Explainable Zero-shot Learning via Attentive Graph Convolutional Network and Knowledge Graphs

Yuxia Geng [a,*], Jiaoyan Chen [b], Zhiquan Ye [a], Wei Zhang [c] and Huajun Chen [a]
[a] *College of Computer Science and Technology, Zhejiang University, Hangzhou, China*
E-mails: gengyx@zju.edu.cn, yezq@zju.edu.cn, huajunsir@zju.edu.cn
[b] *Department of Computer Science, University of Oxford, Oxford, UK*
E-mail: jiaoyan.chen@cs.ox.ac.uk
[c] *Alibaba Group, Hangzhou, China*
E-mail: lantu.zw@alibaba-inc.com

Abstract. Zero-shot learning (ZSL) which aims to deal with new classes that have never appeared in the training data (i.e., unseen classes) has attracted massive research interests recently. Transferring of deep features learned from training classes (i.e., seen classes) are often used, but most current methods are black-box models without any explanations, especially textual explanations that are more acceptable to not only machine learning specialists but also common people without artificial intelligence expertise. In this paper, we focus on explainable ZSL, and present a knowledge graph (KG) based framework that can explain the feature transferring in ZSL in a human understandable manner. The framework has two modules: an attentive ZSL learner and an explanation generator. The former utilizes an Attentive Graph Convolutional Network (AGCN) to match inter-class relationship with deep features (i.e., map *class knowledge* from WordNet into classifiers) and learn unseen classifiers so as to predict the samples of unseen classes, with impressive (important) seen classes detected, while the latter generates human understandable explanations of the feature transferability with *class knowledge* that are enriched by external KGs, including a domain-specific Attribute Graph and DBpedia. We evaluate our method on two benchmarks for animal recognition. Augmented by class knowledge from KGs, our framework makes high quality explanations for the feature transferability in ZSL, and at the same time improves the recognition accuracy.

Keywords: Zero-shot Learning, Knowledge Graph, Explainable AI, Knowledge-based Learning, Graph Convolutional Network

1. Introduction

Recently, object recognition by deep learning which learns features from abundant samples has gained a lot of successes. For example, it even outperforms human beings on the ImageNet ILSVRC challenges [1]. However, it still suffers from challenges from data collection: when a new class emerges, hundreds of samples are needed for training while their labels are usually hard to acquire. This makes the recognition model less competitive. Therefore, the interest in zero-shot learning is growing rapidly. It focuses on developing deep learning models for those emerging classes without training samples.

Zero-shot learning (ZSL) is widely introduced in image classification tasks (e.g., [2]). It predicts the images of new classes (i.e., unseen classes) that do not exist in the training set by transferring features learned from the training classes (i.e., seen classes). The inspiration is that *a human can recognize new objects through the class knowledge (e.g., description) itself, even without labeled samples*. For example, considering the animal class "*Serval*", even though a human might have never seen samples in the past, s/he would still be able to recognize it based on the description:

*Corresponding author. gengyx@zju.edu.cn.

1570-0844/0-1900/$35.00 © 0 – IOS Press and the authors. All rights reserved

"*Serval*, a kind of animal with a *Cat*-like face and a *Cheetah*-like body" (see Figure 1). With previous recognition experience of *Cat* and *Cheetah*, s/he can easily reason about the specific appearance of *Serval* and identify it correctly.

The general principle of most ZSL algorithms is to represent such class knowledge and utilize inter-class relationship to transfer model parameters such as neural network features from seen classes to unseen classes. Some works (e.g., [3, 4]) leverage the embedding of class names learned from text corpora for transferring e.g., CNN features, while others (e.g., [5, 6]) prefer more complex knowledge like class hierarchy and class attributes. These methods aim at learning and predicting for unseen classes ([3–8]), but are black-box models: the transferability of features is uninterpretable. This not only limits human's trust in the prediction results of ZSL model, considering that ZSL is a machine learning algorithm which recognizes the samples of new classes but has never been trained with the labeled samples of these classes, but also restricts the potential of improving ZSL models, for example, with generated explanations, machine learning specialists will be clear which features are helpful for the learning of unseen classes and which are not so that optimizing the ZSL models by adding needed features or removing inadequate features. Therefore, in this paper, we focus on explaining the feature transferability in ZSL and generating textual explanations which can be understood by not only specialists but also non-specialists.

There have been few works that explain ZSL with human understandable knowledge. As far as we know, the only work that is close to ours is by Selvaraju et al. [9]. They first learn the mapping between class attributes and neurons in deep networks, and then transfer neurons (i.e., features) from seen classes to unseen classes, where the attributes are taken as textual explanations to justify the prediction of unseen classifiers (cf. more in Section 2.3). Such work indicates that it is feasible to explain ZSL by class knowledge such as class attributes. However, this work focuses on grounding the transferred neurons in interpretable semantics, but ignores the feature transferability which is the core of ZSL. Moreover, its method is ad-hoc, only working for predefined class attributes, while our explanation method supports not only attributes but also general knowledge in different formats, coming from external KGs like DBpedia.

In this paper, we propose a KG based framework to explain the feature transferability in ZSL. It first adopts

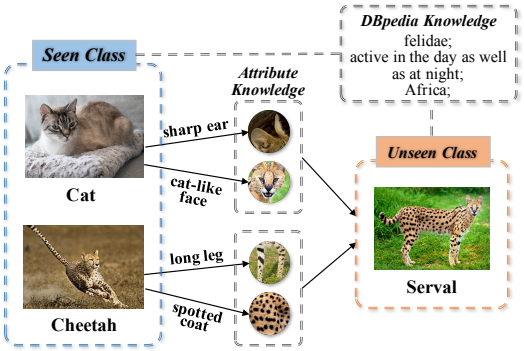


Fig. 1. An example of recognizing *Serval* (unseen class) with two seen classes (*Cat* and *Cheetah*). We focus on explainable ZSL, which captures domain-specific attributes and general knowledge, such as sharp ear, face appearance, long leg, spotted coat and felidae ancestor, as evidence to generate textual explanations that are more understandable by humans.

a KG named WordNet and an Attentive Graph Convolutional Neural Network (AGCN) to model inter-class relationship for ZSL, which is also known as an *Attentive ZSL Learner* (AZSL). Namely, a matching between the inter-class relationship and CNN features is learned. It then uses an *explanation generator* to extract rich class knowledge from a domain-specific Attribute Graph and general external KGs (e.g., DBpedia) as evidence for ZSL explanation. For example, considering the example in Figure 1, the attribute knowledge of *sharp ear* and the DBpedia knowledge of *felidae ancestor* can explain the positive feature transfer from *Cat* and *Cheetah* to *Serval*. Finally, we propose several templates to generate human understandable explanations.

Briefly, our work contributes in the following aspects:

- A KG-based explanation framework for zero-shot learning is proposed. It is among the first to explain the transferability of neural network features in ZSL.
- A novel ZSL algorithm called AZSL is built upon WordNet and AGCN. It models the inter-class relationship and the transferability of CNN features from seen classes to unseen classes, which not only shows improvements over the state-of-the-art baselines, but also enables explaining the transferability of CNN features in ZSL.
- An explanation generator is developed. It can generate ZSL explanations with class semantics from not only domain-specific KGs like Attribute Graph but also general KGs like DBpedia.

– A set of templates are designed to reorganize the transferability explanations and generate human-consumable natural language descriptions for ZSL explanation.
– Lastly, Experiments on two image classification benchmarks are conducted to evaluate the generated explanations and the ZSL learner[1]. The explanation achieves high quality according to the analysis on different metrics and human assessment.

The structure of this paper is as follows. In Section 2, we review the related work. In Section 3, we set up the background of our work. In Section 4, we introduce the details of our KG-based explanation framework, including the attentive ZSL learner in Section 4.2 and the explanation generator in Section 4.3. In Section 5, we display the experiments and the evaluation. Finally, we conclude the paper and discuss some future directions.

2. Related Work

2.1. Zero-shot learning

Zero-shot learning (ZSL) has received a lot of attention in machine learning community. Some work by Larochelle et al. [10] has shown the ability to predict new (unseen) classes of digits that are omitted from the training set, with the features from training (seen) classes being transferred. In computer vision, techniques for utilizing the knowledge of classes to realize the transfer of deep features from seen classes to unseen classes have been investigated [2, 3, 5, 10, 11].

Early algorithms focus on utilizing class attributes to model the semantic relationship of classes [6, 12–14]. For instance, Lampert et al. [6] annotate each class with a set of attributes and propose two attribute-based classification methods, where the features are transferred between seen and unseen classes via attribute sharing. Recent methods prefer to utilize class embeddings trained on class textual descriptions to explore the class semantics [3, 4, 15, 16]. For example, Frome et al. [3] present a *visual-semantic embedding* model, which leverages textual data to model the semantic relationship between classes, and linearly maps the image features into semantic space for feature transferring. However, the state-of-the-art performance in ZSL image classification is achieved by those who utilize KGs for class relationship [5, 17, 18]. For example, Wang et al. [5] use WordNet to model the semantic relationship of hierarchical classes, and encode the relationship using GCN so as to predict classifiers for unseen classes. Considering that the graph convolutional operation in GCN only aggregates the features of first-order neighbors, the authors propose to stack multiple graph convolutional layers (e.g., 6) to propagate features towards distant nodes. While Kampffmeyer et al. [18] propose a dense connection scheme, which connects distant nodes via additional links, to optimize the propagation of features from distant nodes by only using 2 convolutional layers. Following above ideas, we combine class embeddings and class hierarchy as class knowledge to transfer features from seen classes to unseen classes. Different from the GCN-based encoder, we propose to utilize Attentive GCN to encode the inter-class relationship, which can assign different importance to different classes so that augmenting the feature propagation between classes. Moreover, the learned attention weights detect the most contributing seen classes in feature transfer, enabling the feature transferability in ZSL to be explained.

There are also some ZSL methods for dealing with the sample shortage challenge in other domains, such as natural language processing (NLP) including text classification [19, 20], entity linking [21, 22], relation extraction [23] and others [24]. These methods also work on introducing high-level knowledge about labels to conduct feature transfer from seen labels to unseen labels. Notably, NLP data and labels are both symbol-based representation, which leads to benefits in feature transfer, while the the feature transfer learning in our work is more challenging considering the gap between vision and symbol.

2.2. Explainable Artificial Intelligence

Explainable artificial intelligence (AI), which aims to produce interpretable models or predictions, is becoming more and more popular nowadays [25–28]. Such methods enable humans to understand, trust and effectively manage the AI systems and their decisions. Some of the explanation works design white-box and inherently interpretable models like rule-based systems [29], while others try to justify the prediction of a black-box model by for example approximating its behaviour locally with simple interpretable linear mod-

[1] Code and the Attribute Graph are available at https://github.com/genggengcss/X-ZSL.

els [30], or quantifying the contribution of each single input variable [31].

Some explanations target at humans with AI expertise. The generated explanations can be used for system debug so as to efficiently manage and develop machine learning models [32, 33]. While some explanations are for common people without AI expertise such as medical doctors to understand the decisions made by AI-based systems [34]. Most of these works prefer to generate textual explanations, which are more understandable by humans. For example, Biran et al. [35] introduce the linguistic expressions from Wikipedia articles to explain the stock price prediction with natural language sentences. Li et al. [36] propose to generate attributes and captions of images as explanations to indicate whether the system really understands the image content when answering a visual question.

There are also a few works devoted to enrich the explanation with knowledge graphs (KGs), by utilizing human understandable background knowledge and common sense in these KGs, as well as their underlying semantics that can be inferred by reasoning [37, 38]. Tiddi et al. [38] present a framework which exploits Link Data as background knowledge to generate explanations for data clusters. Chen et al. [39] utilize Semantic Web techniques to extract human- understandable evidence from local domain ontologies and external KGs like DBpedia to explain the results of flight delay forecasting.

Another related research direction is to utilize the attention mechanism to make explanations [40, 41]. For example, Yang et al. [40] leverage attention layers to select the words and sentences that have decisive effect on document classification as explanations. Our work also utilizes such attention techniques for ZSL, but goes beyond the attention. It includes a general framework to incorporate semantics from KGs, and generates textual explanations for the core of ZSL – the transferability of deep features.

2.3. Transfer Learning Explanation

ZSL is often regarded as a branch of transfer learning which aims at utilizing samples, features or model parameters learned from one domain to guide the learning in another domain [42, 43]. ZSL algorithms usually transfer features learned by deep neural networks from seen classes (domains with labeled training samples) to predict the testing samples of unseen classes (domains without labeled training samples).

Some works have been proposed to augment transfer learning as well as ZSL with KGs [39, 44–46]. For example, in [46], prior knowledge about the prediction tasks and domains are expressed by ontologies, and further utilized to analyze the transferability of features and samples for augmentation. For another example, Zhang et al. [44] propose a transfer learning based algorithm for long-tail relation extraction, which incorporates the data features from data-rich relations for tackling the prediction of data-poor relations. Knowledge of the relation, which comes from a KG, is investigated to enhance the feature learning for data-poor relations, using KG embeddings and relation hierarchy. In summary, these works indicate the feasibility of studying transfer learning tasks and domains by external knowledge from KGs. In our ZSL study, we not only utilize KGs for performance improvement (i.e., the attentive ZSL learner based on KG and AGCN), but also for human understandable explanations.

Recent studies on transfer learning explanation focus on the analysis of feature transferability [39, 47–49]. For example, Liu et al. [48] assume that the feature is transferable from a source domain to a target domain if the source and target domains have some similar feature structures. For another example, Chen et al. [39] extract knowledge (ontology axioms and DBpedia facts) that co-exist in the source and target domain to explain the transferability of features learned by deep neural networks. These works indicate that the transferability of features is highly related with the knowledge of the source and target domain. In this paper, we also prefer to extract domain knowledge (i.e., class knowledge) to generate explanations for feature transferability. Different from the above works, we on the one hand develop a general framework that can generate explanations from different knowledge resources from multiple KGs such as the domain-specific Attribute Graph and the general DBpedia. On the other hand, we focus on KG-based ZSL – an important and popular transfer learning branch whose current solutions are all black-box models without explanations.

Few works have been found to explain ZSL with human understandable knowledge. The only work we know is by Selvaraju et al. [9]. It first learns a mapping between class attributes and individual neurons in a network, and then predicts unseen neurons based on the attributes of unseen classes to optimize the learning of unseen classifiers. It also generates textual explanations by inversely mapping the transferred neurons to class attributes so as to validate the decisions made by unseen classifiers. The authors focus on ex-

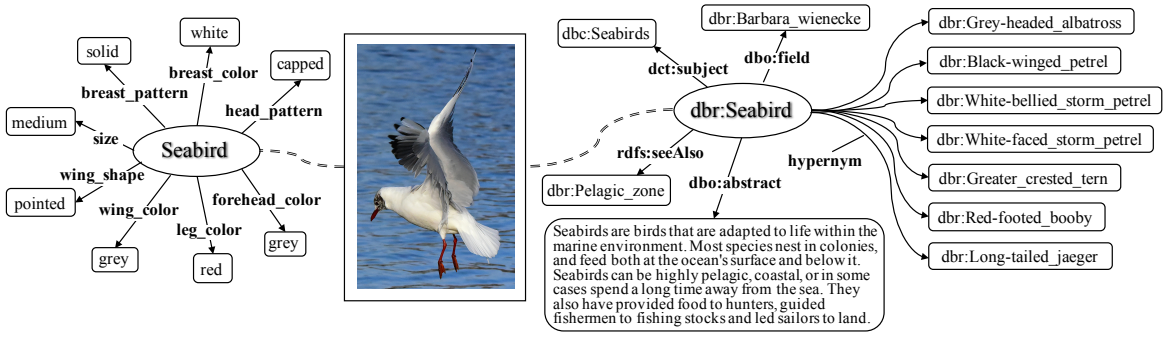


Fig. 2. An example of our two introduced KG resources for animal class *Seabird*. [Left] is the domain-specific Attribute Graph with corresponding entity *Seabird*; [Right] is the general DBpedia with aligned entity *dbr:Seabird*.

plaining the prediction results of unseen classes and grounding the neurons in interpretable semantics, however, ignore the feature transferability in ZSL. By contrast, the explainable ZSL proposed in our work pays attention to the transferability of deep features, which is more important for analyzing the nature of ZSL. In addition, the generation of explanations in [9] relies on the input class attributes, while the explanations we generate contain not only domain-specific attributes but also general knowledge, which are more expressive and flexible compared with the ad-hoc class attributes in [9]. Most importantly, the generation procedure of explanations in our method is relatively independent from the ZSL model, it has no impact on the classification model compared with other explainable methods that need to make a tradeoff between accuracy and interpretability.

3. Preliminaries

3.1. Zero-shot Learning

In zero-shot learning, the training set is denoted as $\mathcal{D}_{tr} = \{(x_i, l_i)\}_{i=1}^{N}$, where N is the number of training samples, x_i represents the i-th training image and l_i is its label. While the testing set is described as $\mathcal{D}_{te} = \{(\tilde{x}_i, \tilde{l}_i)\}_{i=1}^{\tilde{N}}$, and its labels have no overlap with the labels in $\mathcal{D}_{tr}$. We regard the labels in $\mathcal{D}_{tr}$ as *seen classes*, denoted as S, and the labels in $\mathcal{D}_{te}$ as *unseen classes*, denoted as U. Each class involves a unique classifier f for predicting whether a sample is of the class or not. ZSL aims to learn classifiers for unseen classes by transferring features learned from $\mathcal{D}_{tr}$ based on the semantic relationship between seen and unseen classes.

There are usually two prediction settings in ZSL: the standard ZSL setting and the generalized ZSL setting. The former is to predict the labels of testing samples in $\mathcal{D}_{te}$ with candidates from U, while the latter is to predict the testing samples of seen and unseen classes with candidate labels from $S \cup U$. In this paper, in order to investigate the feature transferability from seen classes to unseen classes, we focus on the standard ZSL setting to evaluate the prediction ability of unseen classifiers and generate explanations for them. It is also worth considering how to deal with explainable ZSL in generalized ZSL setting in real-world applications. Maybe we can adopt a two-phase framework – a coarse-grained phase to judge if a testing sample comes from seen classes or unseen classes, and a fine-grained phase to make final predictions, where traditional classifiers (e.g., softmax classifiers) are used to predict its label with candidates from seen class set if the sample is from seen classes, and ZSL classifiers are used to predict its label with candidates from unseen class set if it belongs to unseen classes. We can make further attempts in the future.

3.2. Class Knowledge

In our study, we introduce three kinds of Knowledge Graphs (KGs) to depict the *class knowledge*, which describes the semantic relationship between classes. They are used for feature transfer in ZSL as well as generating explanations. We briefly introduce the three KGs below.

WordNet [50] is a lexical knowledge base for English where nouns, verbs, adjectives and adverbs are organized into sets of synonyms, each representing a lexicalized entity. Semantic relations (hypernym, hyponymy, meronymy, etc.) are used to link these entities. We utilize such a KG to build a hierarchical struc-

ture of classes, by aligning each class with an entity in WordNet. In this structure, the edge that connects two class nodes represents the "subClassOf" relationship.

Attribute Graph is a domain-specific knowledge graph we created by collecting attribute annotations that describe the characteristics of animal classes, especially the visual characteristics such as the color, shape and the important parts of animals. The knowledge about one class from such a special KG is in the form of *key-value*, where *key* represents the attribute item of class and *value* is the corresponding attribute value. We organize the *key-value* knowledge in the format of triples (*c, a, v*), where *c* represents an animal class, *a* and *v* represent *key* and *value* respectively. Taking *Seabird* as an example, as Figure 2 [Left] shows, its attribute annotation "grey wing color" can be described with the triple (*Seabird, wing_color, grey*). We collect these attribute annotations from existing ZSL datasets, e.g., AwA [6] and CUB [51], and Wikipedia descriptions about classes. The constructed KG contains 1,399 classes, 588 attributes and 8,114 triples in total, which is available in our github repository. Note that the scale of Attribute Graph is small compared with other public KGs, we look forward to augmenting it by crowdsourcing in the future. In addition, such attribute annotations are now available in animal recognition task, while in other tasks or domains, they can come from domain knowledge or experts.

DBpedia [52] is a general knowledge graph that includes common sense and background descriptions about classes. With knowledge from Wikipedia encyclopedia, DBpedia is a large scale KG consisting of 4.58 million entities and 3 billion facts. The animal classes in ZSL can be matched to entities of DBpedia. For example, *Seabird* is aligned with entity *dbr:Seabird* as Figure 2 [Right] shows. Different from Attribute Graph, DBpedia contains more general knowledge, including the textual description of entity (i.e., abstract text) from property *dbo:abstract*, the semantics between entities linked by different relations (e.g., *hypernym*), etc. In our experiments, we use public DBpedia SPARQL Endpoint query service, which loads the 2016-10 DBpedia dump, to access the resources (more details at https://wiki.dbpedia.org/public-sparql-endpoint).

In order to provide an overall explanation for ZSL in animal recognition task, we take Attribute Graph as well as DBpedia as external KGs to extract visual knowledge like "red leg color" as well as general knowledge like the descendants of "Seabird" in biology.

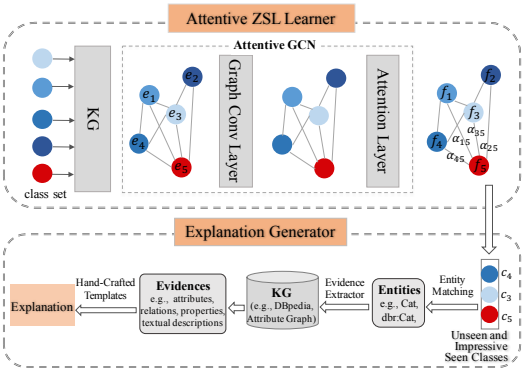


Fig. 3. Our proposed KG-based explainable ZSL framework.

4. Methodology

4.1. Framework Overview

In this paper, we present a KG-based framework to explain the feature transferability in ZSL in a human understandable manner, including an Attentive ZSL learner (AZSL) and an explanation generator (EG), as illustrated in Figure 3. AZSL first models the hierarchical relationship of seen classes, unseen classes as well as their ancestor and descendant classes using WordNet, and then maps this class knowledge into the deep features learned by CNNs, which pursue class discrimination and are core components of a classifier. Specifically, we utilize AGCN to encode the inter-class relationship and match it with CNN classifiers, in which the unseen classifiers are learned by transferring features from seen classifiers. Considering the different contributions of features from different seen classes, we introduce an additional attention layer in AGCN to learn attention weights of seen classes. In this way, we capture the most contributing seen classes for an unseen class so as to explain the feature transferability from seen classes to unseen classes. Based on the learned relationship between seen and unseen classes, EG then aligns both seen and unseen classes with entities of external KGs, from which richer class knowledge, such as entity attributes, relations, properties and textual descriptions, are extracted as evidence to further illustrate the transferability of deep features. It also generates natural language explanations with these evidence using some hand-crafted templates.

4.2. Attentive ZSL Learner

AZSL matches inter-class relationship with deep features to learn classifiers of unseen classes via an

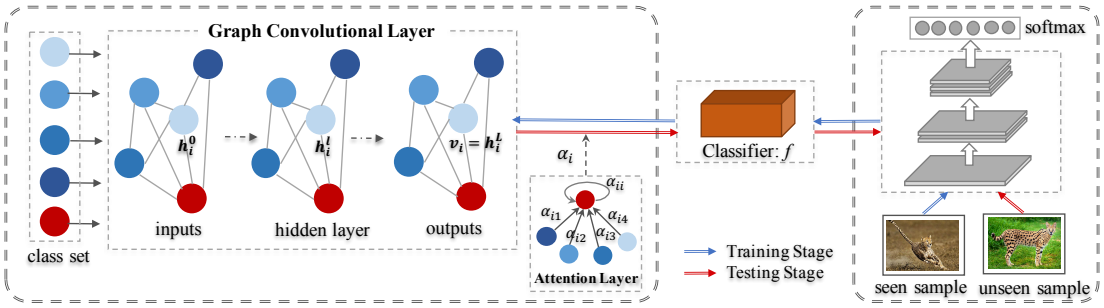


Fig. 4. An overview of our attentive ZSL learner. At training stage, AZSL maps class knowledge into classifiers with multiple graph convolutional layers and an attention layer; At testing stage, AZSL predicts classifiers for unseen classes and conduct nearest neighbor prediction to classify the testing samples of them.

Attentive Graph Convolutional Network (AGCN). As Figure 4 shows, AZSL includes: *(i)* learning a discriminative *classifier* for each class (Figure 4 [Right]); *(ii)* encoding class knowledge into classifiers (Figure 4 [Left]); *(iii)* predicting classifiers for unseen classes.

4.2.1. CNN Classifier

Consider Convolutional Neural Network (CNN), a frequently used network for feature extraction in object recognition, where the significant features of images are extracted to make predictions. Given a object class, we use its images to train a network, the second to the last layer of the network will output a set of class-specific parameters, which can be used to classify the new images of this class. We take these parameters as classifier to represent the discriminative deep features for each class.

In our model, we pre-train classifiers for seen classes with samples from $\mathcal{D}_{tr}$, and use them to train the encoding of class knowledge. After training, the classifiers of unseen classes are predicted to recognize the samples from $\mathcal{D}_{te}$.

4.2.2. Mapping Class Knowledge into Classifier

We utilize Attentive Graph Convolutional Network (AGCN) to encode the graph-structured inter-class relationship. It includes multiple graph convolutional layers and an attention layer.

Graph Convolution Layer is to conduct convolutional operation on graph, which works on propagating information between nodes and capturing the dependence of graph-structured data. In each convolutional layer, the convolution operator computes a node's hidden feature by aggregating features from its neighboring nodes defined in the graph, and updates it to the next layer. Mathematically, given a class node i, its hidden feature at l-th layer is learned as follows:

$$h_i^l = \sigma(W_l \sum_{j \in \mathcal{N}_i} \frac{h_j^{l-1}}{|\mathcal{N}_i|} + B_l h_i^{l-1}) \quad (1)$$

where $\mathcal{N}_i$ is the set of neighboring classes of class i. According to the optimizer in graph convolutional operation, the neighboring classes here means the first-order neighbors of i. W_l and B_l represent the layer-specific trainable weight matrix and bias term, respectively. $\sigma(\cdot)$ denotes an activation function such as LeakyReLU (More details are in [53]).

Stacking the convolutional layer one after another, we can output the feature representation of class i at last layer: $v_i = h_i^L$, with the features of other classes encoded. Motivated by [3, 4], we use pre-trained word embeddings of class names to initialize the class nodes as the input of the first layer. These *embeddings*, learned from large scale corpora ([54]), are semantically meaningful representations of classes. At the same time, we implement the convolutional layer with the models proposed in [5] and [18], in which the different adjacency matrices of graph indicating the class neighbors are defined and different numbers of convolutional layers are proposed, to compare the performance of different graph convolutional operations.

Attention Layer. In the aggregation of graph convolutional layer, we find that different neighboring classes have different effects on the feature learning of a specific class. Therefore, in this paper, we propose to utilize attention mechanism – stacking an attention layer after graph convolutional layers to assign attention weights to the neighboring classes and analyze the different contributions of different classes.

Specifically, for a class i, the attention weight of each neighboring class is computed by the similar-

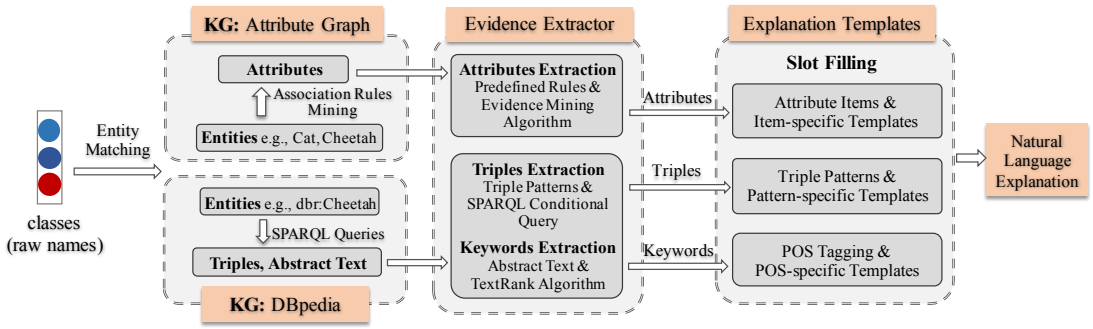


Fig. 5. Illustrating for generating natural language explanation from external Attribute Graph and DBpedia.

ity between its feature vector and v_i, because when a neighboring class contributes more to the class i in the aggregation, their learned feature representations are more similar. For a neighboring class j of class i, its attention weight is computed as:

$$\alpha_{ij} = \frac{exp(cos(v_i, v_j)}{\sum_{k \in \mathcal{N}_i} exp(cos(v_i, v_k))} \tag{2}$$

where $cos(\cdot)$ denotes the cosine similarity, $\mathcal{N}_i$ is the set of neighboring classes of class i, including class i itself. The computed attention weights are used to update the feature vector of class i as:

$$\bar{v}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot v_j \tag{3}$$

It is noted that we stack the attention layer after computing multi-layer graph convolutions as Figure 4 [Left] shows. We also attempted to add attention layer after each graph convolutional layer in the preliminary experiments, however, we found the model is hard to converge, it may be bacause *(i)* the dimension of hidden features in our model (e.g., $2,048$) is large compared with the dimension of hidden features in other graph attention networks (e.g., 8 in [55]), or *(ii)* our model is a regression model rather than a classification model which usually works better.

Training. With learned feature vectors of classes, we use the CNN classifiers of classes as the training supervision to map inter-class relationship into deep CNN features so that predicting a visual classifier for each class node. In other words, the output of AGCN is a series of CNN features.

In ZSL, considering that only seen classes have enough samples to train classifiers, we estimate the parameters of AGCN in a semi-supervised learning manner, and take mean square error as the loss function:

$$\frac{1}{|S|} \sum_{i=1}^{|S|} \mathcal{L}_{MSE}(f_i, \bar{v}_i) \tag{4}$$

where $|S|$ is the number of seen classes, and f_i is the classifier of seen class i.

4.2.3. Predicting Unseen Testing Samples

As previously illustrated, for each unseen class, its deep features can be learned by aggregating features from its neighboring seen classes. Therefore, we can obtain the CNN classifier for each unseen class node using the estimated parameters of AGCN. With predicted unseen classifiers (i.e., $\{\bar{v}_i\}_{i=1}^{|U|}$), we perform nearest neighbor search to predict labels for the testing samples of unseen classes. Specifically, at test time, when a testing image arrives, AZSL first extracts its features using pre-trained CNN, and then these unseen classifiers will be multiplied with the image features to produce classification scores. The class with the highest score (i.e., the nearest one) is the predicted label for this image.

In addition, we also learn contributing seen classes for each unseen class from attention layer, these seen classes have high attention weights and each of them is believed to be important in transferring features to the unseen class. We name them as **impressive seen classes** (IMSCs in short) for this unseen class. By this means, we automatically detect the seen classes which have decisive effects on the feature learning of unseen classes, which is the basis to analyze the transferability of deep features in ZSL.

4.3. Explanation Generator

With the learned relationship between seen and unseen classes, we introduce two external knowledge

graphs, the domain-specific Attribute Graph and the general DBpedia, to extract reliable evidence and generate human understandable explanations for justifying the feature transferability between seen and unseen classes.

The explanation generating procedure is illustrated in Figure 5. Briefly, we *(i)* match raw class names with entities of external KGs; *(ii)* extract supported evidence using different strategies for different external KGs; and *(iii)* generate natural language explanations with some templates.

4.3.1. Domain-specific KG: Attribute Graph

Considering that the ZSL classes are naturally matched with the entities in Attribute Graph, we first begin with extracting evidence. In Attribute Graph, the evidence refers to the common attributes between unseen classes and their impressive seen classes. However, the searching space is often large for finding the common attribute set, especially when multiple impressive seen classes exist. To this end, we develop a rule-mining based method to find out the common attributes by mining association rules of classes from Attribute Graph, with an algorithm named *EvidenceMining* proposed. The mined rules illustrate the semantic association between seen and unseen classes, which can validate the rationality of the learned relationship between them. The supporting set of a rule is the common attributes shared by these classes, which are desired evidence to explain the feature transferability from seen classes to unseen classes.

Association rule mining is a widely used algorithm in Data Mining. It was first proposed for mining the association rules of shopping items from a list of customer transactions [56]. Each transaction consists of a set of items purchased by a customer in a visit. An association rule of items is like $\{bread \Rightarrow milk\}$, meaning that people who purchase *bread* usually also purchase *milk*. In the context of mining association rules of classes, a transaction is defined as a set of classes that have the same attribute. The rule of classes means that these classes are associated because they share a set of attributes.

Let $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ and $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ be the set of attributes and the set of classes of Attribute Graph $\mathcal{G}$ respectively. Let $\mathcal{D}$ be a set of transactions, where each transaction is labeled with an attribute a_i and consists of a set of classes C both having attribute a_i. An association rule is an implication of the form $\{X \Rightarrow Y\}$, where X and Y are sets of classes, $X \subset \mathcal{C}$, $Y \subset \mathcal{C}$, and $X \cap Y = \emptyset$. The rule $\{X \Rightarrow Y\}$ holds in the transaction set $\mathcal{D}$ when $c\%$ of attributes in $\mathcal{D}$ owned by X are also owned by Y; here $c\%$ is called the *confidence* of the rule. The *support* of the rule $\{X \Rightarrow Y\}$ is the ratio of attributes that are owned by both X and Y. The support set of the rule is a series of common attributes of the classes in $X \cup Y$. Regarding the unseen class u and its impressive seen class set $S = \{s_1, ..., s_n\}$, the potential association rule of them can be predefined as:

$$\{s_1\} \Rightarrow \{u\}$$
$$...$$
$$\{s_n\} \Rightarrow \{u\}$$
$$\{s_1, ..., s_n\} \Rightarrow \{u\}$$

Take unseen class *grizzly bear* and its impressive seen classes *polar bear* and *raccoon* as an example. Let $\mathcal{C} = \{polar\ bear, raccoon, grizzly\ bear\}$ and $\mathcal{A} = \{claws, black, furry\}$. Consider the transaction database $\mathcal{D}$ shown in Table 1. The frequent class sets (i.e., the sets of classes with minimum support value) are mined so as to produce the rules of these classes. As Table 1 shows, we can mine the association rule "$\{polar\ bear, raccoon\} \Rightarrow \{grizzly\ bear\}$", as well as its support set: $\{claws, furry\}$, which are common attributes shared by the three classes and can be taken as evidence for validating the transferability from *polar bear* and *raccoon* to *grizzly bear*.

Table 1
Example of mining association rules of classes *polar bear, raccoon* and *grizzly bear*.

(a). Database $\mathcal{D}$

Transaction Label	Class Items
claws	polar bear, raccoon, grizzly bear
black	raccoon, grizzly bear
furry	raccoon, grizzly bear, polar bear

(b). Frequent Class sets

Class set	Support Set (Attributes)
$\{polar\ bear\}$	claws, furry
$\{raccoon\}$	claws, black, furry
$\{grizzly\ bear\}$	claws, black, furry
$\{polar\ bear, grizzly\ bear\}$	claws, furry
$\{raccoon, grizzly\ bear\}$	claws, black, furry
$\{polar\ bear, raccoon\}$	claws, furry
$\{polar\ bear, raccoon, grizzly\ bear\}$	claws, furry

(c). Rules

Rule	*support*	*confidence*
$\{polar\ bear\} \Rightarrow \{grizzly\ bear\}$	66.6%	100%
$\{raccoon\} \Rightarrow \{grizzly\ bear\}$	100%	100%
$\{polar\ bear, raccoon\} \Rightarrow \{grizzly\ bear\}$	66.6%	100%

Algorithm 1 Evidence Mining

Input: Attribute Graph $\mathcal{G}$; Unseen class u and its impressive seen class set S; Minimum support and confidence value s_{min}, c_{min}; Predefined rule set R;
Output: A: explanatory evidence for u and S;

$\mathcal{C} = \{u, s_1, ..., s_n\}$; % *Prepare class set*
R_u = Instantiate($R, \mathcal{C}$); % *Prepare rule set for u*
$\mathcal{A} = \emptyset$; % *Init. of attribute set*
for each class $c \in \mathcal{C}$ **do**
    % *Get attributes of c*
    $\mathcal{A}_c$ = ExtractAttribute($\mathcal{G}$, c);
    Append $\mathcal{A}_c$ to $\mathcal{A}$;
end for
$\mathcal{D}$ = ConstructDataset($\mathcal{G}, \mathcal{C}, \mathcal{A}$); % *Store Trans.*
$k = n + 1$; % *Size of frequent class set*
% *Get frequent class set and support value*
F, v_s = **Apriori**($\mathcal{D}, k, s_{min}$);
% *Filtering for confident rule set*
R'_u, v_c = Filter(F, R_u, c_{min});
% *Output the support sets of rules as evidence*
A = Get($\mathcal{D}, R'_u$);
return A;

Algorithm 1 illustrates the mining pseudocode of rules and common attributes. Given an unseen class and its impressive seen classes, we first instantiate the predefined rules and extract attributes of each class from Attribute Graph $\mathcal{G}$ to construct the transaction database as shown in Table 1(a). Then, we apply Apriori algorithm [57] to mine the frequent class sets with minimum support value. Constraining with predefined rules and minimum confidence value, we filter frequent class sets to get the association rules of classes, and finally get the support sets of rules (i.e., common attributes) as evidence to output.

Thus, we not only mine the association rules of seen and unseen classes with some measurements produced, e.g., *support* and *confidence*, but also extract common attributes from Attribute Graph as evidence to explain the feature transferability between classes.

4.3.2. General KG: DBpedia

Match ZSL class with DBpedia Entity. Different from Attribute Graph whose entities can be directly aligned with ZSL classes by name, the matching between DBpedia entities and ZSL classes are more challenging due to the ambiguity. One widely used and effective approach is lexical matching, with an index on the entity's name, label, anchor text (description), etc. In our paper, we use the DBpedia Lookup service[2], which is based on the index of DBpedia Spotlight [58]. Specifically, we use raw class names as keywords to look up the corresponding DBpedia entities. For example, the entity "*dbr:Cheetah*" can be looked up by the name string "*Cheetah*".[3].

One challenge of class to entity matching is that only a part of all the classes have entity correspondences. On the one hand, the corresponding entity of the class does not exist in the KG. For example, DBpedia only has an entity for *Chicken* but no *Cock* and *Hen*. The latter two however have totally different visual features. On the other hand, some classes are wrongly matched to entities. For example, *Red fox* is incorrectly matched with entity *dbr:Fox*, while the correct matching should be *dbr:Red_fox*. To ensure the correctness of class to entity matching, we manually check the matching results and remove the incorrect ones from the matching set. There are also some algorithms developed to automatically evaluate the entity matching results, for example, the string-similarity-based methods [59, 60] and the embedding-based methods [61, 62] in entity alignment tasks. However, considering that these methods have some limitations and errors in real-world applications (i.e., relying on well-designed matching patterns or labeled entity pairs), and on the other hand our work focuses on studying the interpretability in ZSL, we decide to manually check all matching results. It is expected that we can develop some methods to release the pressure of manual verification in the future.

Different from the attribute annotations of classes in Attribute Graph, the knowledge about classes in DBpedia is massive and diverse. Therefore, with matched entities, we utilize SPARQL queries[4] to retrieve two kinds of evidence: 1) **abstract text** which is an overall description of entity with keywords included, and 2) structured **triples** which describe fine-grained semantics of an entity, e.g., properties and relations with other entities. We first begin with extracting triples.

Extract Triples. Two kinds of triples are extracted: *(i)* object triple, denoted as (h, r, t), where h is the head entity, t is the tail entity, and r is the relation; *(ii)* property triple, denoted as (h, p, v), where h is the head entity, p is the data property and v is the data value (literal). From these triples, we can find some correlations

[2]https://github.com/DBpedia/lookup
[3]dbr, dbo, etc. are URI prefixes in DBpedia. Please see http://DBpedia.org/sparql?help=nsdecl.
[4]https://www.w3.org/TR/rdf-sparql-query/

Table 2

Triple patterns and corresponding SPARQL query items, where s, u represent entity patterns corresponding to seen and unseen entities respectively, $\wedge$ represents the joint operator of *patterns*.

Triple Pattern	Diagram	Query Item	Illustration
(s, r, u)		SELECT ?r WHERE $\{s\ ?r\ u.\}$	s is directly related with u via relation r.
(u, r, s)		SELECT ?r WHERE $\{u\ ?r\ s.\}$	u is directly related with s via relation r.
$(s, r_1, t) \wedge (u, r_2, t)$		SELECT ?r_1 ?r_2 ?t WHERE $\{s\ ?r_1\ ?t.\ u\ ?r_2\ ?t.\}$	s and u both connect with entity t via relation r_1 and r_2, respectively (r_1, r_2 may refer to the same relation).
$(s, p, v) \wedge (u, p, v)$		SELECT ?p ?v WHERE $\{s\ ?p\ ?v.\ u\ ?p\ ?v.\}$	s and u both have property p and share the same property value v.
$(s, r_1, t) \wedge (t, r_2, u)$		SELECT ?r_1 ?r_2 ?t WHERE $\{s\ ?r_1\ ?t.\ ?t\ ?r_2\ u.\}$	s and u is related via a transitional entity t.

between seen and unseen entities, which can be taken as evidence to illustrate the feature transferability from seen to unseen classes. However, only a portion of all the triples of an entity are useful for describing the correlations. Thus, we need a method to extract valid triples efficiently.

To this end, we design some *triple patterns* for extraction, as shown in Table 2. Based on these patterns, the SPARQL queries are developed to retrieve the evidence, including relations or entities that associate seen and unseen entities, and common properties shared by seen and unseen entities. Considering the example in Figure 1, *Cat* and *Serval* share the same ancestors *Felidae*. The fact can be verified by two triples *(dbr:Cat, hypernym, dbr:Felidae)* and *(dbr:Serval, hypernym, dbr:Felidae)*, which are extracted according to the pattern $((s, r_1, t) \wedge (u, r_2, t))$. Note that some extracted triples may be not accurate enough. For example, the object entity *dbc:Birds_of_Europe* in the triple *(dbr:Ruddy_Turnstone, dbo:family, dbc:Birds_of_Europe)* is too broad in terms of bird species. This may bring useless knowledge, and is a factor that impacts the quality of explanations.

Extract Keywords from Abstract Text. Apart from the evidence from triples, each entity in DBpedia has a textual description with a set of sentences (i.e., abstract text), which is the data value of the property *dbo:abstract* and thus can also be accessed by SPARQL queries. From the abstract text, we can obtain some informative descriptions of class, especially those about visual characteristics. However, some descriptions in abstract text are lacking information: e.g., a sentence, "Dogs perform many roles for people, such as hunting, herding, protection, assisting police and military, companionship and, more recently, aiding handicapped individuals", describes the social background of *dbr:Dog*, words and phrases in this sentence do not mention much useful knowledge about *Dog*.

Therefore, we adopt TextRank [63], an unsupervised automatic summarization algorithm, to extract keywords from the abstract text. The extracted words and phrases are core and descriptive to represent the class-specific properties so that illustrating the knowledge shared between entities. For example, the extracted keyword *Africa* in Figure 1 illustrates the same living environment of *Cheetah* and *Serval*.

4.3.3. Template-based Explanation Generator

Aforementioned extracted items, including attributes, triples and keywords from Attribute Graph and DBpedia, constitute the fine-grained class knowledge that can be used as evidence to explain the transferability between seen and unseen classes. In order to make them more understandable, we feed them into handcrafted templates to generate natural language explanations.

Inspired by *Slot Filling*, a popular method of completing information in dialogue system, we design templates with entities (classes), attributes, relations, properties and keywords as slots, and take the extracted items as values to fill in. We design three different kinds of templates in total, as shown in Table 3, for the structured triples, unstructured attributes and keywords respectively.

Attributes are domain-specific descriptions used for annotating objects. Attributes of the same attribute item describe the same aspect of object. For example, attributes like *head*, *tail*, *claws* and *leg* describe the

Table 3

Explanation templates for generating textual explanation. A overall illustration is first provided. ***s*,*u*,*t*,*r*,*p*,*a***, **ADJ**, etc. are slots in templates to be filled. The Left is for attributes, where we list part of all attribute items. The Center is for structured triples. The Right is for textual keywords, where we list all possible Parts-of-Speech (POSs) of keywords, including adjective (ADJ), noun (NOUN) and *named entity* (e.g., LOC). Notably, the POS tags attached to the attribute item (e.g., *coat (ADJ)*) mean the attribute values with different POSs.

Overall Illustration: The prediction for samples of $\boldsymbol{u}$ is supported by $\boldsymbol{s}$.

Attributes & Templates		**Triples & Templates**		**Keywords & Templates**	
Attribute Item	**Template**	**Triple Pattern**	**Template**	**POS of Keyword**	**Template**
color, size, species, coat (ADJ), ...	They are both ***a***.	$(s,r,u),(u,r,s)$	$\boldsymbol{s(u)}$ is $\boldsymbol{r}$ of $\boldsymbol{u(s)}$.	ADJ	They are both **[ADJ]** animal/bird/other.
body part, shape, coat (NOUN), ...	They both have ***a***.	$(s,r_1,t)\wedge(u,r_2,t)$, $(s,r_1,t)\wedge(t,r_2,u)$	***s*** and ***u*** are both relevant to ***t*** via relation $\boldsymbol{r_1}$, $\boldsymbol{r_2}$. (or ***s*** and ***u*** are both a member of ***t***.)	NOUN	They both have **[NOUN]**. (or they are similar in **[NOUN]**.)
feeding, habitat	They both eat (or live in) ***a***.	$(s,p,v)\wedge(u,p,v)$	***s*** and ***u*** share the same ***v*** of property ***p***.	ADJ+NOUN	***s*** and ***u*** are similar in **[ADJ+NOUN]**. (or ***s*** and ***u*** both have **[ADJ+NOUN]**.)
behaviour, habits	They both behave (or like) ***a***.	$(s_1,r_1,u)\wedge(s_2,r_2,u)$	$\boldsymbol{s_1}$,$\boldsymbol{s_2}$ both belong to ***u***. (or $\boldsymbol{s_1}$,$\boldsymbol{s_2}$ are both species of ***u***.)	*named entity* (LOC, GPE)	They both live in **[LOC].**

body parts of animals, which belong to the attribute item *body part*, while others like *red*, *green* and *blue* describe the appearance color of animal, which belong to the attribute item *color*. Considering that the attributes of the same attribute item can be expressed in similar sentences, we design templates based on the different items, which are fed by attributes extracted from Attribute Graph. For example, the common attribute *sharp ear* of *Cat* and *Serval*, which belongs to the attribute item *body part*, can be expressed with the sentence: "They both have sharp ear". Table 3 lists some of the items and their corresponding templates. It is noted that the number of extracted common attributes varies a lot. In order to restrict the length of generated sentences and avoid excessively repetitive expressions in the generated explanations, we randomly select 10 attributes to describe the knowledge between seen and unseen classes when the number of common attributes exceeds 10.

Structured triples follow some fixed formats, especially for those extracted based on identical *triple patterns*. Thus, we design templates to textualize triples according to the *triple patterns* as shown in the center of Table 3. The entities, properties and relations in the extracted triples are used as values to fill the corresponding slots in templates. Taking triples *(dbr:Cat, hypernym, dbr:Felinae)* and *(dbr:Serval, hypernym, dbr:Felinae)*, which are extracted for *Cheetah* and *Serval* with *pattern* "$(s,r_1,t)\wedge(u,r_2,t)$", as examples, the following textual explanation can be generated: "*Cat* and *Serval* are both relevant to *Felinae* via relation *hypernym*". Notice that the DBpedia prefixes such as "dbr" in the triple will be removed when generating sentences.

Keywords extracted from the abstract text are natural expressions consisting of adjectives, nouns, their combinations and so on. One example is *spotted coat*. Therefore, we design templates based on different parts of speech of keywords and utilize *POS Tagging* to generate natural language sentences. Specifically, each keyword is first labeled with a Part-of-Speech (POS) tag, and then the keyword will be filled into the corresponding template which has the same POS slot. For example, the abstract text of *Cheetah* and *Serval* both involves the keyword *spotted coat* labeled with an ADJ-NOUN tag, we can refer to the template with ADJ-NOUN slot to generate the sentence: "They are similar in spotted coat". Additionally, some nouns have special meanings (i.e., *named entities*), for example, *Africa* which describes the habitat of *Serval* is a location noun. Therefore, we also recognize the named entities in keywords and design corresponding templates. More templates are shown in the right of Table 3. In our experiments, we utilize SpaCy, a frequently used natural language processing toolkit, to identify named entities in the text and assign POS tags for keywords. It can return the results of NER and tagger simultaneously by only loading model once. Moreover, in order to generate more diverse expressions, we enrich the word in templates with its synonyms from WordNet [64].

5. Evaluation

We conduct experiments on image classification task and evaluate our framework in the following aspects: (1) accuracy of our attentive ZSL learner (**AZSL**) in comparison with the state-of-the-art ZSL baselines; (2) quantitative analysis of the feature transferability; (3) evaluation on the textual explanations of transferability, including human scoring, qualitative analysis and case studies. Based on the evaluation, we also discuss the relationship between feature transferability and ZSL prediction performance.

5.1. Experiment Setting

5.1.1. Datasets

Two widely used image sets are adopted: Animals with Attributes (AwA) [6] and ImageNet [65]. AwA is a coarse-grained dataset, while ImageNet is diverse in terms of granularity, i.e. it contains a collection of fine-grained datasets, e.g. different vehicle types, as well as coarse-grained datasets. Each ImageNet class corresponds to an entity of WordNet. For each dataset, we split the classes into two disjointed parts – *seen classes* and *unseen classes* as in [17]. The former have training samples (images) while the latter have no training samples but are semantically related to the former. Specifically, in ImageNet, 398 animal classes are used as seen classes, each of which contains around $1,000$ images, while the classes that are one-hop away from the seen ones in WordNet are taken as unseen classes. In AwA, 40 classes are used as seen classes and 10 as unseen classes.

Considering that AwA unseen classes are contained in the ImageNet unseen set and several of the seen classes (24 out of 40) overlap with the ImageNet seen set, we leverage WordNet to build a universal hierarchical graph for experiments on two datasets, in which seen classes and unseen classes of the two datasets as well as their ancestors, descendants and siblings are connected via "subClassOf" relation edge. The total number of graph nodes is $3,969$. It is noted that we train different AZSL models for different datasets considering the different granularity and data distribution of datasets, although AwA classes have overlap with ImageNet classes.

Specially, considerring ImageNet contains not only coarse-grained subsets but also fine-grained subsets, the density of the connection between seen classes and unseen classes varies a lot: some seen (unseen) classes whose first-order neighbors contain multiple (e.g., 5) unseen (seen) classes (i.e., dense connection), while some seen (unseen) classes whose first-order neighbors contain very few (e.g., 1) unseen (seen) classes (i.e., sparse connection). To evaluate the performance of AZSL on different connection density, we extract a subset ImageNet* from ImageNet with all sparsely connected classes removed, where each seen (unseen) class connect with more than two unseen (seen) classes. More statistics of the datasets are listed in Table 4.

Table 4
Statistics of the image sets. "Train/Val/Test" denotes the images for training/validation/testing.

Dataset	Classes #	Seen Classes #	Unseen Classes #	Images # (Train/Val/Test)
AwA	50	40	10	23,513 / 5,896 / 7,913
ImageNet	895	398	497	318,400 / 79,600 / 409,307
ImageNet*	473	174	299	139,200 / 34,800 / 244,925

5.1.2. Baselines

The following ZSL methods are used as baselines: **DAP** [6] and **IAP** [6] which utilize the image attributes to model the inter-class relationship, **DeViSE** [3] and **ConSE** [4] which linearly map the image features learned by CNNs into the word embeddings of class labels, **SYNC** [7] which develops a series of "phantom" classes as bases to connect seen and unseen classes in the word embedding space, **GCNZ** [5] and **DGP** [18] which utilize GCN and WordNet to learn classifiers for unseen classes.

In our AZSL, we propose to leverage Attentive GCN to optimize the encoding of class knowledge so as to learn more effective unseen classifiers. For demonstrating the ability of attention layer in AGCN, we implement the graph convolutional layers with the graph convolutional operations proposed in GCNZ and DGP. The model referring to GCNZ is denoted as **AZSL-G**, while referring to DGP is denoted as **AZSL-D**.

5.1.3. Model Configuration

We adopt ResNet50 – a successful CNN architecture for extracting the features of images [66]. When training it with the samples of a specific class, the output of its second to last layer, whose dimension is 2048, is adopted as the classifier for this class. Following GCNZ [5] and DGP [18], AZSL adopts 6 convolutional layers and 2 convolutional layers for AZSL-G and AZSL-D respectively, as well as one attention layer with attention weight threshold $\alpha = 0.01$. The initial embeddings of class nodes (i.e., word embed-

Table 5
Performance (%) of AZSL-G, AZSL-D and the baselines on AwA, ImageNet and ImageNet*. † indicates the results come from the original paper. "–" means the method cannot be applied to the dataset. "±" represents the variation range of the results in the repeated experiments.

(a). AwA and ImageNet

Model	AwA	ImageNet		
	Hit@1	Hit@1	Hit@2	Hit@5
DAP	41.4†	–	–	–
IAP	42.2†	–	–	–
DeViSE	54.2†	5.40	8.53	14.02
ConSE	45.6†	9.04	13.96	20.53
SYNC	54.0†	13.08	20.35	30.80
GCNZ	68.72 ± 0.08	29.31 ± 0.12	47.11 ± 0.13	71.63 ± 0.07
AZSL-G	**69.39 ± 0.10**	**30.57 ± 0.09**	**48.23 ± 0.10**	71.32 ± 0.08
DGP	83.98 ± 0.09	34.47 ± 0.04	51.59 ± 0.07	74.79 ± 0.09
AZSL-D	**84.80 ± 0.13**	**34.81 ± 0.05**	**51.72 ± 0.07**	74.54 ± 0.15

(b). ImageNet*

Model	ImageNet*		
	Hit@1	Hit@2	Hit@5
GCNZ	23.02	43.22	73.95
AZSL-G (us)	**25.67**	**46.84**	**74.99**
DGP	32.67	53.60	79.37
AZSL-D (us)	**33.44**	**54.63**	**79.89**

dings of class names) are trained on Wikipedia 2014 dump and Gigaword 5 corpus using Glove [67] model, whose dimension is set to 300. The activation function used in graph convolutional layer is LeaklyReLU with negative input slope 0.2. Besides, we split the training samples of each seen class into 2 parts – 80% of them are used for training, while 20% are used as validation set to tune the hyperparameters of AZSL. During validation, the testing samples are predicted on seen classes to evaluate the prediction ability of learned seen classifiers so as to obtain well-trained unseen classifiers. Besides, a grid search is conducted over the parameter pools to explore the best parameters, such as {0.0001, 0.0002, 0.005} for the learning rate, {5e-3, 5e-4, 5e-5} for the L_2 regularization. Finally, the initial learning rate is set to 0.0002, the dropout parameter is set to 0.5, L_2 regularization parameter is set to 0.005, and Adam optimizer is adopted. In *EvidenceMining* algorithm, the minimum *support* and minimum *confidence* value are set to 10% and 30% respectively.

We evaluate the model with **Hits@k** metric, which represents the ratio of samples whose top k scored labels hit the ground-truth label and is widely used for performance measurement in ZSL. We set k to 1, 2 and 5, where $k = 1$ is widely believed to be the most important [17]. As AwA has only 10 unseen classes, we use **Hits@1** (i.e., accuracy) alone.

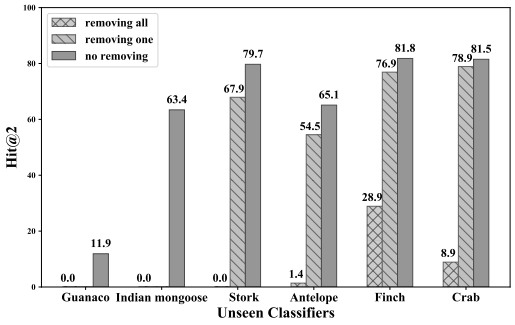


Fig. 6. Hits@2 of AZSL-G when one IMSC is removed, all IMSCs are removed and NO IMSCs are removed.

5.2. *Evaluation of Attentive ZSL Learner*

5.2.1. *Performance Comparison*

As shown in Table 5, the performance of KG-based ZSL methods, including GCNZ [5], DGP [18] and our AZSL, is much higher than that of traditional methods, especially on ImageNet. This verifies that class semantics extracted from a KG is more effective in modeling the inter-class relationship and can dramatically improve ZSL performance. This is as expected, because the semantics of class names and attributes utilized by traditional methods is not as rich as that of KG.

Compared with GCNZ and DGP – the state-of-the-art ZSL methods utilizing KG semantics, our AZSL-G and AZSL-D perform better in most of settings. This indicates the effectiveness of our Attentive GCN architecture in dealing with the ZSL problem. Considering the main goal of AZSL is to provide human understandable ZSL explanations and it is widely believed that there is a compromise between a machine learning model's interpretation and accuracy [68], the performance improvement of AZSL over GCNZ and DGP is still very promising.

5.2.2. *Performance on Dense Graph*

We also compare the performance of KG-based ZSL methods on ImageNet*, a dense graph we extract from ImageNet. We find AZSL-G and AZSL-D both have more significant outperformance over GCNZ and DGP respectively. For example, on ImageNet the Hit@1 outperformance rate of AZSL-G is 4.3%, while on ImageNet* it increases to 11.5%. This indicates the superiority of AZSL in dealing with densely connected KG. And it also validates the assumption: the performance of ZSL model can be improved by taking the different contributions of different seen classes into consideration.

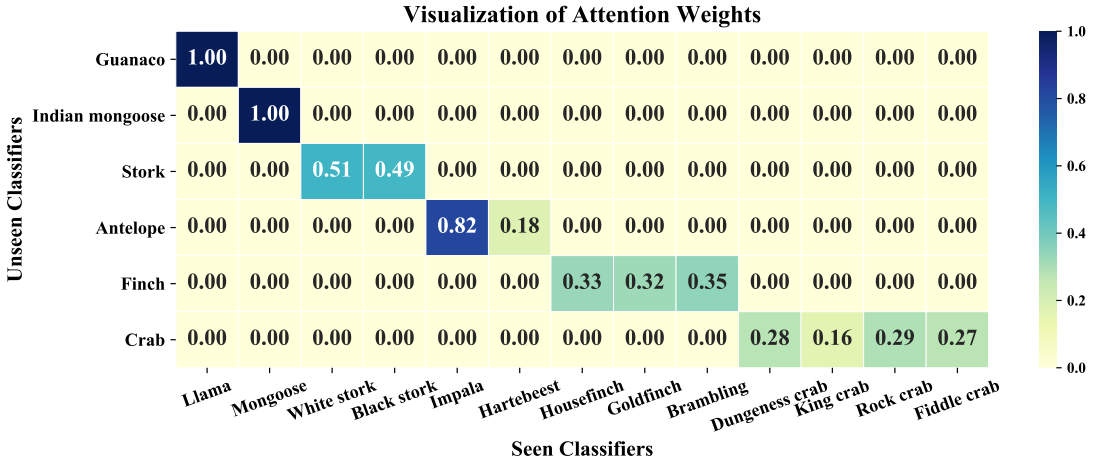


Fig. 7. Impressive seen classes (IMSCs) as well as their normalized attention weights of 6 randomly selected unseen classes. 0.00 here means a weight value below a threshold (very close to zero).

5.3. *Quantitative Analysis of Transferability*

In this subsection, we illustrate the feature transferability from seen classes to unseen classes with impressive seen classes (IMSCs) described in Section 4.2.3. Quantitatively, our AZSL learns fairly good IMSCs.

Figure 7 visualizes some unseen classes and their corresponding IMSCs, which displays the relationship between seen and unseen classes in transferring deep features in ZSL. The presented examples of IMSCs in Figure 7 are mostly consistent with our common sense on visual features of the animals, for example, in our impression, *Guanaco* and *Llama* are two animals that are similar in appearance. We also evaluate the impact of IMSCs by analyzing the performance drop when some IMSCs are removed, as shown in Figure 6. Taking the results of AZSL-G as examples, the performance decreases in all cases when some IMSCs are removed, in comparison with NO removing. Specially, it drops to 0 in most cases when all IMSCs are removed. From these observations, we can conclude that our AZSL is capable of learning reasonable IMSCs for unseen classes and these IMSCs play a key role in transferring features from seen classes to unseen classes. They can be used to generate textual explanations for the feature transferability in ZSL.

It is noted that the number of IMSCs of different unseen classes varies dramatically. For example, *Indian mongoose* and *Guanaco* have only one IMSC, while *Finch* and *Crab* have 3 and 4 respectively. We further count the distribution of IMSC size in Table 6. An interesting finding is that most unseen classes have only one IMSC (78.90%) or two IMSCs (5.72%). While around 2% of all unseen classes have more than two IMSCs. For example, class *Rat* has three impressive seen classes *Mouse*, *Hamster* and *Beaver*, each of them whose attributes extracted from Attribute Graph are more than 30. The searching space is large for finding common attribute set among these classes. From our statistics, although the proportion of unseen classes with multiple IMSCs is not high, the *EvidenceMining* algorithm provides a demonstration for our system to be applied to other datasets and tasks. Moreover, we not only extract common attributes using the algorithm but also mine the potential rules of seen and unseen classes with some measurements produced – *support* and *confidence* values, which illustrate the ratio of common attributes to all attributes of these classes.

Table 6
The distribution of impressive seen classes (IMSCs).

Size of IMSC	0	1	2	3	4	5	6	7
Number of Unseen Class	67	400	29	5	3	1	1	1
Ratio (%)	13.20	78.90	5.72	0.99	0.59	0.20	0.20	0.20

We also note that some unseen classes (around 13.20%) do not have any impressive seen classes. It is probably because: *(i)* there are no neighboring seen classes in the WordNet graph, *(ii)* the visual features of neighboring seen classes is quite different from these unseen classes (meaning the gap between semantic domain and vision domain), or *(iii)* the shortcomings of

AZSL model which can not be capable of transferring deep features from seen classes to these unseen classes.

5.4. Evaluation of Explanations

In this subsection, we demonstrate how human beings are satisfied with the generated textual explanations. We also compare the impact of different external KGs, and present some case studies.

5.4.1. Human Evaluation

For human evaluation, we invite 25 volunteers without AI expertise to score the generated explanations. The first language of these volunteers is Chinese, they are all undergraduate students who are fluent in reading English. We divide unseen class set into 5 parts. Each part has about 100 unseen classes and explanations. Each explanation is assessed by 5 volunteers, and the final decision is made by majority voting.

We defined two metrics – *readability* and *rationality* for evaluation. "G" (Good), "M" (Median) or "B" (Bad) are scored for each metric.

Readability: it measures whether an explanation is natural and fluent. Good: fluent, Medium: unnatural but still understandable, Bad: confused and incomprehensible.

Rationality: it measures whether an explanation illustrates the transferability from seen to unseen classes. Good: well illustrated, Medium: insufficient and weakly illustrated, Bad: totally unconvincing.

Note that the two metrics are scored independently. To help volunteers master the domain of the classification application and the evaluation procedure, we prepare some guidelines and examples for them before scoring to make sure they are familiar with the scoring process. Besides, we also provide some additional images and textual illustrations of classes as references as well as some notes of generated explanations during scoring. It is allowed that volunteers can skip the scoring item if they are not sure about their judgment.

Table 7 presents the human evaluation results. We can find that the explanations of most unseen classes obtain a highly satisfied result, especially on the rationality, and only a very small ratio of explanations get "Bad" on readability and rationality. It can also be seen that 60.77% of the explanations get Median on readability, but they do not have much impact on people's satisfaction with rationality. This indicates that the templates for explanation generation need further refinement, which is among our future work.

Table 7
Results of human evaluation on the explanations.

Readability		Rationality	
Score	**Ratio of Explanations**	**Score**	**Ratio of Explanations**
G	36.58%	G	73.20%
M	60.77%	M	20.30%
B	2.65%	B	6.50%

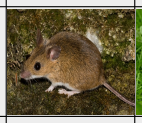
5.4.2. Impact of Different Types of KGs

We present the generated explanations of unseen classes *Horse, Rat, Dolphin* and *Stork* in Figure 8, with evidence from the domain-specific Attribute Graph and the general DBpedia. For example, *Horse* is related with its impressive seen classes *Zebra*, with the attention weight of 1.0. As the upper left case of Figure 8 shows, our explanation generator extracts attribute evidence such as *hooves, longneck, chewteech, tail* from Attribute Graph, to illustrate the common attributes between *Horse* and *Zebra* and mine the association rule: "$\{zebra \Rightarrow horse\}$", with a *support* value 73.0% and a *confidence* value 90.0%. In addition, generator also extracts general knowledge from DBpedia: some triples such as *(dbr:Horse, dct:subject, dbr:Equus)* and *(dbr:Zebra, dct:subject, dbr:Equus)* which describe their common ancestor, and some keywords such as *night vision* and *ears* which describe their common characteristics. With these evidence, we can justify that predicting an image as *Horse* is based on the feature transfer from *Zebra*, and is reasonable.

We find that the evidence from different external KGs have different characteristics. Taking *Rat* in Figure 8 as an example, the evidence from Attribute Graph are more likely to generate explanations in vision, including body part (*quadrupedal, paws*), coat appearance (*furry*), while evidence from DBpedia often generate more general explanations, including not only visual descriptions such as *incisors*, but also general descriptions such as *rodent* ancestor and *invasive mammal* in biology. Other cases have similar performances, this is because of the nature of these two KGs – Attribute Graph is a domain-specific knowledge graph while DBpedia is a general one.

We also find and compare some limitations of these two KGs. Due to the great cost on attribute annotations, the scale of Attribute Graph is limited, and a considerable portion of classes (about 90%), especially those from ImageNet, whose attributes extracted from Attribute Graph have no more than 10. For example, there is few attribute descriptions about *Stork* and its IMSCs (see the bottom right of Figure 8). In contrast,

		Horse (Unseen Class)	IMSCs of Horse
			Zebra (*w*: 1.0)
Image			
DBpedia Entity		dbr:Horse	dbr:Zebra
Knowledge from Attribute Graph		hooves, longneck, chewteeth, tail, muscle, grazer, plains, grouped, quadrupedal, timid	
		For rule $\{zebra\} \Rightarrow \{horse\}$: sup. = 73.0%; con. = 90.0%	
Knowledge from DBpedia	DBPedia Property/ Relation	(dbr:Horse, dct:subject, dbc:Equus) (dbr:Horse, dct:subject, dbc: Herbivorous_animals)	(dbr:Zebra, dct:subject, dbc:Equus) (dbr:Zebra, dct:subject, dbc:Herbivorous_animals)
	Keywords of Abstract Text	**ungulate** mammal, domesticated, day and **night vision**, balance, **large ears**	black-and-white striped coats, **ungulates**, social, **large ears**, **night vision**
Generated Explanation & Score		The prediction for samples of **horse** is supported by **zebra**. *They are both grazer, grouped, quadrupedal, both have hooves, muscle, longneck, chewteeth, both live in plains, and behave timid.* They are both species of **equus**, and are both a member of **herbivorous animals**. They are both **ungulate** animal, and are similar in **night vision** and **large ears**.	
		Readability [G/M/B]: M **Rationality [G/M/B]: G**	

		Rat (Unseen Class)	IMSCs of Rat		
			Mouse (*w*: 0.36)	Hamster(w:0.34)	Beaver (*w*: 0.30)
Image					
DBpedia Entity		dbr:Rat	bdr:Mouse	dbr:Hamster	dbr:Beaver
Knowledge from Attribute Graph		quadrupedal, ground, nocturnal, paws, small, buckteeth, hibernate, nestspot, agility, furry			
		For rule $\{mouse, hamster, beaver\} \Rightarrow \{rat\}$: sup. = 26.8%; con. = 88.2%			
Knowledge from DBpedia	DBPedia Property/ Relation	(dbr:Rat, hypernym, dbr:Rodents) (dbr:Hamster, hypernym, dbr:Rodents) (dbr:Rat, dct:subject, dbc:Invasive_mammal) (dbr:Mouse, dct:subject, dbc:Invasive_mammal)			
	Keywords of Abstract Text	medium-sized, long-tailed **rodents**, **incisors**	pointed **incisors**, rounded ears, **Rodentia**, high breeding rate	**rodents**, sharp **incisors**, house pets, underground,	semiaquatic **rodent**, nocturnal, building dams, four **incisors**
Generated Explanation & Score		The prediction for samples of **rat** is supported by **mouse**, **hamster** and **beaver**. *They are both quadrupedal, nocturnal, small, furry, both have paws, buckteeth, both live in ground, nestspot, both hibernate and behave agility.* **Rat** and **hamster** are both relevant to **rodents** via relation **hypernym**, **rat** and **mouse** both belong to **invasive mammal**. They are both have **incisors**.			
		Readability [G/M/B]: M **Rationality [G/M/B]: G**			

		Dolphin (Unseen Class)	IMSCs of Dolphin
			Killer whale (*w*: 1.0)
Image			
DBpedia Entity		dbr:Dolphin	dbr:Killer_whale
Knowledge from Attribute Graph		hairless, toughskin, flippers, swims, tail, ocean, grouped, smart, fast, active	
		For rule $\{killer\ whale\} \Rightarrow \{dolphin\}$: sup. = 48.6%; con. = 62.1%	
Knowledge from DBpedia	DBpedia Property/ Relation	(dbr:Dolphin, dct:subject, dbc:Animals_that_use_echolocation)	(dbr:Killer_whale, dct:subject, dbc:Animals_that_use_echolocation)
	Keywords of Abstract Text	aquatic, shaped **teeth**, well-developed **hearing**, widespread, **blubber** under skin	oceanic, apex predators, **toothed** whale, a layer of **blubber**, excellent **hearing**, diverse diet
Generated Explanation & Score		The prediction for samples of **dolphin** is supported by **killer whale**. *They are both hairless, grouped, both have tough skin, flippers, tail, both swim, live in ocean, and behave smart, fast and active.* They are both **animals that use echolocation**. They both have **teeth**, **hearing** and **blubber**.	
		Readability [G/M/B]: M **Rationality [G/M/B]: M**	

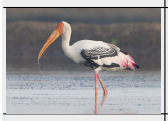
		Stork (Unseen Class)	IMSCs of Stork	
			White stork (*w*: 0.51)	Black stork (*w*: 0.49)
Image				
DBpedia Entity		dbr:Ciconiiformes	dbr:White_stork	dbr:Black_stork
Knowledge from Attribute Graph		white, black, water, wild, fish		
		For rule $\{white\ stork, black\ stork\} \Rightarrow \{stork\}$: sup. = 83.3%; con. = 100.0%		
Knowledge from DBpedia	DBpedia Property/ Relation	(dbr:White_stork, dbo:order, dbr:Ciconiiformes) (dbr:Black_stork, dbo:order, dbr:Ciconiiformes)		
	Keywords of Abstract Text	**large**, **long-legged**, long-necked, **wading**, **birds**, soaring, long, stout bills, **migratory**	**large bird**, **long** red **legs**, **wading**, family, Ciconiidae, **migrant**, carnivore	**large bird**, **wading**, family, Ciconiidae, black plumage, **long** red **legs**, red beak
Generated Explanation & Score		The prediction for samples of **stork** is supported by **white stork** and **black stork**. *They are both white, black, wild, both live in water and eat fish.* **White stork** and **black stork** both belong to **stork** biologically. They are both **large wadding birds**, and are similar in **long legs**.		
		Readability [G/M/B]: G **Rationality [G/M/B]: G**		

Fig. 8. The explanations of unseen classes: *Horse, Rat, Dolphin* and *Stork*. In each case, images of the unseen class and its impressive seen classes (IMSCs), their matched DBpedia entities, the extracted attributes, triples and keywords are displayed. The association rules of seen and unseen classes from *EvidenceMining* algorithm are listed with their measurements ("sup.": the *support* value, "con.": the *confidence* value). "(w:∗)" behind IMSCs denotes the attention weights. The textual explanations as well as the evaluations from volunteers are also displayed. The sentences in italic are explanations generated with knowledge from Attribute Graph (i.e., attributes), while those marked with underline are generated with knowledge from DBpedia (i.e., triples and keywords).

Table 8
The evaluation results of explanations with different numbers of common attributes.

Attributes #	Rationality			Readability		
	G	M	B	G	M	B
>10	85.71%	14.29%	0.00%	0.00%	100.00%	0.00%
5 ∼ 10	75.00%	21.88%	3.12%	40.63%	56.25%	3.12%
<5	72.68%	20.36%	6.96%	37.37%	59.79%	2.84%

DBpedia owns abundant resources, the knowledge can be accessed as long as the ZSL classes are matched with DBpedia entities. This setting is very friendly for providing explanations for the classes of large scale ZSL datasets. However, different from the human-annotated attributes in Attribute Graph, the knowledge from DBpedia sometimes is noisy such as keywords extracted from the abstract text. It can be found in the case of *Dolphin* (the bottom left of Figure 8). Taking *hearing* as an example, the keywords "well-developed hearing" and "excellent hearing" are extracted for *Dolphin* and *Killer whale* respectively, which describe the same characters about their hearing, however, only "hearing" is taken as common keywords because of the different adjective prefix, causing the incompleteness of knowledge. This also motivates us to improve the algorithm for keyword extraction to make full use of the knowledge in the abstract text.

In summary, the evidence from Attribute Graph are more applicable to generate explanations for specific domain such as vision, especially when the attribute annotations are sufficient, while the evidence from DBpedia are more general and accessable: it can deal with large scale ZSL problems with a number of classes and can also be applied to different ZSL applications like text classification. Meanwhile, the explanations from Attribute Graph and DBpedia are compatible with each other, and can be combined.

5.4.3. Impact of Attribute Graph

From Figure 8, we also find that the number of attributes extracted from Attribute Graph varies a lot with regard to different classes. For example, *Horse*, *Dolphin* and *Rat* both have 10 common attributes while *Stork* only has 5. In our statistics, 3.23% of all unseen classes (or explanations) own more than 10 common attributes extracted from Attribute Graph, 7.37% have 5 ∼ 10 common attributes while 89.4% have less than 5 attributes. Therefore, we further analyze the impact of the coverage of attributes on generating explanations. Specifically, with different numbers of extracted common attributes, we reevaluate the quality of generated explanations, the results are shown in Table 8.

In all explanations with more than 10 common attributes, 85.71% of them are evaluated with "Good" on rationality. 14.29% are scored with "Medium" on rationality, it may be because *(i)* we randomly select 10 attributes for generating explanations when the number of attributes more than 10, in which the representative characteristics may miss, or *(ii)* the knowledge extracted from DBpedia may be not convincing enough. It can also be seen that the proportion of explanations scored with "B" increases as the number of attributes decreases, illustrating that more attributes can make up for the shortcomings of evidence from DBpedia. Besides, most of explanations can get "G" or "M" on rationality when the number of attributes is less than 10 or even less than 5, meaning that our system can still work well with DBpedia even though the attributes extracted from Attribute Graph are not rich enough.

As for readability, we find that there is a slip in the quality of the explanations when packing too many attributes, it might be because there are some repetitive or redundant expressions in the generated sentences. It is believed that randomly taking 10 attributes to generate explanations is a good choice when the number of attributes exceeds 10, however, some significant attributes may be missing as we mentioned above. Therefore, we look forward to adopting some strategies to improve the selection of attributes in the future, for example, evaluating the relevancy between the classes and the attributes to select the most relevant ones.

Generally speaking, the high coverage of attributes has a positive effect on generating high quality explanations, especially in terms of the rationality. It is better to combine the knowledge from Attribute Graph and DBpedia to complement each other.

5.5. Discussion on Feature Transferability and ZSL Prediction

In this subsection, we analyze the relationship between feature transferability and ZSL prediction according to our generated explanations. We take the prediction results of AZSL-G as examples.

5.5.1. Successful and Failed Transfer

In ZSL, the testing samples of unseen classes are predicted by transferring features from seen classes. However, we find a case where some unseen classes have no features transferred from seen classes, and the

Table 9
Performance (Hit@1 and Hit@2) of AZSL-G according to different types of transferability.

Transferability	Ratio of Unseen Classes	Performance (%)	
		Hit@1	Hit@2
ancestor	49.2 %	25.06	46.79
sibling	38.1 %	29.10	50.58
ancestor-sibling	1.2 %	66.05	79.52
other	11.5 %	37.40	49.56

prediction results (Hits@1 and Hits@2) are both 0. We call such a case as a *failed transfer* (FT). In contrast, the case where some unseen classes have effective features transferred from seen classes is named as a *successful transfer* (SF). For example, the Hit@1 and Hit@2 accuracy of unseen class *Eared seal* are both 0 and its feature transferring is failed, while another unseen class *Frog* transferring features from seen class *Tree frog* and *Tailed frog* own performance on Hit@1 with 61.70 and Hit@2 with 79.62, is a successful case.

Those explanations that are scored "Good" on rationality well illustrate the feature transferability from seen classes to unseen classes, and are regarded as *good explanations*, while others are not. We find that 48% of the unseen classes are SF with good explanations, while 39% are SF without good explanations. The shortage of good explanations may be due to *(i)* the noise of knowledge extracted from the KG, *(ii)* the incorrect or absent matching between classes and DBpedia entities as mentioned in Section 4.3.2, or *(iii)* the absence of impressive seen classes of unseen classes, while the successful transfer relies on the semantics contained in the word embeddings of class names (i.e., the initialization of class nodes). The first point can be solved by developing more advanced methods to extract knowledge, while the second point can be improved by traditional ontology alignment systems and modern semantic embedding methods. We also find around 3% of the unseen classes are FT with good explanations. This may be because the learned unseen classifiers are not accurate enough (learning to very low Hit@1 and Hit@2) although features are transferred, or the feature extraction in CNN needs to be refined. This indicates that on the one hand, the class knowledge can be further enhanced, on the other hand, the encoded class knowledge can be utilized to improved the CNN module. The rest (10%) of unseen classes are FT without good explanations. Maybe it is because some related seen classes are missing in the ZSL datasets, resulting in no features can be transferred to these unseen classes.

5.5.2. *Different Types of Transferability*

We find that the feature transfer from seen classes to unseen classes have different types. For example, some features are transferred between two sibling classes, and some others are transferred from one class to its children or parents. Given a successful transfer between a seen class and an unseen class, we divide it into several types: *(i) ancestor* which refers to the case where the seen class is the ancestor of the unseen classes or vice versa (e.g., the unseen class *Stork* is the ancestor of one of the seen classes *White stork* in Figure 8); *(ii) sibling* which refers to the case where the seen class and the unseen class are siblings (e.g., the unseen class *Horse* and the seen class *Zebra* in Figure 8 are both children of *Equus*); *(iii) ancestor-sibling*, which refers to the case where the transfer type between seen and unseen classes includes both *ancestor* and *sibling*; *(iv) other* which refers to the case where there are no ancestor or sibling relationship between seen and unseen classes.

We count all successful transfer predictions of unseen classes in ImageNet according to the different types of transferability. As Table 9 shows, 49.2% of unseen classes learn their classifiers by transferring features from *ancestor* type seen classes and own performance on Hit@1 with 25.06, while 38.1% transfer features from *sibling* type seen classes and own performance on Hit@1 with 29.10. Nearly 90% of unseen classes focus on these two kinds of feature transfer, it is because the inter-class relationship our model input is hierarchical. We also find that the prediction results of unseen classes with *sibling* type are superior to those with *ancestor* type, probably because of the divergence of feature distribution between ancestor classes and descendant classes considering the feature distribution of ancestor classes is more complex than that of descendant classes. This motivates us that the performance of ZSL model may be improved by introducing *sibling* type feature transfer, for example, introducing related sibling seen classes for unseen classes. It is also noted that the combination of these two types achieves the highest prediction results, indicating that richer class semantics are more helpful for the transfer of features.

6. Conclusion and Outlook

In this study, we investigate explainable ZSL with (1) a new ZSL learner that utilizes inter-class relationships extracted from WordNet as well as an Attentive

Graph Convolutional Network, and (2) an explanation generator that can make human understandable explanations with external Attribute Graph and DBpedia to justify the transferability of deep features in ZSL. The study is evaluated with two image sets. We not only achieve higher overall ZSL accuracy than the state-of-the-art baselines, but also generate high quality explanations that can make the feature transfer procedure and ZSL prediction more explainable. With the generated explanations, we also investigate the relationship between feature transferability and ZSL performance, which has the potential of further improving the performance of ZSL algorithms.

In our work, we extract class knowledge e.g., attributes, keywords from Attribute Graph and DBpedia as evidence to illustrate the transferability of deep features from seen classes to unseen classes. From another point of view, these evidence also provide detailed semantic descriptions about seen and unseen classes, which is helpful for feature transfer in ZSL. Therefore, in the future work, we may integrate these common attributes, keywords or triples into the hierarchical inter-class relationship to enrich the class semantics and improve the performance of ZSL models.

We also consider further improving the explanation quality by utilizing richer semantics from KGs. For example, to make full use of the knowledge from Attribute Graph and DBpedia, we can utilize Semantic Web techniques such as ontology that involves the domain and range of properties to assist the knowledge extraction from Attribute Graph and DBpedia. Note that the ontology of DBpedia is ready-made, while the ontology of Attribute Graph needs to be designed considering the source of Attribute Graph – a domain-specific KG collected from attribute annotations. We will explore this direction in the future.

Our work currently focuses on the image classification tasks, we also look forward to applying the method in other domains like KG construction and natural language processing, for example, long-tail relation extraction task [44] and zero-shot knowledge graph completion task [69]. In these methods that utilize label knowledge to transfer data features from seen (data-rich) labels to unseen (data-poor) labels, we can also introduce attention mechanism or other strategies into the classification model to attentively select the contributing seen (or data-rich) labels in the feature transfer process. With the learned relationship between seen and unseen labels, we can access knowledge from external KGs to generate human understandable explanations by aligning the labels with external KG entities as the methods we propose. It is noted that the external KGs such as DBpedia can still be used to generate explanations in NLP or KG construction domain, while Attribute Graph is specific for the vision domain. If necessary, we can explore other external KGs for new applications or introduce domain knowledge from experts or public resources.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., Imagenet large scale visual recognition challenge, *International journal of computer vision* **115**(3) (2015), 211–252.

[2] M. Palatucci, D. Pomerleau, G.E. Hinton and T.M. Mitchell, Zero-shot learning with semantic output codes, in: *Advances in neural information processing systems*, 2009, pp. 1410–1418.

[3] A. Frome, G.S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov et al., Devise: A deep visual-semantic embedding model, in: *Advances in neural information processing systems*, 2013, pp. 2121–2129.

[4] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G.S. Corrado and J. Dean, Zero-shot learning by convex combination of semantic embeddings, *arXiv preprint arXiv:1312.5650* (2013).

[5] X. Wang, Y. Ye and A. Gupta, Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.

[6] C.H. Lampert, H. Nickisch and S. Harmeling, Attribute-based classification for zero-shot visual object categorization, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(3) (2014), 453–465.

[7] S. Changpinyo, W.-L. Chao, B. Gong and F. Sha, Synthesized classifiers for zero-shot learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5327–5336.

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

[9] R.R. Selvaraju, P. Chattopadhyay, M. Elhoseiny, T. Sharma, D. Batra, D. Parikh and S. Lee, Choose Your Neuron: Incorporating Domain Knowledge through Neuron-Importance, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 526–541.

[10] H. Larochelle, D. Erhan and Y. Bengio, Zero-data learning of new tasks., in: *AAAI*, Vol. 1, 2008, p. 3.

[11] C.H. Lampert, H. Nickisch and S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 951–958.

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

[13] Z. Akata, F. Perronnin, Z. Harchaoui and C. Schmid, Label-embedding for attribute-based classification, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 819–826.

[14] P. Morgado and N. Vasconcelos, Semantically consistent regularization for zero-shot recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6060–6069.

[15] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein and B. Schiele, Latent embeddings for zero-shot classification, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 69–77.

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

[17] Y. Xian, C.H. Lampert, B. Schiele and Z. Akata, Zero-shot learning-A comprehensive evaluation of the good, the bad and the ugly, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[18] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang and E.P. Xing, Rethinking knowledge graph propagation for zero-shot learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11487–11496.

[19] P.K. Pushp and M.M. Srivastava, Train Once, Test Anywhere: Zero-Shot Learning for Text Classification, *arXiv preprint arXiv:1712.05972* (2017).

[20] J. Zhang, P. Lertvittayakumjorn and Y. Guo, Integrating Semantic Knowledge to Tackle Zero-shot Text Classification, *arXiv preprint arXiv:1903.12626* (2019).

[21] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin and H. Lee, Zero-Shot Entity Linking by Reading Entity Descriptions, *arXiv preprint arXiv:1906.07348* (2019).

[22] S. Rijhwani, J. Xie, G. Neubig and J. Carbonell, Zero-shot neural transfer for cross-lingual entity linking, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 6924–6931.

[23] O. Levy, M. Seo, E. Choi and L. Zettlemoyer, Zero-Shot Relation Extraction via Reading Comprehension, in: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 333–342.

[24] P. Pasupat and P. Liang, Zero-shot entity extraction from web pages, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, 2014, pp. 391–401.

[25] O. Biran and C. Cotton, Explanation and justification in machine learning: A survey, in: *IJCAI-17 Workshop on Explainable AI (XAI)*, 2017, p. 8.

[26] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A survey of methods for explaining black box models, *ACM Computing Surveys (CSUR)* **51**(5) (2018), 93.

[27] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A survey of methods for explaining black box models, *ACM computing surveys (CSUR)* **51**(5) (2019), 93.

[28] Z.C. Lipton, The mythos of model interpretability, *arXiv preprint arXiv:1606.03490* (2016).

[29] C. Rudin, B. Letham and D. Madigan, Learning theory analysis for association rules and sequential event prediction, *The Journal of Machine Learning Research* **14**(1) (2013), 3441–3492.

[30] M.T. Ribeiro, S. Singh and C. Guestrin, Why should i trust you?: Explaining the predictions of any classifier, in: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, 2016, pp. 1135–1144.

[31] W. Landecker, M.D. Thomure, L.M. Bettencourt, M. Mitchell, G.T. Kenyon and S.P. Brumby, Interpreting individual classifications of hierarchical networks, in: *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, IEEE, 2013, pp. 32–38.

[32] T. Kulesza, M. Burnett, W.-K. Wong and S. Stumpf, Principles of explanatory debugging to personalize interactive machine learning, in: *Proceedings of the 20th international conference on intelligent user interfaces*, 2015, pp. 126–137.

[33] D.S. Weld and G. Bansal, The challenge of crafting intelligible intelligence, *Communications of the ACM* **62**(6) (2019), 70–79.

[34] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm and N. Elhadad, Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission, in: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 1721–1730.

[35] O. Biran and K.R. McKeown, Human-Centric Justification of Machine Learning Predictions., in: *IJCAI*, 2017, pp. 1461–1467.

[36] Q. Li, J. Fu, D. Yu, T. Mei and J. Luo, Tell-and-answer: Towards explainable visual question answering using attributes and captions, *arXiv preprint arXiv:1801.09041* (2018).

[37] D. Ratcliffe and K. Taylor, Closed-world concept induction for learning in OWL knowledge bases, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2014, pp. 429–440.

[38] I. Tiddi, M. dAquin and E. Motta, Dedalo: Looking for clusters explanations in a labyrinth of linked data, in: *European Semantic Web Conference*, Springer, 2014, pp. 333–348.

[39] J. Chen, F. Lécué, J.Z. Pan, I. Horrocks and H. Chen, Knowledge-based Transfer Learning Explanation, in: *KR*, 2018.

[40] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, Hierarchical attention networks for document classification, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[41] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang and G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 2627–2633.

[42] S.J. Pan and Q. Yang, A survey on transfer learning, *IEEE Transactions on knowledge and data engineering* **22**(10) (2009), 1345–1359.

[43] K. Weiss, T.M. Khoshgoftaar and D. Wang, A survey of transfer learning, *Journal of Big data* **3**(1) (2016), 9.

[44] N. Zhang, S. Deng, Z. Sun, G. Wang, X. Chen, W. Zhang and H. Chen, Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks, *arXiv preprint arXiv:1903.01306* (2019).

[45] C.-W. Lee, W. Fang, C.-K. Yeh and Y.-C. Frank Wang, Multi-label zero-shot learning with structured knowledge graphs, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.

[46] F. Lecue, J. Chen, J.Z. Pan and H. Chen, Augmenting Transfer Learning with Semantic Reasoning, in: *IJCAI*, 2019.

[47] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, How transferable are features in deep neural networks?, in: *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[48] T. Liu, Q. Yang and D. Tao, Understanding How Feature Structure Transfers in Transfer Learning., in: *IJCAI*, 2017, pp. 2365–2371.

[49] Y. Geng, J. Chen, E. Jimenez-Ruiz and H. Chen, Human-centric Transfer Learning Explanation via Knowledge Graph, *arXiv preprint arXiv:1901.08547* (2019).

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

[51] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie and P. Perona, Caltech-UCSD birds 200 (2010).

[52] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195.

[53] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* (2016).

[54] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed representations of words and phrases and their compositionality, in: *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[55] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, Graph attention networks, *arXiv preprint arXiv:1710.10903* (2017).

[56] R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, in: *Acm sigmod record*, Vol. 22, ACM, 1993, pp. 207–216.

[57] R. Agrawal, R. Srikant et al., Fast algorithms for mining association rules, in: *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215, 1994, pp. 487–499.

[58] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: *Proceedings of the 7th international conference on semantic systems*, ACM, 2011, pp. 1–8.

[59] A.-C.N. Ngomo and S. Auer, LIMESa time-efficient approach for large-scale link discovery on the web of data, in: *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[60] M. Pershina, M. Yakout and K. Chakrabarti, Holistic entity matching across knowledge graphs, in: *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 1585–1590.

[61] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo and Y. Qu, Multi-view knowledge graph embedding for entity alignment, *arXiv preprint arXiv:1906.02390* (2019).

[62] Y. Wu, X. Liu, Y. Feng, Z. Wang and D. Zhao, Jointly Learning Entity and Relation Representations for Entity Alignment, *arXiv preprint arXiv:1909.09317* (2019).

[63] R. Mihalcea and P. Tarau, Textrank: Bringing order into text, in: *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[64] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K.J. Miller, Introduction to WordNet: An on-line lexical database, *International journal of lexicography* **3**(4) (1990), 235–244.

[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database (2009).

[66] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[67] J. Pennington, R. Socher and C. Manning, Glove: Global vectors for word representation, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[68] H. Paulheim, Make embeddings semantic again!, in: *International Semantic Web Conference (P&D/Industry/BlueSky)*, 2018.

[69] P. Qin, X. Wang, W. Chen, C. Zhang, W. Xu and W.Y. Wang, Generative Adversarial Zero-Shot Relational Learning for Knowledge Graphs, *arXiv preprint arXiv:2001.02332* (2020).

[70] G. Salton, A. Wong and C.-S. Yang, A vector space model for automatic indexing, *Communications of the ACM* **18**(11) (1975), 613–620.

[71] G. Patterson and J. Hays, Sun attribute database: Discovering, annotating, and recognizing scene attributes, in: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 2751–2758.

[72] C.H. Lampert, H. Nickisch and S. Harmeling, Attribute-based classification for zero-shot visual object categorization, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(3) (2013), 453–465.

[73] O. Russakovsky and L. Fei-Fei, Attribute learning in large-scale datasets, in: *European Conference on Computer Vision*, Springer, 2010, pp. 1–14.

[74] H. Liu, Q. Yin and W.Y. Wang, Towards Explainable NLP: A Generative Explanation Framework for Text Classification, *arXiv preprint arXiv:1811.00196* (2018).

[75] G. Piao and J.G. Breslin, Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model, in: *European Semantic Web Conference*, Springer, 2018, pp. 496–511.