Generative Adversarial Zero-shot Learning via Knowledge Graphs

Yuxia Geng[1], Jiaoyan Chen[2], Zhuo Chen[1], Zhiquan Ye[1], Zonggang Yuan[3],
Yantao Jia[3], and Huajun Chen[1]

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
{gengyx,chenzhuo98,yezq,huajunsir}@zju.edu.cn
[2] Department of Computer Science, University of Oxford, Oxford, UK
jiaoyan.chen@cs.ox.ac.uk
[3] Huawei Technologies Co., Ltd, China
yuanzonggang@huawei.com,jamaths.h@163.com

Abstract. Zero-shot learning (ZSL) is to handle the prediction of those unseen classes that have no labeled training data. Recently, generative methods like Generative Adversarial Networks (GANs) are being widely investigated for ZSL due to their high accuracy, generalization capability and so on. However, the side information of classes used now is limited to text descriptions and attribute annotations, which are in short of semantics of the classes. In this paper, we introduce a new generative ZSL method named KG-GAN by incorporating rich semantics in a knowledge graph (KG) into GANs. Specifically, we build upon Graph Neural Networks and encode KG from two views: *class* view and *attribute* view considering the different semantics of KG. With well-learned semantic embeddings for each node (representing a visual category), we leverage GANs to synthesize compelling visual features for unseen classes. According to our evaluation with multiple image classification datasets, KG-GAN can achieve better performance than the state-of-the-art baselines.

Keywords: Zero-shot Learning, Generative Adversarial Networks, Knowledge Graphs, Graph Neural Networks

1 Introduction

Machine learning often operates on a closed world assumption: it trains the model with a number of labeled samples and makes prediction with classes that have appeared in the training stage (i.e., seen classes) alone. This limitation raises a hot research interest in Zero-shot Learning (ZSL), which aims to handle novel classes without any training samples (i.e., unseen classes). An intuitive idea to deal with such unseen classes is to take advantages of *side information* of classes, which builds the semantic relationships among classes and enables transferring knowledge obtained from seen classes to the unseen. For example, in classifying animal images, the side information usually contains the visual characteristics of classes (e.g., human-annotated attributes or textual descriptions from external sources like Wikipedia) and how unseen classes are related to seen classes (e.g., class hierarchy in taxonomy), as Fig. 1 shows.

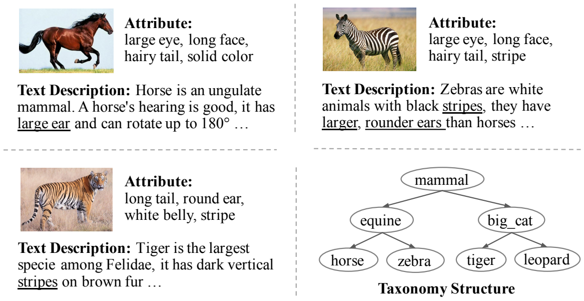

Attribute:
large eye, long face, hairy tail, solid color
Text Description: Horse is an ungulate mammal. A horse's hearing is good, it has large ear and can rotate up to 180° …
Attribute:
large eye, long face, hairy tail, stripe
Text Description: Zebras are white animals with black stripes, they have larger, rounder ears than horses …
Attribute:
long tail, round ear, white belly, stripe
Text Description: Tiger is the largest specie among Felidae, it has dark vertical stripes on brown fur …
mammal
equine
big_cat
horse
zebra
tiger
leopard
Taxonomy Structure

Fig. 1. The general side information of three classes: *horse*, *zebra* and *tiger*, including attribute annotations, text descriptions and their relationship in taxonomy structure.

ZSL is often divided into two paradigms. One is based on mapping [14,19,13,24]. It learns a general mapping function to map visual features and/or semantic features into the same latent space and then conduct nearest neighbor search to predict the class labels. However, it is a non-trivial task to bridge the *semantic gap* between such two spaces since the class-level semantic descriptions produce non-visual features. Additionally, the nearest neighbor search suffers from the *hubness* problem, that is, the neighborhoods of the mapped elements are biased to some hubs vectors, pushing the correct labels down the neighbor list [21].

The other ZSL paradigm is developed upon generative models such as generative adversarial network (GANs) [26,28,9,16,29,22]. These methods utilize side information of classes to synthesize samples (or features) for unseen classes, circumvent the need for space mapping, thus avoiding the *semantic gap* problem as well. Such a generative solution transforms the ZSL problem into a traditional supervised learning problem which can be flexibly dealt with by kinds of existing methods. Moreover, the generated unseen samples (or features) can alleviate the training bias towards seen classes and avoid the above mentioned *hubness* problem.

However, most of these generative methods are built upon one type of side information such as attribute annotations, taxonomy structure or textual descriptions. Thus, they often generate less discriminative samples (or features) without enough variation. Considering the example of human-annotated attributes of animal classes, when we use attribute “stripe” to generate samples for class *zebra*, another significantly different class *tiger* which is also annotated with “stripe” may also obtain synthesized samples with similar features as *zebra* (i.e., domain-shift problem [6]), especially when *tiger* lacks of other representative attribute annotations. Taxonomy structure describes the inter-class relationship in taxonomy, e.g., *horse* belongs to *equine* while *tiger* belongs to *big cat*. However, it will generate less discriminative samples for sibling classes which may look quite different, such as *horse* and *zebra*. Textual descriptions can be easily collected but are prone to introduce much noise due to the ambiguity and irrelevant words and phrases.

In this paper, we propose to incorporate a knowledge graph (KG) which contains semantics of all the above mentioned side information and can lead to a higher ZSL performance. To this end, we first build the KG with two views[4]: a *class* view for taxonomy structure and an *attribute* view for attribute annotations, and embed it into a vector space with Graph Neural Networks (GNNs) together with the class name word vector learned from textual descriptions. We then propose KG-GAN – a new generative ZSL framework utilizing the above KG embeddings, and Generative Adversarial Networks (GANs) which synthesize discriminative visual features for each class. Unlike previous generative methods using engineered regularizers or complex networks, our framework adopts the basic GAN model without any additions. Our main contributions are as follows:

- As far as we know, KG-GAN is among the first to utilize formal and rich semantics represented by a KG in generative ZSL. The KG describes different aspects of ZSL classes, promoting the knowledge transfer between seen and unseen classes.
- We develop Graph Neural Networks to learn semantically meaningful class embeddings so as to investigate how class semantics influences the feature transfer in ZSL.
- We contribute two new ZSL benchmarks on image classification as well as their corresponding KGs. Experiments on these two benchmarks show that the generated features are quite effective to both seen and unseen classes, and promising results have been achieved in comparison with the state-of-the-art baselines including both generative and none generative ZSL methods.

2 Related Work

2.1 Mapping-based vs Generative ZSL

In mapping-based zero-shot learning literature, methods like [5,15,19,13] tried to map visual features into semantic space, and found the most similar class by computing nearest neighbor on class embeddings. However, these methods tend to aggravate *hubness* problem [23] because a number of visual features are mapped into a point in the semantic space for a certain class, leading to an increasing probability of irrelevant points (hubs) being the nearest neighbors (i.e., the correct labels). Some other methods proposed to map class embeddings into visual space to suppress this problem [4,27,2], in which the features of unseen classes are learned by transferring features of seen classes based on the class relatedness in semantic space. However, the feature transfer is restricted by the

[4] Note that a KG can be constructed by various automatic and semi-automatic tools with domain knowledge and domain experts.

mapping of two spaces and heavily dependent on the semantic relationships of classes, which may be undermined by the *semantic gap* problem. Besides, the learning of unseen classifiers only rely on the samples of seen classes, which may have strong bias towards seen classes during prediction.

Different from mapping-based ZSL, generative zero-shot learning directly synthesizes unseen samples (or features) with random noises which are conditioned by the class side information. For example, Zhu et al. [28] utilized GANs as generative models which took the textual descriptions of classes from Wikipedia articles as input and generated visual features for these classes, with a fully connected layer being used to reduce text noise. Additionally, they also proposed a visual pivot regularization to preserve the inter-class discrimination of generated features. Huang et al. [9] introduced a generator to synthesize sample features with class embeddings and a regressor to project generated features back to their corresponding class embeddings, while a discriminator was to evaluate the closeness of visual features and class embeddings. However, most of these methods rely on engineered regularizers or auxiliary networks to guarantee the quality of generated samples. Few of them consider the effectiveness of class semantics. In our paper, we leverage knowledge graph which contains rich class semantics to enhance the feature generation in ZSL, making the synthesized data discriminative and variational.

2.2 KG-based ZSL

There have been some studies that utilize KGs to enhance mapping-based ZSL. For instance, Wang et al. [24] proposed a KG which depicted the class hierarchy in taxonomy, and used this hierarchical relationship to predict classifiers for unseen classes, where Graph Convolutional Network (GCN) was applied to transfer features from seen classifiers to unseen classifiers. This method as well as its derivative [10] have the following problems: 1) the KG used is homogeneous, where the class semantics is limited especially for discriminating those sibling classes; 2) the models are trained only using seen samples and have strong bias towards seen classes during prediction, especially in the generalized ZSL scenario where the prediction involves both seen and unseen class labels. In contrast, we use a KG with richer semantics from various side information including class structure in taxonomy, class attributes and class name word vectors, while the generative solution which generates training samples for unseen classes is free of the bias issue in the generalized ZSL scenario. As far as we know, there are currently no works that incorporate such a KG in generative ZSL.

3 Methodology

In this section, we introduce KG-GAN in details, as shown in Fig. 2. We first utilize unsupervised graph neural networks, i.e., Graph Auto-Encoders (GAEs), to embed our knowledge graph which includes two views: the *class* view which models the hierarchical relationship among classes, and the *attribute* view which

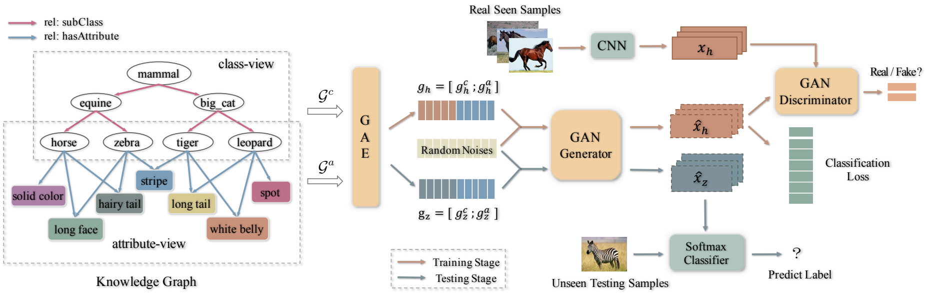


Fig. 2. An overview of our KG-GAN framework. The left is the knowledge graph consisting of nodes (i.e., *classes* and *attributes*) and relation edges (i.e., *subClass* and *hasAttribute*). The right is the GAN module. g_h and g_z represent the class embeddings of *horse* and *zebra* respectively, which consist of embeddings learned from *class*-view graph $\mathcal{G}^c$ and *attribute*-view graph $\mathcal{G}^a$. *horse* is seen in the training set, x_h and $\hat{x}_h$ are its real image features and synthesized image features, while *zebra* is unseen, whose features $\hat{x}_z$ are synthesized to learn classifier for it at testing stage.

models the human annotated visual characteristics. Briefly, we first obtain one embedding vector for each class according to its corresponding node in the KG. Secondly, a GAN with Wasserstein distance [1] and a classification loss is adopted for feature generation. It includes (i) a generator for synthesizing visual features from random noises that are conditioned on the class embedding, (ii) a discriminator for distinguishing the generated features from real ones, and (iii) a supervised classification loss for discriminating classes with the generated features. Note that we generate image features instead of raw images for both higher accuracy and computation efficiency [26], and we adopt some pre-trained CNN models (e.g., ResNet) for extracting image features. Finally, with well-trained generator, we can synthesize compelling image features for each unseen class and train a softmax classifier for it.

3.1 Preliminaries: ZSL and KG

We first formalize the ZSL problem as follows. Let $\mathcal{D}_{tr} = \{(x, y)|x \in \mathcal{X}_s, y \in \mathcal{Y}_s\}$ be the training set of ZSL, where x is the CNN feature of training image, y represents the class label in $\mathcal{Y}_s$ consisting of seen classes. While the testing set is denoted as $\mathcal{D}_{te} = \{(x, y)|x \in \mathcal{X}_u, y \in \mathcal{Y}_u\}$, where $\mathcal{Y}_u$, the set of unseen classes, has no overlap with $\mathcal{Y}_s$. At training stage, $\mathcal{D}_{tr}$ and $\mathcal{Y}_u$ are usually used to learn one classifier for each unseen class. We study two settings in prediction: the ZSL setting and the generalized ZSL (GZSL) setting. The former is to predict the label of testing samples in $\mathcal{X}_u$ with candidates from $\mathcal{Y}_u$, while the latter extends the testing set to $\mathcal{X}_s \cup \mathcal{X}_u$, with candidate labels from both seen classes and unseen classes i.e., $\mathcal{Y}_s \cup \mathcal{Y}_u$.

A KG is used as additional input for the above mentioned training. The KG, denoted as $\mathcal{G}$, includes a set of class nodes, denoted as $\mathcal{C} = \{c_1, c_2, ..., c_n\}$, and a set of attribute nodes, denoted as $\mathcal{A} = \{a_1, a_2, ..., a_m\}$, as the left of Fig. 2 shows. The KG is also composed of two views (parts): *class*-view denoted as $\mathcal{G}^c$, and *attribute*-view denoted as $\mathcal{G}^a$. $\mathcal{G}^c$ is formed by the class nodes in $\mathcal{C}$, where a relation (edge) named "*subClass*" is used to model the hierarchical relationship of classes defined in taxonomy. $\mathcal{G}^a$ is a heterogeneous bipartite graph consisting of class nodes and attribute nodes, where each class node is connected with a series of attributes nodes annotated for it via "*hasAttribute*" relation edge. In KG embedding, we learn a function $g(\cdot)$ to encode each class node as a vector known as *class embedding*. For each class y in $\mathcal{Y}_s \cup \mathcal{Y}_u$, we can learn two embeddings $g^c(y)$ and $g^a(y)$, according to the *class*-view and the *attribute*-view respectively. For convenience, the embeddings of i-th class are denoted with the subscript, i.e., g_i^c and g_i^a.

3.2 Class Embedding from KG

Graph Auto-Encoder (GAE) [12] is a method for unsupervised learning on graph-structured data based on the variational auto-encoder (VAE) [11]. It takes graph convolutional network (GCN) as encoder and inner product as decoder, enabling the latent representations of graph nodes to be learned, which are the class embeddings we desire.

Graph Encoder GCN works on propagating information between the nodes in graph via a series of graph convolutions and capturing the dependence of graph-structured data. We therefore use GCN to encode the inter-class relationship and the class-attribute correlation reflected in the proposed KG.

In each layer of GCN, the convolutional operation computes a node's vector representation by aggregating the vectors of its neighboring nodes defined in the graph, and update it to the next layer. Stacking the convolutional operation one after another, we can output the latent embeddings of graph nodes at last layer.

Given the KG $\mathcal{G}$, we first apply GCN on graph $\mathcal{G}^c$ and compute the embeddings of class nodes from the *class*-view. For class i, its k-th layer vector is represented as:

$$g_{i,k}^c = \sigma(W_k^c \sum_{j \in N_i} \frac{g_{j,k-1}^c}{|N_i|} + B_k^c g_{i,k-1}^c) \tag{1}$$

where N_i is the set of neighboring classes of class i, W_k^c and B_k^c denote the layer-specific trainable weight matrix and bias term in *class*-view, respectively. The latent embedding of class i from class-view is outputted at last layer: $g_i^c = g_{i,K}^c$.

We then apply another GCN on graph $\mathcal{G}^a$, where the vectors of class nodes are aggregated by the vectors of their neighboring attribute nodes. Similar with the *class*-view, the vector of class i in the l-th layer is computed as:

$$g_{i,l}^a = \sigma(W_l^a \sum_{j \in M_i} \frac{g_{j,l-1}^a}{|M_i|} + B_k^a g_{i,l-1}^a) \tag{2}$$

where M_i is the set of neighboring attributes of class i (i.e., the annotated attributes of class i), and $g^a_{j,l-1}$ is the vector representation of attribute j at $(l-1)$-th layer. W^a_l and B^a_l represent the layer-specific weight and bias in *attribute*-view. We also obtain the *attribute*-view latent embedding of class i from the output of last layer of GCN: $g^a_i = g^a_{i,L}$.

Finally, we concatenate the *class*-view embedding g^c_i and the *attribute*-view embedding g^a_i to form the *class embedding* for class i:

$$g_i = [g^c_i ; g^a_i] \tag{3}$$

To enrich the semantics of graph nodes, we initialize the node representations with word embeddings of class name and attribute name that are trained on skip-gram model on Wikipedia articles. These embeddings are taken as the input of the two GCNs.

Graph Decoder To preserve the relationship between two nodes that are connected via a relation edge, the decoder performs proximity calculation between these linked nodes in the latent space. Specifically, for each linked node pair, we conduct the inner product between their latent embeddings. With observed links in the graph, we can optimize the model by minimizing the following loss function:

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} log\ \sigma(g_i^\top \cdot g_j) - w \sum_{(i,j')\in\Omega^-} log\ \sigma(-g_i^\top \cdot g_{j'}) \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, (i, j) is a pair of linked nodes, and j' is a node not connected with i. Ω is the observed (positive) link set, Ω^- is the negative set, which involves all unlinked node pairs (the complement of Ω), and w is a weight computed by the ratio of number of positive and negative links. In this way, we keep nodes with links to be close to each other and nodes without links far apart, and optimize the learning of latent representation of graph nodes.

3.3 Feature Generation with GAN

With well-learned semantically meaningful *class embeddings*, we learn a generator G, which takes class embedding $g(y)$ and random noise vector z sampled from Gaussian distribution $\mathcal{N}(0, 1)$ as its inputs and synthesizes a CNN image feature $\hat{x}$ of class y. The loss of G is defined as:

$$\mathcal{L}_G = -\mathbb{E}[D(\hat{x})] - \lambda\mathbb{E}[logP(y|\hat{x})] \tag{5}$$

where $\hat{x} = G(z, g(y))$. The first term of loss function is the Wasserstein loss [1], and the second term is the supervised classification loss for classifying the synthesized features. λ is the corresponding weight coefficient.

The discriminator D then takes synthesized features $\hat{x}$ and real features x extracted from a training image of y as input, the loss can be formulated as:

$$\mathcal{L}_D = \mathbb{E}[D(x, g(y))] - \mathbb{E}[D(\hat{x})] - \beta\mathbb{E}[(|| \bigtriangledown_{\tilde{x}} D(\tilde{x})||_p - 1)^2] \tag{6}$$

where the first two terms approximate the Wasserstein distance of the distribution of real features and synthesized features, and the last term is the gradient penalty to enforce the gradient of D to have unit norm (i.e., Lipschitz constraint proposed in [7]), in which $\tilde{x} = \varepsilon x + (1-\varepsilon)\hat{x}$ with $\varepsilon \sim U(0,1)$, and β is the corresponding weight coefficient.

The GAN is optimized by a minimax game, which minimizes the loss of G but maximizes the loss of D. We also note that the generator and discriminator are both incorporated with class embeddings during training. This is a typical method of conditional GANs [18] that introduce external information to guide the training of GANs, which is completely consistent with generative ZSL – synthesizing visual features based on the side information of classes. In addition, unlike most proposed generative methods that introduce auxiliary regularization or networks to ensure the inter-class discrimination of generated features, we implement our feature generation module with basic Wasserstein GAN and classification loss. The core of our KG-GAN is to produce discriminative visual features conditioned on diverse and characteristic class semantics from KG.

3.4 Softmax Classifiers for Unseen Classes

At training stage, the image features and class embeddings of seen classes are used to train the GAN model. Once well trained, the KG-GAN is able to synthesize visual features of unseen classes from random noises with their corresponding class embeddings, since these unseen classes are semantically related to seen classes in knowledge graph. Consequently, with synthesized unseen data $\hat{\mathcal{X}}_u$, we can learn a softmax classifier for each unseen class and classify its testing samples. The classifier is optimized by:

$$\min_{\theta} -\frac{1}{|\mathcal{X}|} \sum_{(x,y)\in(\mathcal{X},\mathcal{Y})} logP(y|x;\theta) \tag{7}$$

where $\mathcal{X}$ represents the image features for training, $\mathcal{Y}$ is the label set to be predicted, θ is the training parameter and $P(y|x;\theta) = \frac{exp(\theta_y^T x)}{\sum_i^{|\mathcal{Y}|} exp(\theta_i^T x)}$. Regarding the different prediction setting, $\mathcal{X} = \hat{\mathcal{X}}_u$ when it is ZSL and $\mathcal{X} = \mathcal{X}_s \cup \hat{\mathcal{X}}_u$ when it is GZSL, while the label set $\mathcal{Y}$ is set to $\mathcal{Y}_u$ and $\mathcal{Y}_s \cup \mathcal{Y}_u$ respectively.

4 Experiments

We now perform experiments on image classification task to evaluate our proposed KG-GAN. Firstly, we compare KG-GAN against the state-of-the-art baselines in ZSL and GZSL setting, and then we explore whether the rich semantics from our KG is more effective than other side information like textual descriptions (i.e., class word vectors). We also analyze the impact of the *class embeddings* learned from different views of KG, and validate that incorporating semantic embeddings with the basic GAN is superior than incorporating additional regularization which is widely used in those baselines.

Table 1. Statistics of the two proposed datasets.

Dataset	Classes #	Seen Classes #	Unseen Classes #	Attributes #	Total Images #
ImNet-A	80	25	55	76	77,173
ImNet-O	35	10	25	38	39,361

4.1 Experiment Setting

Datasets We extract evaluation data from widely used benchmark ImageNet [3]. It is a large scale image classification dataset including a total of 21K classes and these classes are hierarchically related as the taxonomy structure stored in WordNet [17]. Predicting on ImageNet is challenging since $1,000$ classes are taken as *seen* classes that have training samples but about 20K classes without training samples are taken as *unseen* classes. Moreover, ImageNet contains a collection of fine-grained datasets as well as coarse-grained datasets. The classes from fine-grained subset are usually grouped into different families, e.g. different vehicle types and different bird types.

To study the semantic relationship between classes in ZSL, we extract two datasets from the fine-grained subset for evaluation. One is domain-specific for animal classification (i.e., ImNet-A) and the other is general for object classification (i.e., ImNet-O). The dataset split follows the original *seen-unseen* split proposed in [25]. Statistically, there are 25 seen classes in ImNet-A, each of which contains around 1300 images as training samples, and there are 55 unseen classes which are ancestors, descendants or siblings of seen classes but without training samples. Similarly, in ImNet-O, 10 classes are taken as seen classes and 25 as unseen classes. The details of dataset are listed in Table 1.

Knowledge Graph Construction We adopt the original taxonomy structure of WordNet as the backbone of knowledge graph, in which ImageNet classes are extracted and connected with each other via *subClass* relation, as Fig. 2 shows. Moreover, as attributes of ImageNet classes are not available, we invite volunteers to manually annotate attributes for these classes. Specifically, for each class, annotators are asked to assign $3 \sim 6$ attributes from an attribute list[5] with 25 images as references. Each class is reviewed by $3 \sim 4$ volunteers, and we take consensus among the annotators as the final annotations. Finally, we annotate 76 attributes for ImNet-A classes and 38 attributes for ImNet-O classes. We add these attributes into knowledge graph, and link them with corresponding class nodes via *hasAttribute* relation. The details of constructed KG are attached in the supplemental material.

Baselines and Metrics We adopt both classic and the state-of-the-art ZSL methods as baselines. Specifically, **DeViSE** [5], **CONSE** [19] and **SAE** [13] are

[5] The list is collected from attribute annotations of well-known ZSL datasets, e.g., AWA, CUB and SUN, and appearance descriptions of classes from Wikipedia.

methods that map visual features into semantic space, and predict the labels of testing samples by computing nearest neighbor on class name word vectors; Methods like **SYNC** [2], **GCNZ** [24] and **DGP** [10] propose to map class name word vectors into visual space to learn classifiers for unseen classes. Note that GCNZ and DGP are two state-of-the-art ZSL methods that utilize KGs. While **GAZSL** [28], **LisGAN** [16] and **ZSL-ABP** [29] are generative methods which generate visual features conditioned on class name word vectors (i.e., utilizing textual descriptions). For fair comparisons, we re-evaluate these baselines on our proposed datasets and re-implement all methods in the same setting[6].

Following previous work [25], we evaluate baselines and KG-GAN with **Hit@k**, i.e., the ratio of samples whose top k predicted labels hit the ground-truth label. Considering the unbalanced number of samples of unseen classes in ImageNet, we compute the Hit@k independently for each class and average them as the results. In ZSL testing setting, the result is the average of Hit@k of each unseen class. While in GZSL, we compute the harmonic mean $H = (2 * H_s * H_u)/(H_s + H_u)$, where H_s and H_u represent the average per-class Hit@k on seen classes and unseen classes respectively. Notably, we set k to $1, 2, 5$ in ZSL setting, and set k to 1 in GZSL setting.

Implementation We employ well-performed CNN model ResNet101 [8] to extract $2,048$-dimensional visual features for images, which is pre-trained on the samples of seen classes in ImageNet. As for class name word vectors, we use pre-trained word embeddings provided by Changpinyo et al. [2], they train skip-gram model on Wikipedia corpus and learn a 500-dimensional word vector for each class. Since there is no pre-trained attribute name word embeddings, we train them on Wikipedia corpus using Glove [20] model.

In our KG-GAN, the encoder of GAE adopts a two-layer GCN. After encoding the KG from two views, we obtain a 100-dimensional *class embedding* for each class. And we also set the dimension of noise vector z to 100. The generator and discriminator of GAN are both implemented with two fully connected layers, in which the generator has $4,096$ hidden units and outputs synthesized features with $2,048$ dimensions, and the discriminator also has $4,096$ hidden units and outputs a 2-dimensional vector to indicate the input feature is real or not. In training GAE module, the learning rate is set to 0.001, while in training GAN, the learning rate is set to 0.0001. We set the weight λ for classification loss to 0.01, and the weight β for gradient penalty to 10.

4.2 Comparison with Baselines

ZSL Setting We first report the zero-shot learning results in Table 2. It can be seen that our method achieves the best results on Hit@1 and Hit@2 one two datasets, and also achieves state-of-the-art results on Hit@5. In particular, on ImNet-A, the performance is improved by 6.17% over the state-of-the-art DGP

[6] The implementations of our method and all these baselines, as well as our data sets will be released if the paper is accepted.

Table 2. Performance (%) of KG-GAN and baselines on ImNet-A and ImNet-O in ZSL setting. § and † indicate generative and non-generative methods respectively. The best and the second best results are marked in bold and underlined respectively. The Hit@2 and Hit@5 of ZSL-ABP are omitted due to its KNN-based classifier during prediction.

	Methods	ImNet-A			ImNet-O		
		Hit@1	Hit@2	Hit@5	Hit@1	Hit@2	Hit@5
†	DeViSE [5]	14.42	20.08	39.34	14.52	22.79	41.63
	CONSE [19]	20.28	32.58	48.64	12.41	23.30	**86.82**
	SAE [13]	18.84	32.42	50.92	14.84	26.83	51.47
	SYNC [2]	20.52	35.86	59.97	18.58	33.99	65.48
	GCNZ [24]	31.62	60.19	93.45	30.05	50.19	84.50
	DGP [10]	33.07	60.34	**93.49**	31.23	50.32	85.82
§	GAZSL [28]	20.57	35.82	60.55	19.40	35.93	68.58
	LisGAN [16]	21.00	34.44	59.29	20.20	33.34	61.98
	ZSL-ABP [29]	22.05	-	-	22.18	-	-
	KG-GAN	**39.24**	**63.66**	93.10	**34.65**	**55.15**	86.48

on Hit@1 and by 3.32% on Hit@2. On ImNet-O, the performance is improved by 3.42% and 4.83% on Hit@1 and Hit@2 respectively. It is widely believed that the Hit@1 and Hit@2 are relatively more important [25], because these two metrics indicate that the models are capable of predicting the label to it right position more accurately.

From these results, we can observe that these generative methods perform better than most mapping-based ones except for GCNZ and DGP. These mapping-based methods process the prediction of unseen testing samples via the mutual mapping between semantic space and visual space, while those generative methods directly generate training samples for unseen classes conditioned on class semantics. In mapping-based methods, we note that the performance of methods whose mappings are from visual space to semantic space, e.g., DeViSE, CONSE and SAE, is much lower than those from semantic space to visual space, e.g., SYNC. This indicates that the mapping from visual space to semantic space is less competitive due to the *hubness* problem. As for GCNZ and DGP, they are KG-based methods which take the hierarchical relationship of classes as auxiliary semantics to assist the mapping from class name word vectors to visual features, and have superior performance than other baselines that only take class name word embeddings as class semantics. We can conclude that the performance of ZSL can be significantly improved with the enrichment of class semantics, especially when we introduce knowledge graph that contains various class side information into generative ZSL model, the best results are achieved.

GZSL Setting We further evaluate the results of generalized ZSL in Table 3, from which we can draw a similar conclusion as Table 2. Our KG-GAN performs better than listed methods and obtains significant outperformance on the prediction of unseen testing samples (H_u) and harmonic mean value (H), which means our KG-GAN has a better generalized capability. However, we notice that the performance of mapping-based methods dramatically drop. Methods

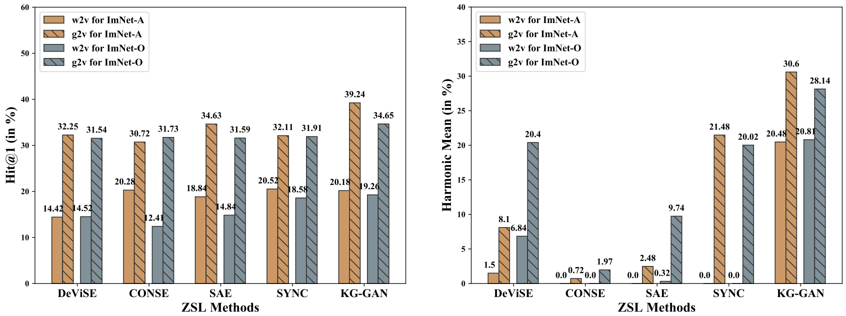
Table 3. The Hit@1 (i.e., accuracy %) results of generalized ZSL. H is the harmonic mean of per-class accuracy on seen classes and unseen classes. § and † indicate generative and non-generative methods respectively. The best and the second best results are marked in bold and underlined respectively.

	Methods	ImNet-A			ImNet-O		
		H_s	H_u	H	H_s	H_u	H
†	DeViSE [5]	62.64	0.76	1.50	64.00	3.61	6.84
	CONSE [19]	86.40	0.00	0.00	62.00	0.00	0.00
	SAE [13]	86.48	0.00	0.00	**92.60**	0.16	0.32
	SYNC [2]	**88.72**	0.00	0.00	62.53	0.00	0.00
	GCNZ [24]	49.12	15.85	23.96	44.60	14.48	21.87
	DGP [10]	44.72	17.92	25.59	47.40	19.00	27.13
§	GAZSL [28]	86.56	1.28	2.52	86.80	6.16	11.50
	LisGAN [16]	39.84	13.82	20.52	35.00	13.87	19.87
	ZSL-ABP [29]	53.60	12.11	19.75	49.20	12.81	20.33
	KG-GAN	39.28	**25.06**	**30.60**	43.40	**20.82**	**28.14**

like CONSE and SYNC even drop to 0.00 on two datasets. This illustrates that these methods have strong bias towards seen classes during prediction, i.e., the model tends to predict the unseen testing samples with labels from seen class set even if the label space contains unseen classes, which may be because these models overfit the training data of seen classes and can not generalize well on unseen classes. By contrast, these generative methods weaken this trend and arise the performance on unseen classes. We also find that although our framework does not achieve the best results on the prediction of seen testing samples (H_s), it still accomplishes comparable performance with the state-of-the-arts. This motivates us to explore optimized algorithms to predict unseen testing samples correctly as well as maintain the high accuracy on seen classes.

4.3 Class Semantics Analysis

To validate the superiority of our knowledge graph based class semantics, we replace the class embeddings of mapping-based methods (i.e., class name word vectors) with the class embeddings learned in our model (cf. Section 3.2) and retrain these models to predict unseen testing samples. In Fig. 3, we report the comparison results of mapping-based baselines and our KG-GAN. We find that the performance of all baselines has a significant improvement no matter what mapping direction the model is. All of these methods have a more than 10% increment on two datasets in ZSL setting, and more than 6% in GZSL setting, especially for SYNC whose harmonic mean value increases from 0% to 21.48% on ImNet-A and from 0% to 20.02% on ImNet-O. On the other hand, our KG-GAN taking class name word embeddings as inputs expectedly performs worse due to the limited class semantics. We also enhance the class semantics in previous KG-based methods (i.e., GCNZ and DGP). Specifically, we add attribute nodes produced in our method into the KG they used to learn unseen classifiers. As a result, taking DGP as an example, its performance is improved by 3.21% on ImNet-A and by 2.72% on ImNet-O respectively in ZSL setting, and improves



(a) ZSL

(b) Generalized ZSL

Fig. 3. Performance comparison of mapping-based baselines and KG-GAN in different class semantics setting. w2v: word2vec based class embedding; g2v: knowledge graph based class embedding. In ZSL setting, we report the Hit@1 results.

5.26% on ImNet-A and 0.93% on ImNet-O in GZSL setting. To sum up, the knowledge graph which involves rich semantics about seen and unseen classes is of great advantage for ZSL problem, and class embeddings learned from it can remedy the weakness of ZSL models to some extent.

4.4 Impact of KG Views

In this subsection, we analyze the contribution of different views of KG for learning class embedding. Specifically, we separately take class embeddings of different views, i.e., the *class*-view class embedding g^c (denoted as GC) and *attribute*-view class embedding g^a (denoted as GA), as the input of KG-GAN to synthesize visual features, and evaluate the quality of generated features of different views. We present the results in the last line of Table 4, from which we can see that the *attribute*-view class embedding has superior performance compared with the *class*-view one in ZSL and GZSL setting. The higher performance may be due to (i) the attribute annotations describe discriminative visual characteristics of class object, enabling different categories can be distinguished in ZSL model, especially for those having similar appearance; (ii) our proposed datasets are fine-grained, which contains some sibling classes whose differences in taxonomy are not obvious such as *horse* and *zebra* in Fig. 1, making it difficult to differentiate the testing samples of these classes when only considering the *class*-view class embedding. However, when combining these two class embeddings, we can achieve impressive prediction results, illustrating that the different views of class semantics from our knowledge graph is meaningful and complementary.

4.5 Regularizers or Class Semantics?

In the literature of generative ZSL methods, some of them incorporate well-designed regularizers with GAN to improve the quality of synthesized features.

Table 4. The results of KG-GAN with different class embeddings learned from different views of KG, and the comparison results of generative methods with regularizer or not. GC and GA represent the class embedding from *class*-view KG and *attribute*-view KG respectively, w2v is the class name vector originally used in these baselines, and reg refers to the model optimized with regularization.

		ZSL		Generalized ZSL	
Methods	Setting	ImNet-A	ImNet-O	ImNet-A	ImNet-O
		Hit@1	Hit@1	H	H
GAZSL [28]	w2v + reg	20.57	19.40	2.52	11.50
	GC + reg	32.23	31.52	8.38	24.18
	GA + reg	36.32	33.43	26.82	24.87
LisGAN [16]	w2v + reg	21.00	20.20	20.52	19.87
	GC + reg	31.08	31.27	25.01	24.63
	GA + reg	36.96	33.45	29.43	25.75
KG-GAN	GC	32.95	32.18	26.06	26.50
	GA	36.82	34.22	29.18	27.09
	GC+GA	**39.24**	**34.65**	**30.60**	**28.14**

For example, in our baselines, GAZSL exploits visual pivot to regularize the mean of generated features of each class to be the mean of real feature distribution; LisGAN regularizes the generated samples to be close to the soul samples, which represents the multi-view prototype representation of real samples. In contrast, our KG-GAN only relies on rich class semantics to synthesize discriminative features for classes without any regularization. To compare the effectiveness of regularizers and class semantics in generating high-quality samples, we input two class embeddings learned from two-views KG (i.e., GC and GA) into the above generative models, and analyze the prediction results in ZSL and GZSL setting. As Table 4 shows, our method combining GC and GA outperforms the baselines combining GC (GA) and regularizers. This indicates that rich class semantics may be more important than optimized regularizers for generative ZSL model.

5 Conclusions

In this paper, we propose to leverage a knowledge graph with two views as class semantics to synthesize sample features in generative zero-shot learning using Generative Adversarial Networks. Our method KG-GAN achieves promising results on different datasets from ImageNet, outperforming the state-of-the-art in both none generative ZSL and generative ZSL. It uses Graph Auto-Encoder to learn semantically meaningful class embeddings, which are superior than traditional class name word vectors and class hierarchical relationship in semantics, and are superior than well-designed regularization in generating discriminative visual features. This inspires us to explore more potential side information and build richer knowledge graphs to tackle the problem of ZSL, especially for improving the performance on both seen and unseen classes.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
2. Changpinyo, S., Chao, W., Gong, B., Sha, F.: Synthesized classifiers for zero-shot learning. In: CVPR. pp. 5327–5336. IEEE Computer Society (2016)
3. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. IEEE Computer Society (2009)
4. Dinu, G., Baroni, M.: Improving zero-shot learning by mitigating the hubness problem. In: ICLR (Workshop) (2015)
5. Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., Mikolov, T.: Devise: A deep visual-semantic embedding model. In: Advances in neural information processing systems. pp. 2121–2129 (2013)
6. Fu, Y., Hospedales, T.M., Xiang, T., Gong, S.: Transductive multi-view zero-shot learning. IEEE Trans. Pattern Anal. Mach. Intell. **37**(11), 2332–2345 (2015)
7. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: NIPS. pp. 5767–5777 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778. IEEE Computer Society (2016)
9. Huang, H., Wang, C., Yu, P.S., Wang, C.: Generative dual adversarial network for generalized zero-shot learning. In: CVPR. pp. 801–810. Computer Vision Foundation / IEEE (2019)
10. Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., Xing, E.P.: Rethinking knowledge graph propagation for zero-shot learning. In: CVPR. pp. 11487–11496. Computer Vision Foundation / IEEE (2019)
11. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
12. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
13. Kodirov, E., Xiang, T., Gong, S.: Semantic autoencoder for zero-shot learning. In: CVPR. pp. 4447–4456. IEEE Computer Society (2017)
14. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 951–958. IEEE (2009)
15. Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. IEEE transactions on pattern analysis and machine intelligence **36**(3), 453–465 (2013)
16. Li, J., Jing, M., Lu, K., Ding, Z., Zhu, L., Huang, Z.: Leveraging the invariant side of generative zero-shot learning. In: CVPR. pp. 7402–7411. Computer Vision Foundation / IEEE (2019)
17. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
18. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
19. Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G.S., Dean, J.: Zero-shot learning by convex combination of semantic embeddings. arXiv preprint arXiv:1312.5650 (2013)
20. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543. ACL (2014)
21. Radovanovic, M., Nanopoulos, A., Ivanovic, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. J. Mach. Learn. Res. **11**, 2487–2531 (2010)

22. Sariyildiz, M.B., Cinbis, R.G.: Gradient matching generative networks for zero-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2168–2178 (2019)
23. Shigeto, Y., Suzuki, I., Hara, K., Shimbo, M., Matsumoto, Y.: Ridge regression, hubness, and zero-shot learning. In: ECML/PKDD (1). Lecture Notes in Computer Science, vol. 9284, pp. 135–151. Springer (2015)
24. Wang, X., Ye, Y., Gupta, A.: Zero-shot recognition via semantic embeddings and knowledge graphs. In: CVPR. pp. 6857–6866. IEEE Computer Society (2018)
25. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly. IEEE Trans. Pattern Anal. Mach. Intell. **41**(9), 2251–2265 (2019)
26. Xian, Y., Lorenz, T., Schiele, B., Akata, Z.: Feature generating networks for zero-shot learning. In: CVPR. pp. 5542–5551. IEEE Computer Society (2018)
27. Zhang, L., Xiang, T., Gong, S.: Learning a deep embedding model for zero-shot learning. In: CVPR. pp. 3010–3019. IEEE Computer Society (2017)
28. Zhu, Y., Elhoseiny, M., Liu, B., Peng, X., Elgammal, A.: A generative adversarial approach for zero-shot learning from noisy texts. In: CVPR. pp. 1004–1013. IEEE Computer Society (2018)
29. Zhu, Y., Xie, J., Liu, B., Elgammal, A.: Learning feature-to-feature translator by alternating back-propagation for generative zero-shot learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9844–9854 (2019)