Rethinking Knowledge Graph Propagation for Zero-Shot Learning

Michael Kampffmeyer*[1], Yinbo Chen*[2], Xiaodan Liang†[3], Hao Wang[4], Yujia Zhang[5], and Eric P. Xing[3]

[1] UiT The Arctic University of Norway, [2] Tsinghua University, [3] Carnegie Mellon University, [4] Massachusetts Institute of Technology, [5] Chinese Academy of Sciences
michael.c.kampffmeyer@uit.no, cyb15@mails.tsinghua.edu.cn, xiaodan1@andrew.cmu.edu, hwang87@mit.edu, zhangyujia2014@ia.ac.cn, epxing@cs.cmu.edu

Abstract

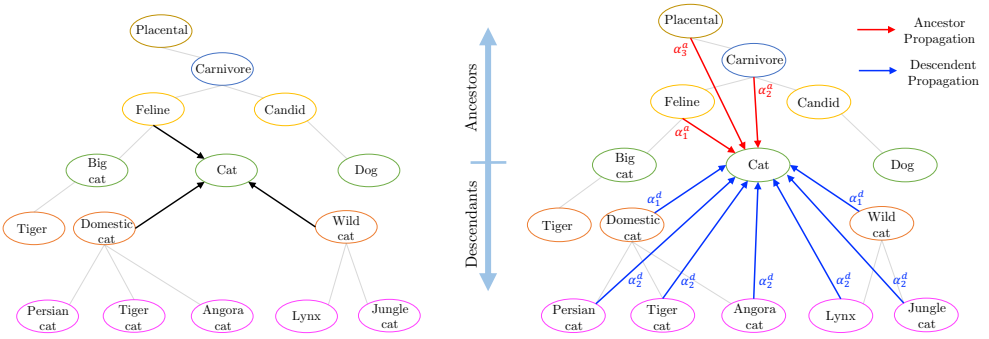
The potential of graph convolutional neural networks for the task of zero-shot learning has been demonstrated recently. These models are highly sample efficient as related concepts in the graph structure share statistical strength allowing generalization to new classes when faced with a lack of data. However, knowledge from distant nodes can get diluted when propagating through intermediate nodes, because current approaches to zero-shot learning use graph propagation schemes that perform Laplacian smoothing at each layer. We show that extensive smoothing does not help the task of regressing classifier weights in zero-shot learning. In order to still incorporate information from distant nodes and utilize the graph structure, we propose an Attentive Dense Graph Propagation Module (ADGPM). ADGPM allows us to exploit the hierarchical graph structure of the knowledge graph through additional connections. These connections are added based on a node's relationship to its ancestors and descendants and an attention scheme is further used to weigh their contribution depending on the distance to the node. Finally, we illustrate that finetuning of the feature representation after training the ADGPM leads to considerable improvements. Our method achieves competitive results, outperforming previous zero-shot learning approaches.

1 Introduction

With the ever-growing supply of image data, from an ever-expanding number of classes, there is an increasing need to use prior knowledge to classify images from unseen classes into correct categories based on semantic relationships between seen and unseen classes. This task is called zero-shot image classification. To obtain satisfactory performance on this task, it is crucial to model precise class relationships based on prior class knowledge. Previously prior knowledge has been incorporated in form of semantic descriptions of classes, such as attributes [1, 27, 19] or word embeddings [29, 11, 18], or by using semantic relations such as knowledge graphs [23, 26, 28]. Approaches that use knowledge graphs are less-explored and generally are based on the assumption that unknown classes can exploit similarity to known classes. Recently the benefit of hybrid approaches that combine knowledge graph and semantic class descriptions has been illustrated [31].

The current state-of-the-art approach [31] processes knowledge graphs by making use of recent developments in applying neural network techniques to non-euclidean spaces, such as graph and manifold spaces [2]. They employ a 6-layer graph convolutional neural network (GCN) [14] and phrase the problem as weight regression, where the GCN is trained to regress classifier weights for each class. GCNs balance model complexity and expressiveness with a simple scalable model

* indicates equal contribution. † indicates the corresponding author.



(a) Graph Propagation (b) Dense Graph Propagation

Figure 1: a) Illustration of graph propagation in GPM for node 'Cat'. Similarly, the propagation represents the knowledge that a node receives in a single layer for previous approaches. b) Proposed dense graph propagation for node 'Cat'. The node receives knowledge from all its descendents during the descendent phase (blue arrows) and its ancestors during the ancestor phase (red arrows). This leads to a densely connected graph where knowledge can directly propagate between related nodes. Attention weights α_k are used to weigh nodes that are k-hops away from a given node.

relying on the idea of message passing, i.e. nodes pass knowledge to their neighbors. However, these models were originally designed for classification tasks, albeit semi-supervised, an arguably simpler task than regression. In recent work, it has been shown that GCNs perform a form of Laplacian smoothing, where feature representations will become more similar as depth increases leading to easier classification [17]. In the regression setting, instead, the aim is to exchange information between nodes in the graph and extensive smoothing is not desired as it dilutes information and does not allow for accurate regression. For instance, in a connected graph all features in a GCN with n layers will converge to the same representation as $n \to \infty$ under some conditions, hence washing out all information [17].

We, therefore, argue that this approach is not ideal for the task of zero-shot learning and that the number of layers in the graph should be small in order to avoid smoothing. We illustrate this phenomenon in practice, by proposing a Graph Propagation Module (GPM) that only consists of a single layer and that consistently outperforms previously reported results. As in [31] we use the GPM in a model-of-models framework by training it to predict a set of logistic regression classifier for each class on top of a set of extracted features produced by a CNN.

Choosing a small number of layers, however, has the effect that knowledge will not propagate well through the graph as a 1-layer GCNs only considers neighbors that are two hops away in the graph. This means that only immediate neighbors influence a given node. Thus, we propose a dense connectivity scheme, where nodes are connected directly to descendants/ancestors in order to include distant information. These connections allow us to propagate information without many smoothing operations. However, this leads to the problem that all descendants/ancestors are weighed equally when computing the regression weight vector for a given class. However, intuitively, nodes closer to a given node should have higher importance. To remedy this, we extend this framework by adding an attention scheme that considers the distance between nodes in order to weigh the contribution of different nodes. Figure 1 illustrates the difference in the way knowledge is propagated in this Attentive Dense Graph Propagation Module (ADGPM) compared to the GPM.

We further consider the problem of domain-shift in zero-shot learning, which is the problem that current methods often struggle to perform well on both seen and unseen classes [6]. Seen and unseen classes can be considered overlapping domains with a set of shared appearances, however, they also exhibit domain differences [15]. To remedy this and allow the feature extraction stage of the pre-trained CNN to adjust to the newly learned classifiers we propose a two-phase training scheme. In the first step, the ADGPM is trained to predict the last layer CNN weights. In the second phase, we finetune the CNN to predict the ADGPM output in order to allow the CNN feature representation to adjust to the predicted weights, incorporating the implicit constraint of the knowledge graph.

We present an analyzes of our intuitions for zero-shot learning and illustrate how these intuitions can be combined to design a ADGPM that outperforms previous zero-shot learning results. To summarize,

the key benefit of the proposed ADGPM module is that it explicitly exploits the hierarchical structure of the knowledge graph in order to perform zero-shot learning by more efficiently propagating knowledge through the proposed dense connectivity structure. Specifically, we perform experiments on various splits of the 21K ImageNet dataset as well as the AWA2 dataset. On the 21K classes of ImageNet, we obtain relative improvements of more than 50% over the previously reported best results, and our proposed ADGPM improves on the GPM by 7.1%. In a realistic setting zero-shot models should generalize well to the unseen classes but still perform well on seen classes [22]. We therefore also evaluate the ability of the models to perform well on the seen classes and illustrate that the performance on the seen classes benefits from our finetuning approach. The source code for the experiments performed in this paper is available at: https://github.com/cyvius96/adgpm.

2 Related Work

Graph convolutional networks are a class of graph neural networks, based on local graph operators [3, 7, 14]. Their advantage is that their graph structure allows the sharing of statistical strength between classes making these methods highly sample efficient. After being introduced in [3], they were extended with an efficient filtering approach based on recurrent Chebyshev polynomials, reducing their computational complexity to the equivalent of the commonly used CNNs in image processing operating on regular grids [7]. [14] further proposed simplifications to improve scalability and robustness and applied their approach to semi-supervised learning on graphs. Their approach is termed graph convolutional network (GCN).

Zero-shot learning has in recent years been considered from various set of viewpoints such as manifold alignment [9, 18], linear auto-encoder [15], and low-rank embedded dictionary learning approaches [10], using semantic relationships based on attributes [21, 29, 11] and relations in knowledge graphs [31, 20, 26, 23]. One of the early works [16] proposed a method based on the idea of a model-of-model class approach, where a model is trained to predict models based on their description. Each class is modeled as a function of its description. This idea has recently been used in another work in [31], the work most similar to our own, where a graph convolutional neural network is trained to predict logistic regression classifiers on top of pre-trained CNN features. [31] proposed to use GCNs [14] to predict a set of logistic regression classifiers, one for each class, on top of pre-trained CNN features in order to predict unseen classes. Their approach has yielded impressive performance on a set of zero-shot learning tasks and can, to the author's knowledge be considered to be the current state-of-the-art.

3 Approach

Here we first formalize the problem of zero-shot learning, provide information on a simple one layer model GPM and then extend this model to the full ADGPM. Our zero-shot learning framework to address this task is illustrated in Figure 2. Inspired by [31] we train a model, in our case the ADGPM, to predict the last layer CNN weights for each class/concept.

3.1 Zero-shot learning

Zero-shot classification aims to predict the class labels of a set of test data points to a set of classes $\mathcal{C}_{te}$. However, unlike in common supervised classification, the test data set points have to be assigned to previously unseen classes, given a L dimensional semantic representation vector $z \in \mathbb{R}^L$ per class $\mathcal{C}$ and a set of training data points $\mathcal{D}_{tr} = \{(\vec{X}_i, c_i)\, i = 1, ..., N\}$, where $\vec{X}_i$ denotes the i-th training image and $c_i \in \mathcal{C}_{tr}$ the corresponding class label. Here $\mathcal{C}$ denotes the set of all classes and $\mathcal{C}_{te}$ and $\mathcal{C}_{tr}$ the test and training classes, respectively. Note that training and test classes are disjoint $\mathcal{C}_{te} \cap \mathcal{C}_{tr} = \emptyset$ for the zero-shot learning task. In this work, we perform zero-shot classification by using the word embedding of the class labels and the knowledge graph to predict classifiers for each unknown class in form of last layer CNN weights.

3.2 Graph Propagation Module

Given a graph with N nodes and with C input features per node, $X \in \mathbb{R}^{N \times C}$ is used to denote the feature matrix. Here each node represents a distinct concept/class in the classification task. In this

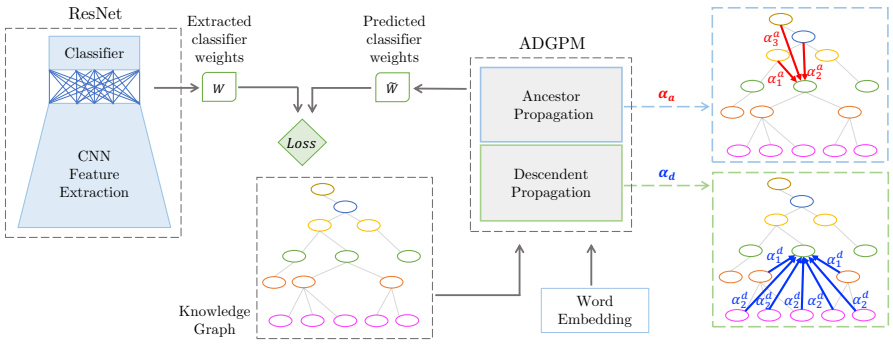


Figure 2: ADGPM is trained to predict classifier weights W for each node/class in a graph. These weights are extracted from the final layer of a pre-trained ResNet. The graph is constructed from a knowledge graph and each node is represented by its word embedding (semantic information). The network consists of two phases, a descendent phase where each node receives knowledge form its children and an ancestor phase, where it receives knowledge from its ancestors.

work, each concept is represented by a word vector of the class name. The connections between the classes in the knowledge graph are encoded in form of a symmetric adjacency matrix $A \in \mathbb{R}^{N \times N}$, which also includes self-loops. We employ a simple propagation rule to perform convolutions on the graph

$$H^{(l+1)} = \sigma\left(D^{-1}AH^{(l)}\Theta^{(l)}\right) , \tag{1}$$

where $H^{(l)}$ represents the activations in the l^{th} layer and $\Theta \in \mathbb{R}^{C \times F}$ denotes the trainable weight matrix for layer l. For the first layer, $H^{(0)} = X$. $\sigma(\cdot)$ denotes a nonlinear activation function, in our case ReLU. $D_{ii} = \sum_j A_{ij}$ is a degree matrix $D \in \mathbb{R}^{N \times N}$, which normalizes rows in A to ensure that the scale of the feature representations is not modified by A. Similar to previous work done on graph convolutional neural networks, this propagation rule can be interpreted as a spectral convolution [14]. Guided by experimental evidence, we only employ a single hidden layer model to avoid the smoothing effect and also do not employ the symmetric normalization $D^{-1/2}AD^{-1/2}$ that was used in [31].

The model is trained to predict the classifier weights for the seen classes by optimizing the loss

$$\mathcal{L} = \frac{1}{2M}\sum_{i=1}^{M}\sum_{j=1}^{L}(W_{i,j} - \widetilde{W}_{i,j})^2 , \tag{2}$$

where $\widetilde{W} \in \mathbb{R}^{M \times L}$ denotes the prediction of GPM for the known classes and therefore corresponds to the M rows of $H^{(2)}$, which correspond to the training classes. M denotes the number of training classes and L denotes the dimensionality of the weight vectors. The ground truth weights are obtained by extracting the last layer weights of a pre-trained CNN and denoted as $W \in \mathbb{R}^{M \times L}$.

During testing, the features of new images are extracted from the CNN and the GPM predicted classifiers are used to classify the features.

3.3 Attentive Dense Graph Propagation Module

Our ADGPM for zero-shot learning aims to utilize the hierarchical graph structure for the zero-shot learning task and avoids the dilution of knowledge by intermediate nodes. This is achieved using a dense graph connectivity scheme consisting of two phases, namely the descendant propagation phase and the ancestor propagation phase. This two-phase approach further enables the model to learn separate relations between a node and its parents and a node and its children. Unlike in the GPM, we do not use the knowledge graph relations directly as an adjacency graph to include information from neighbors further away. We do therefore not suffer from the problem of knowledge being washed out due to averaging over the graph. Instead, we introduce two separate connectivity patterns, one where nodes are connected to all their ancestors and one where nodes are connected to all descendants. We utilize two adjacency matrices $A_a \in \mathbb{R}^{N \times N}$ that denotes the connections between nodes and their ancestors and adjacency matrix A_d that denotes the connections between nodes and their descendants.

Note, $A_d = A_a^T$. Unlike in previous approaches, this connectivity pattern allows nodes direct access to knowledge in their extended neighborhood as opposed to knowledge that has been modified by intermediate nodes. Note that both these adjacency matrices include self-loops. The connection pattern is illustrated in Figure 1. The same propagation rule as in Equation 1 is applied consecutively for the two connectivity patterns leading to the overall ADGPM propagation rule

$$H = \sigma\left(D_a^{-1} A_a \sigma\left(D_d^{-1} A_d X \Theta_d\right) \Theta_a\right) \ . \tag{3}$$

Attention weights In order to allow ADGPM to weigh the contribution of various neighbors in the dense graph, we propose an attention scheme that weighs a given nodes neighbors based on the graph distance from the node. Note, the distance is computed on the knowledge graph and not the dense graph. We use $w^a = \{w_i^a\}_{i=0}^K$ and $w^d = \{w_i^d\}_{i=0}^K$ to denote the attention weights for the ancestor and the descendent propagation phase, respectively. w_i^a and w_i^d correspond to weights for nodes that are i hops away from the given node. w_0^a, w_0^d correspond to self-loops and w_K^a, w_K^d correspond to the weights for nodes further than $K-1$ hops away. We normalize the weights using a softmax function $\alpha_k^a = \text{softmax}(w_k^a) = \frac{\exp(w_k^a)}{\sum_{i=0}^K \exp(w_i^a)}$. Similarly, $\alpha_k^d = \text{softmax}(w_k^d)$. The weighted propagation rule in Equation 3 becomes

$$H = \sigma\left(\sum_{k=0}^{K} \alpha_k^a D_a^{-1} A_a \sigma\left(\sum_{k=0}^{K} \alpha_k^d D_d^{-1} A_d X \Theta_d\right) \Theta_a\right) \ , \tag{4}$$

where A_k is used to denote the adjacency matrix that only contains the k-hop edges.

3.4 Finetuning

Training of the proposed model is done in two stages, where the first stage trains the ADGPM to predict the last layer weights of a pre-trained CNN using equation 2. Note, $\widetilde{W}$, in this case, contains the M rows of H, which correspond to the training classes. In order to allow the feature representation of the CNN to adapt to the new class classifiers, we train the CNN in a second stage where the last layer weights are fixed to the predicted weights of the training classes in the ADGPM and only the feature representation is updated. This can be viewed as utilizing the ADGPM as a constrained for the CNN finetune, as we indirectly incorporate the graph information in order to constrain the CNN output space.

3.5 Training details

We use a ResNet-50 [12] model that has been pre-trained on the ImageNet 2012 dataset. Following [31], we use the GloVe text model [25], which has been trained on the Wikipedia dataset, as the feature representation of our concepts in the graph. The ADGPM model consists of two layers as illustrated in Equation 3 with feature dimensions of 2048 and the final output dimension corresponds to the number of weights in the last layer of the ResNet-50 architecture, 2049 for weights and bias. Following the observation of [31], we perform L2-Normalization on the outputs as it regularizes the outputs into similar ranges. Similarly, we also normalize the ground truth weights produced by the CNN. We further make use of Dropout [30] with a dropout rate of 0.5 in each layer. The model is trained for 3000 epochs with a learning rate of 0.001 and weight decay of 0.0005 using Adam [13]. We make use of leaky ReLUs with a negative slope of 0.2. The number of attention values per phase K was set to 5 as additional weights had diminishing returns. The proposed ADGPM model is implemented in PyTorch [24] and training and testing are performed on a GTX 1080Ti GPU. Finetuning is done for 20 epochs using SGD with a learning rate of 0.0001 and momentum of 0.9.

4 Experiments

We performed a comparative evaluation of the proposed GPM and ADGPM against previous state-of-the-art on two common zero-shot learning datasets, namely ImageNet and AWA2.

ImageNet [8] is the largest commonly used dataset for zero-shot learning. In our work, we follow the train/test split suggested by [11], who proposed to use the 21K ImageNet dataset for zero-shot evaluation. They define three tasks in increasing difficulty, denoted as "2-hops", "3-hops" and "All".

ResNet: baboon, langur, koala, macaque, madagascar cat
GCNZ: phalanger, kangaroo, lemur, marsupial, **tree squirrel**
GPM: phalanger, kangaroo,**tree squirrel**, lemur, tree wallaby
ADGPM: **tree squirrel**, kangaroo, phalanger, lemur, tree wallaby

ResNet: sea lion, oystercatcher, king penguin, ruddy turnstone, meerkat
GCNZ: pelagic bird, wandering albatross, **penguin**, black-footed albatross, california sea lion
GPM: **penguin**, california sea lion, steller sea lion, south american sea lion, australian sea lion
ADGPM: **penguin**, california sea lion, south american sea lion, hoary marmot, yellowbelly marmot

ResNet: plane, shoe shop, hook, sundial, electric fan
GCNZ: fastener, block plane, jointer, dovetail plane, scrub plane
GPM: dovetail plane, beading plane, jointer, circular plane, block plane
ADGPM: circular plane, dovetail plane, **opener**, jointer, router plane

ResNet: bookcase, entertainment center, library, file, comic book
GCNZ: wall unit, furniture, secretary, davenport, writing desk
GPM: furniture, office furniture, dining-room table, wall unit, writing desk
ADGPM: furniture, office furniture, chest of drawers, **cabinet**, wall unit

Figure 3: Qualitative result comparison on Imagenet. The correct class is highlighted in bold. We report the top-5 classification results.

Hops refer to the distance that classes are away from the ImageNet 2012 1K classes in the ImageNet hierarchy and thus is a measure of how far unseen classes are away from seen classes. "2-hops" contains all the classes two hops from the seen classes and consists of 1,589 classes, while "3-hops" contains 7,860 classes. "All" contains all 20,841 classes. None of the classes are contained in the ImageNet 2012 dataset, which was used to pre-train the ResNet-50 model. Mirroring experiment setup in [11, 22, 31] we further evaluate the performance when training categories are included as potential labels. Note that since the only difference is the number of classes during testing, the model does not have to be retrained. We denote the splits as "2-hops+1K", "3-hops+1K", "All+1K".

AWA2 [32] is a replacement for the original AWA dataset and represents more traditional zero-shot learning datasets, where most approaches rely on class-attribute information. It consists of 50 animal classes, with a total of 37,322 images and an average of 746 per class. The dataset further consists of 85-attribute features per class. We report results on the proposed split in [32] to ensure that there is no overlap between the test classes and the ImageNet 2012 dataset. In the proposed split, 40 classes are used for training and 10 for testing.

AWA2 test classes are contained in the 21K ImageNet classes and several of the training classes (24 out of 40) that are in the proposed split overlap with the ImageNet 2012 dataset. We, therefore, use a unified approach for both datasets. Note, we are not using attribute information but rely on the ImageNet graph obtained from WordNet and word embedding as semantic class information.

4.1 Baseline approaches

We compare our ADGPM to the following baselines: **Devise** [11] linearly maps visual information in form of features extracted by a convolutional neural network to the semantic word-embedding space. The transformation is learned using a hinge ranking loss. Classification is performed by assigning the visual features to the class of the nearest word-embedding. **ConSE** [22] projects image features into a semantic word embedding space as a convex combination of the T closest seen classes semantic embedding weighted by the probabilities that the image belongs to the seen classes. The probabilities are predicted using a pre-trained convolutional classifier. Similar to Devise, ConSE assigns images to the nearest classes in the embedding space. **EXEM** [5] creates visual class exemplars by averaging the PCA projections of images belonging to the same seen class. A kernel-based regressor is then learned to map a semantic embedding vector to the class exemplar. For zero-shot learning visual exemplars can be predicted for the unseen classes using the learned regressor and images can be assigned using nearest neighbor classification. **SYNC** [4] aligns a semantic space (e.g., the word-embedding space) with a visual model space, adds a set of phantom object classes in order to connect seen and unseen classes, and derives new embeddings as convex combination of these phantom classes. **GCNZ** [31] represents the current state of the art and is the approach most related to our proposed ADGPM. A GCN is trained to predict last layer weights of a convolutional neural network.

4.2 Comparison to state-of-the-art methods: ImageNet

Quantitative results for the comparison on the ImageNet datasets are shown in Table 1. We use DGPM to denote the accuracy that is achieved after training the ADGPM model without attention and before finetuning the CNN. DGPM(f) and ADGPM(f) are used to denote the results for DGPM and ADGPM after finetuning, respectively. We further report the accuracy achieved by the one

Table 1: Top-k accuracy for the different models on the ImageNet dataset. Accuracy when only testing on unseen classes.

Test set	Model	Hit@k (%) 1	2	5	10	20
2-hops	ConSE [4]	8.3	12.9	21.8	30.9	41.7
	EXEM [5]	12.5	19.5	32.3	43.7	55.2
	SYNC [4]	10.5	17.7	28.6	40.1	52.0
	GCNZ [31]	19.8	33.3	53.2	65.4	74.6
	GPM	24.8	38.3	57.5	69.9	79.6
	DGPM	23.8	36.9	56.2	69.1	78.6
	ADGPM	24.6	37.8	56.9	69.6	79.3
	GPM(f)	26.2	40.4	60.2	71.9	81.0
	DGPM(f)	25.4	39.5	59.9	72.0	80.9
	ADGPM(f)	**26.6**	**40.7**	**60.3**	**72.3**	**81.3**
3-hops	ConSE [4]	2.6	4.1	7.3	11.1	16.4
	SYNC [4]	2.9	4.9	9.2	14.2	20.9
	EXEM [5]	3.6	5.9	10.7	16.1	23.1
	GCNZ [31]	4.1	7.5	14.2	20.2	27.7
	GPM(f)	6.0	10.4	18.9	27.2	36.9
	DGPM(f)	6.2	10.5	19.2	**27.7**	**37.7**
	ADGPM(f)	**6.3**	**10.7**	**19.3**	**27.7**	**37.7**
All	ConSE [4]	1.3	2.1	3.8	5.8	8.7
	SYNC [4]	1.4	2.4	4.5	7.1	10.9
	EXEM [5]	1.8	2.9	5.3	8.2	12.2
	GCNZ [31]	1.8	3.3	6.3	9.1	12.7
	GPM(f)	2.8	4.9	9.1	13.5	19.3
	DGPM(f)	2.9	**5.0**	**9.3**	**13.9**	**19.9**
	ADGPM(f)	**3.0**	**5.0**	**9.3**	**13.9**	19.8

Table 2: Top-k accuracy for the different models on the ImageNet dataset. Accuracy when testing on seen and unseen classes.

Test set	Model	Hit@k (%) 1	2	5	10	20
2-hops+1K	DeViSE [11]	0.8	2.7	7.9	14.2	22.7
	ConSE [22]	0.3	6.2	17.0	24.9	33.5
	ConSE [31]	0.1	11.2	24.3	29.1	32.7
	GCNZ [31]	9.7	20.4	42.6	57.0	68.2
	GPM	11.4	25.2	47.7	61.8	73.3
	DGPM	9.6	22.5	44.9	60.6	72.3
	ADGPM	10.6	24.1	46.2	61.4	73.1
	GPM(f)	**11.9**	**27.0**	**50.8**	65.1	75.9
	DGPM(f)	10.6	25.1	49.2	64.6	75.5
	ADGPM(f)	10.3	26.4	50.3	**65.2**	**76.0**
3-hops+1K	DeViSE [11]	0.5	1.4	3.4	5.9	9.7
	ConSE [22]	0.2	2.2	5.9	9.7	14.3
	ConSE [31]	0.2	3.2	7.3	10.0	12.2
	GCNZ [31]	2.2	5.1	11.9	18.0	25.6
	GPM(f)	**3.2**	**7.1**	**16.1**	24.6	34.6
	DGPM(f)	3.0	6.9	15.8	24.6	35.0
	ADGPM(f)	2.9	**7.1**	**16.1**	**24.9**	**35.1**
All+1K	DeViSE [11]	0.3	0.8	1.9	3.2	5.3
	ConSE [22]	0.2	1.2	3.0	5.0	7.5
	ConSE [31]	0.1	1.5	3.5	4.9	6.2
	GCNZ [31]	1.0	2.3	5.3	8.1	11.7
	GPM(f)	**1.5**	**3.4**	7.8	12.3	18.2
	DGPM(f)	**1.5**	**3.4**	7.8	12.4	18.6
	ADGPM(f)	1.4	**3.4**	**7.9**	**12.6**	**18.7**

layer GPM model and a finetuned version of it GPM(f). Compared to previous results such as ConSE [4], EXEM [5], and GCNZ [31] our proposed methods outperform the previous results with a considerable margin, achieving, for instance, more than 50% relative improvement for Top-1 accuracy on the 21K ImageNet All dataset. We observe that our methods especially outperform the baseline models on the "All" task, illustrating the potential of our methods to more efficiently propagate knowledge. Furthermore, we perform an ablation experiment for the 2-hops dataset where we analyze the effect of adding attention and finetuning to the model and observe that finetuning the DGPM model and introducing attention consistently leads to improvements in model performance. Results further illustrate that the proposed ADGPM(f), which learns both feature extraction and the zero-shot classifiers, achieves better performance than all variants, demonstrating that our dense attention scheme can make features more general and incorporates more graph information.

Qualitative results of the finetuned ADGPM and GPM are shown in Figure 4. Example images from unseen test classes are displayed and we compare the results of our proposed ADGPM and GPM to results produced by a pre-trained ResNet. Note, ResNet can only predict training classes while the others predict classes not seen in training. For comparison we also provide results for our re-implementation of GCNZ. We observe that GPM and ADGPM generally provide coherent top-5 results. All methods struggle to predict the *opener* and tend to predict some type of *plane* instead, however, ADGPM does include *opener* in the top-5 results. We further observe that the prediction task on this dataset for zero-shot learning is difficult as it contains classes of fine granularity, such as many different types of squirrels, planes and furniture. Additional examples are provided in the appendix.

Study of the generalized setting. Results are shown in Table 2. This includes the training labels as potential labels during classification of the zero-shot examples. For the baselines, we include two implementations of ConSE, one that uses AlexNet as a backbone [22] and one that uses ResNet-50 [31]. Compared to Table 1, we observe that the accuracy is considerably lower, but GPM, DGPM and ADGPM still outperform the previous state-of-the-art approach GCNZ on most metrics. For the 2-hop dataset DGPM and GCNZ perform similar on the Top-1 accuracy, but DGPM performs better on the remaining Top-k measures. Results for the finetuned methods show that similar to the non-generalized setting, considerable improvements can be obtained compared to the baseline. Note,

Table 3: Performance on the seen ImageNet classes. ResNet represents ideal performance.

Model	Hit@k (%)			
	1	2	5	10
ResNet	75.1	85.5	92.7	95.7
GCNZ(us)	38.3	62.9	82.3	89.8
GPM	44.8	63.8	80.1	87.0
DGPM	48.7	63.6	78.8	85.6
ADGPM	47.0	63.5	79.1	86.0
GPM(f)	49.1	68.7	**83.9**	**89.4**
DGPM(f)	53.0	68.5	83.7	89.2
ADGPM(f)	**54.6**	**69.7**	83.8	89.1

Table 4: Results for unseen classes in AWA2.

Model	ACC (%)
ConSE [22]	44.5
Devise [11]	59.7
SYNC [4]	46.6
GCNZ(us)	70.7
GPM	77.3
DGPM	67.2
ADGPM	76.0
GPM(f)	**79.3**
DGPM(f)	70.4
ADGPM(f)	77.3

that the difference in performance between GPM(f) and ADGPM(f) for the generalized setting is less than in the non-generalized setting.

Study of domain shift issue. Performance on seen classes also needs to be considered as there is a tradeoff. In a realistic setting, zero-shot learning models should perform well on both seen and unseen classes. To put the zero-shot performance into perspective, we perform experiments where we analyze how the model's performance on the original 1000 seen classes is affected by domain shift as additional unseen classes (all 2-hop classes) are introduced. Table 3 shows the results when the model is tested on the validation dataset from ImageNet 2012. We compare the performance to our re-implementation of the GCNZ model with ResNet-50 backbone and also the performance from the original ResNet-50 model, which is trained only on the seen classes. It can be observed that all our methods outperform GCNZ on Hit@1 and Hit@2 accuracy. We also observe that finetuning the feature extraction part of the CNN improves the performance on the train classes considerably, decreasing the gap to the performance of the pre-trained ResNet-50.

Scalability. To obtain good scalability it is important that the adjacency matrix A is a sparse matrix so that the complexity of computing $D^{-1}AX\Theta$ is linearly proportional to the number of edges present in A. Our approach utilizes the structure of knowledge graphs, where entities only have few ancestors and descendents, to ensure this. The adjacency matrix for the ImageNet hierarchy used in our experiments, for instance, has a density of 9.3×10^{-5}, while our dense connections only increase the density of the adjacency matrix to 19.1×10^{-5}.

4.3 Comparison to state-of-the-art methods: AWA2

The results are presented in Table 4. We observe that GPM, DGPM and ADGPM outperform the baseline approaches. Note that our models differ considerably from the baselines as it does not make use of the attributes provided in the dataset. To illustrate the merits of our approach, we re-implement [31], as it represents the method which is closest related to our approach and also makes use of word embeddings and a knowledge graph. We observe that our methods also outperforms [31], however, the improvement is lower than on the ImageNet dataset, which we believe is due to the arguably simpler task with the number of classes being considerably lower. We observe an additional increase in performance by making use of finetuning. Note, we observe that the simple GPM outperforms the proposed ADGPM. However, similar to the ImageNet experiments, we observe a large improvement by adding the simple attention scheme to DGPM, and therefore believe that a more granular attention scheme can allow ADGPM to outperform GPM. The large improvement that is achieved by adding attention to DGPM illustrates this. The results illustrate that the proposed GPM and ADGPM can universally address the problem of zero-shot learning as long as test classes are represented in the knowledge graph.

5 Conclusion

In contrast to previous approaches using graph convolutional neural networks for zero-shot learning, our proposed GPM and ADGPM tailor the graph approach to the problem by exploiting the hierarchical structure of the knowledge graph. This allowed us to address the problem of knowledge being diluted as it is passed along the graph. Experiments illustrate the ability of the proposed methods, outperforming previous state-of-the-art methods for zero-shot learning. In future work, we aim to

investigate the potential of more advanced attention mechanisms to further improve the performance of ADGPM compared to GPM. The inclusion of additional semantic information for settings where these are available for a subset of nodes is another future direction.

References

[1] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015. 1

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 3

[4] 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*, pages 5327–5336, 2016. 6, 7, 8

[5] S. Changpinyo, W.-L. Chao, and F. Sha. Predicting visual exemplars of unseen classes for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3485, 2017. 6, 7

[6] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European Conference on Computer Vision*, pages 52–68. Springer, 2016. 2

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 3

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5

[9] S. Deutsch, S. Kolouri, K. Kim, Y. Owechko, and S. Soatto. Zero shot learning via multi-scale manifold regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7112–7119, 2017. 3

[10] Z. Ding, M. Shao, and Y. Fu. Low-rank embedded ensemble semantic dictionary for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2050–2058, 2017. 3

[11] 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*, pages 2121–2129, 2013. 1, 3, 5, 6, 7, 8

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 5

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015. 5

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representation*, 2017. 1, 3, 4

[15] 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*, pages 3174–3183, 2017. 2, 3

[16] H. Larochelle, D. Erhan, and Y. Bengio. Zero-data learning of new tasks. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 2*, pages 646–651. AAAI Press, 2008. 3

[17] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 33rd national conference on Artificial intelligence*. AAAI Press, 2018. 2

[18] Y. Li, D. Wang, H. Hu, Y. Lin, and Y. Zhuang. Zero-shot recognition using dual visual-semantic mapping paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 3

[19] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[20] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Proceedings of the European Conference on Computer Vision*, pages 488–501. Springer, 2012. 3

[21] I. Misra, A. Gupta, and M. Hebert. From red wine to red tomato: Composition with context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1792–1801, 2017. 3

[22] 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. *International Conference for Learning Representation*, 2014. 3, 6, 7, 8

[23] 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*, pages 1410–1418, 2009. 1, 3

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems Workshop*, 2017. 5

[25] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical methods in Natural Language Processing*, pages 1532–1543, 2014. 5

[26] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1641–1648. IEEE, 2011. 1, 3

[27] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015. 1

[28] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1481–1488. IEEE, 2011. 1

[29] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013. 1, 3

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 5

[31] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *arXiv preprint arXiv:1803.08035*, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 11

[32] Y. Xian, B. Schiele, and Z. Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017. 6

A Qualitative results

Figure 4 and 5 provide further qualitative results of our finetuned Graph Propagation Module GPM and Attentive Dense Graph Propagation Module ADGPM compared to a standard ResNet and GCNZ, our reimplementation of [31].

ResNet: upright, grand piano, organ, accordion, barbershop
GCNZ: piano, spinet, keyboard instrument, concert grand, **baby grand**
GPM: piano, spinet, concert grand, **baby grand**, keyboard instrument
ADGPM: piano, **baby grand**, concert grand, spinet, keyboard instrument

ResNet: breakwater, aircraft carrier, seashore, wing, sandbar
GCNZ: barrier, **bar**, shore, grate, geological formation
GPM: littoral, **bar**, seaside, barrier, landfall,
ADGPM: **bar**, littoral, shore, seaside, landfall

ResNet: lemon, orange, banana, spaghetti squash, fig
GCNZ: **bitter orange**, temple orange, citrus, sweet orange, edible fruit
GPM: citrus, **bitter orange**, temple orange, sweet orange, edible fruit,
ADGPM: citrus, **bitter orange**, sweet orange, temple orange, edible fruit

ResNet: lycaenid, cabbage butterfly, ringlet, sulphur butterfly, damselfly
GCNZ: pierid, small white, large white, hairstreak, southern cabbage butterfly
GPM: **blue**, hairstreak, copper, pierid, butterfly,
ADGPM: **blue**, hairstreak, copper, pierid, butterfly

ResNet: candle, altar, lighter, lipstick, perfume
GCNZ: vigil light, rushlight, **chandlery**, dip, lamp
GPM: vigil light, rushlight, **chandlery**, dip, high altar
ADGPM: vigil light, **chandlery**, rushlight, dip, flambeau

ResNet: bagel, french loaf, cheeseburger, dough, hotdog
GCNZ: onion bagel, bun, loaf of bread, **cracker**, bread dough
GPM: onion bagel, bun, bread dough, pastry, sandwich
ADGPM: bun, onion bagel, bread dough, **cracker**, pastry

ResNet: walking stick, jacamar, hip, house finch, chainlink fence
GCNZ: **diapheromera**, phasmid, finch, oscine, praying mantis
GPM: **diapheromera**, phasmid, neuropteron, thrush, finch
ADGPM: **diapheromera**, phasmid, thrush, titmouse, oscine

Figure 4: Qualitative result comparison on Imagenet. The correct class is highlighted in bold. We report the top-5 classification results.

ResNet: desktop computer, monitor, screen, computer keyboard, mouse
GCNZ: personal computer, portable computer, planner, computer, computer screen
GPM: personal computer, computer, computer screen, **display**, television monitor
ADGPM: personal computer, background, computer screen, portable computer, **display**

ResNet: bittern, partridge, coucal, ruffed grouse, kite
GCNZ: least bittern, american bittern, **european bittern**, phasianid, crow pheasant
GPM: american bittern, **european bittern**, least bittern, plain turkey, great bustard
ADGPM: american bittern, least bittern, **european bittern**, heron, egret

ResNet: damselfly, dragonfly, lacewing, walking stick, grasshopper
GCNZ: **odonate**, neuropteron, hymenopterous insect, phasmid, brown lacewing
GPM: **odonate**, neuropteron, brown lacewing, green lacewing, phasmid
ADGPM: **odonate**, brown lacewing, green lacewing, neuropteron, phasmid

ResNet: macaw, lorikeet, bee eater, sulphur-crested cockatoo, house finch
GCNZ: lory, **parrot**, rainbow lorikeet, varied lorikeet, cockatoo
GPM: **parrot**, lory, rainbow lorikeet, varied lorikeet, cockatoo
ADGPM: lory, **parrot**, cockatoo, rainbow lorikeet, varied lorikeet

ResNet: grocery store, confectionery, tobacco shop, restaurant, butcher shop
GCNZ: marketplace, greengrocery, **supermarket**, shop, tuck shop
GPM: **supermarket**, marketplace, greengrocery, tuck shop, shop
ADGPM: **supermarket**, greengrocery, marketplace, tuck shop, shop

ResNet: cliff, valley, lakeside, alp, promontory
GCNZ: geological formation, natural elevation, natural depression, mountain, ravine
GPM: **precipice**, crag, natural depression, ravine, natural elevation
ADGPM: natural depression, geological formation, natural elevation, crag, **precipice**

ResNet: church, monastery, dome, bell cote, mosque
GCNZ: kirk, cathedral, abbey, basilica, cathedral
GPM: abbey, cathedral, friary, basilica, cathedral
ADGPM: cathedral, abbey, cathedral, basilica, kirk

true label: **place_of_worship**

Figure 5: Qualitative result comparison on Imagenet. The correct class is highlighted in bold. We report the top-5 classification results.