

Kumar Verma, V.; Arora, G.; Mishra, A.; and Rai, P. 2018. Generalized zero-shot learning via synthesized examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4281–4289.

Liu, L.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2020. Attribute Propagation Network for Graph Zero-shot Learning. In *AAAI Conference on Artificial Intelligence*, volume 34, 4868–4875.

Liu, Z.; Guo, S.; Guo, J.; Xu, Y.; and Huo, F. 2022. Towards unbiased multi-label zero-shot learning with pyramid and semantic attention. *IEEE Transactions on Multimedia*.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11): 39–41.

Ni, J.; Zhang, S.; and Xie, H. 2019. Dual adversarial semantics-consistent network for generalized zero-shot learning. In *Advances in Neural Information Processing Systems*, 6143–6154.

Romera-Paredes, B.; and Torr, P. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, 2152–2161.

Sarlin, P.-E.; Cadena, C.; Siegwart, R.; and Dymczyk, M. 2019. From coarse to fine: Robust hierarchical localization at large scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12716–12725.

Schonfeld, E.; Ebrahimi, S.; Sinha, S.; Darrell, T.; and Akata, Z. 2019. Generalized zero-and few-shot learning via aligned variational autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8247–8255.

Su, H.; Li, J.; Chen, Z.; Zhu, L.; and Lu, K. 2022. Distinguishing Unseen from Seen for Generalized Zero-shot Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7885–7894.

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1199–1208.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1–9.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11): 2579–2605.

Van Horn, G.; Branson, S.; Farrell, R.; Haber, S.; Barry, J.; Ipeirotis, P.; Perona, P.; and Belongie, S. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 595–604.

Vyas, M. R.; Venkateswara, H.; and Panchanathan, S. 2020. Leveraging seen and unseen semantic relationships for generative zero-shot learning. In *European Conference on Computer Vision*, 70–86. Springer.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. In *Computation & Neural Systems Technical Report (California Institute of Technology)*, 1–8.

Wang, X.; Ye, Y.; and Gupta, A. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6857–6866.

Xian, Y.; Lorenz, T.; Schiele, B.; and Akata, Z. 2018. Feature generating networks for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5542–5551.

Xian, Y.; Schiele, B.; and Akata, Z. 2017. Zero-shot learning-the good, the bad and the ugly. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4582–4591.

Xie, G.-S.; Liu, L.; Jin, X.; Zhu, F.; Zhang, Z.; Qin, J.; Yao, Y.; and Shao, L. 2019. Attentive region embedding network for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9384–9393.

Xie, G.-S.; Liu, L.; Zhu, F.; Zhao, F.; Zhang, Z.; Yao, Y.; Qin, J.; and Shao, L. 2020. Region graph embedding network for zero-shot learning. In *European Conference on Computer Vision*, 562–580. Springer.

Xu, W.; Xian, Y.; Wang, J.; Schiele, B.; and Akata, Z. 2020. Attribute prototype network for zero-shot learning. In *Advances in Neural Information Processing Systems*, 21969–21980.

Xu, W.; Xian, Y.; Wang, J.; Schiele, B.; and Akata, Z. 2022. VGSE: Visually-Grounded Semantic Embeddings for Zero-Shot Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9316–9325.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, 4800–4810.

Yu, H.; and Lee, B. 2019. Zero-shot Learning via Simultaneous Generating and Learning. In *Advances in Neural Information Processing Systems*, 46–56.

Yu, Y.; Ji, Z.; Guo, J.; and Zhang, Z. 2018. Zero-shot learning via latent space encoding. *IEEE Transactions on Cybernetics*, 49(10): 3755–3766.

Zhang, L.; Xiang, T.; and Gong, S. 2017. Learning a deep embedding model for zero-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021–2030.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 4438–4445.

Zhang, Z.; and Saligrama, V. 2015. Zero-shot learning via semantic similarity embedding. In *IEEE International Conference on Computer Vision*, 4166–4174.

Zhang, Z.; and Saligrama, V. 2016. Zero-shot learning via joint latent similarity embedding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6034–6042.

Zhao, X.; Shen, Y.; Wang, S.; and Zhang, H. 2022. Boosting Generative Zero-Shot Learning by Synthesizing Diverse Features with Attribute Augmentation. In *AAAI Conference on Artificial Intelligence*, volume 36, 3454–3462.

Zhu, Y.; Xie, J.; Tang, Z.; Peng, X.; and Elgammal, A. 2019. Semantic-Guided Multi-Attention Localization for Zero-Shot Learning. In *Advances in Neural Information Processing Systems*, 14917–14927.