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[6]

[1]UiT The Arctic University of Norway, [2]Tsinghua University, [3]Sun Yat-sen University, [4]Massachusetts Institute of Technology, [5]Institute of Automation, Chinese Academy of Sciences, [6]Carnegie Mellon University

Abstract

Graph convolutional neural networks have recently shown great potential for the task of zero-shot learning. 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, multi-layer architectures, which are required to propagate knowledge to distant nodes in the graph, dilute the knowledge by performing extensive Laplacian smoothing at each layer and thereby consequently decrease performance. In order to still enjoy the benefit brought by the graph structure while preventing dilution of knowledge from distant nodes, we propose a Dense Graph Propagation (DGP) module with carefully designed direct links among distant nodes. DGP 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. A weighting scheme is further used to weigh their contribution depending on the distance to the node to improve information propagation in the graph. Combined with finetuning of the representations in a two-stage training approach our method outperforms state-of-the-art zero-shot learning approaches.

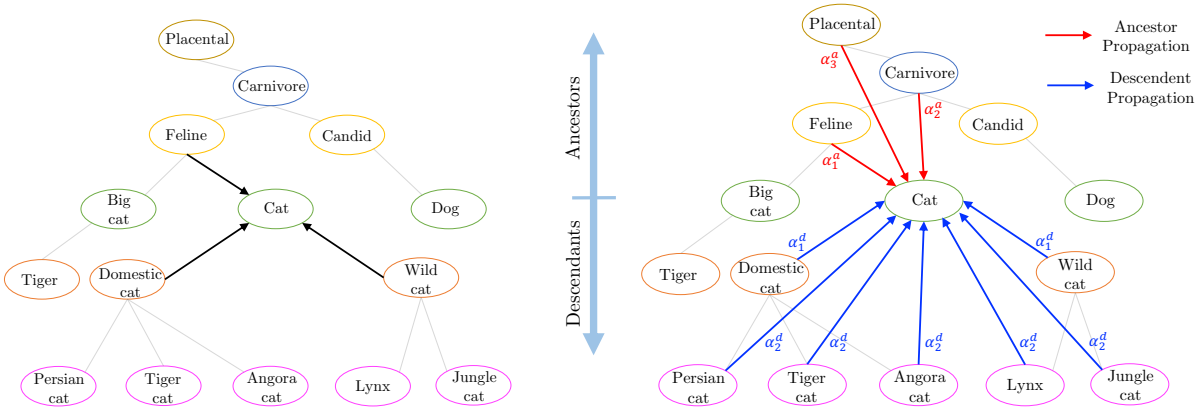
1. Introduction

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. Crucial to this task is precise modeling of class relationships based on prior class knowledge. Previously, prior knowledge has been incorporated in form of semantic descriptions of classes, such as attributes [1, 29, 20] or word embeddings [31, 11, 19], or by using semantic relations such as knowledge graphs [25, 28, 30, 21]. Approaches that use knowledge graphs are less-explored and generally assume 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 [34].

The current state-of-the-art by Wang *et al*. [34] processes the unweighted knowledge graph by exploiting recent developments in neural networks for non-Euclidean spaces, such as graph and manifold spaces [2]. A deep graph convolutional neural network (GCN) [14] is used and the problem is phrased as a regression task, where the GCN is trained to output a classifier for each class by regressing real-valued weight vectors. These weight vectors correspond to the last layer weights of a pretrained convolutional neural network (CNN) and can be viewed as logistic regression classifiers on top of the feature extraction produced by the CNN. GCNs balance model complexity and expressiveness with a simple scalable model 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 [18]. 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 $n \rightarrow \infty$ under some conditions, hence washing out all information [18].

Therefore, we argue that this approach is not ideal for the task of zero-shot learning and that the number of lay-

*Indicates equal contribution.
†Corresponding Author.



(a) Graph Propagation

(b) Dense Graph Propagation

Figure 1: a) Illustration of graph propagation in a GCN [14] for node 'Cat'. Here, graph 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 descendants during the descendant 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. The learned weights α_k^a and α_k^d are used to weigh nodes that are k-hops away from a given node in the ancestor and the descendants phase, respectively.

ers in the GCN should be small in order to avoid smoothing. We illustrate this phenomenon in practice, by showing that a shallow GCN consistently outperforms previously reported results. Choosing a small number of layers, however, has the effect that knowledge will not propagate well through the graph. A 1-layer GCN for instance only considers neighbors that are two hops away in the graph such 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 new connections allow us to propagate information without over-smoothing but remove important structural information in the graph since all descendants/ancestors would be included in the one-hop neighborhood and would be weighed equally when computing the regression weight vector for a given class. To address this issue, we further propose a weighting scheme that considers the distance between nodes in order to weigh the contribution of different nodes. This allows the model to not only recover the original structure in the graph but further provides an additional degree of flexibility that enhances the inference capabilities of our model. Introducing distance-based shared weights also has the benefit that it only adds a minimal amount of parameters, is computationally efficient, and balances model flexibility and restrictiveness to allow good predictions for the nodes of the unseen classes. Fig. 1 illustrates the difference in the way knowledge is propagated in this proposed Dense Graph Propagation (DGP) module compared to a GCN layer.

To 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 DGP is trained to predict the last layer CNN weights. In the second phase, we replace the last layer weights of the CNN with the weights predicted by the DGP, freeze the weights and finetune the remaining weights of the CNN by optimizing the cross entropy classification loss on the seen classes.

Our main contributions are the following:

- An analysis of our intuitions for zero-shot learning and an illustration of how these intuitions can be combined to design a DGP that outperforms previous state-of-the-art approaches.[1]
- Our DGP module, which explicitly exploits the hierarchical structure of the knowledge graph to perform zero-shot learning by efficiently propagating knowledge through the proposed dense connectivity structure.
- A novel weighting scheme for DGP where weights are learned based on the distance between nodes.
- Experimental results on various splits of the 21K ImageNet dataset, a popular large-scale dataset for zero-shot learning. We obtain relative improvements of more than 50% over previously reported best results.

2. Related Work

Graph convolutional networks are a class of graph neural networks, based on local graph operators [3, 7, 14].

[1]The source code for the experiments performed in this paper is available at: https://github.com/cyvius96/adgpm.

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 Bruna *et al*. [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]. Kipf *et al*. [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) and provides the foundation for the model in this paper.

Zero-shot learning has in recent years been considered from various set of viewpoints such as manifold alignment [9, 19], linear auto-encoder [15], and low-rank embedded dictionary learning approaches [10], using semantic relationships based on attributes [23, 31, 11] and relations in knowledge graphs [34, 22, 28, 25]. One of the early works [17] proposed a method based on the idea of a model-of-models approach, where a model is trained to predict class 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 Wang *et al*. [34], 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 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 the current state-of-the-art.

3. Approach

Here we first formalize the problem of zero-shot learning and provide information on how a GCN model can be utilized for the task. We then describe our proposed model DGP.

Let $\mathcal{C}$ denote the set of all classes and $\mathcal{C}_{te}$ and $\mathcal{C}_{tr}$ the test and training classes, respectively. Further, assume that the training and test classes are disjoint $\mathcal{C}_{te} \cap \mathcal{C}_{tr} = \emptyset$ and that we are given a S dimensional semantic representation vector $z \in \mathbb{R}^S$ for all classes 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. In this setting, zero-shot classification aims to predict the class labels of a set of test data points to the set of classes $\mathcal{C}_{te}$. Note that, unlike traditional classification, the test data set points have to be assigned to previously unseen classes.

3.1. Graph Convolutional Networks for Zero-Shot Learning

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. Our zero-shot learning framework is illustrated in Fig. 2. The last layer CNN weights are interpreted as a class-specific classifier for a given output class on top of the extracted CNN features. The zero-shot task can then be expressed as predicting a new set of weights for each of the unseen classes in order to extend the output layer of the CNN. Our DGP takes as input the combined knowledge graph for all seen and unseen classes, where each class is represented by a word embedding vector that encodes the class name. It is then trained to predict the last layer CNN weights for all (seen and unseen) classes in a semi-supervised manner. Exploiting the knowledge graph allows us to capture semantic relationships between classes, while the word embedding provides a semantic description of each specific class. During inference, the predicted weights can then be used to extend the set of output classes in the original CNN to enable classification of datapoints from unseen classes.

More specifically, given a graph with N nodes and S input features per node, $X \in \mathbb{R}^{N \times S}$ denotes the feature matrix. Here each node represents one distinct concept/class in the classification task and each concept is represented by the 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 $\Theta \in \mathbb{R}^{S \times F}$ denotes the trainable weight matrix for layer l with F corresponding to the number of learned filters. For the first layer, $H^{(0)} = X$. $\sigma(\cdot)$ denotes a nonlinear activation function, in our case a Leaky 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. Similarly to previous work done on graph convolutional neural networks, this propagation rule can be interpreted as a spectral convolution [14].

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

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

where $\widetilde{W} \in \mathbb{R}^{M \times P}$ denotes the prediction of the GCN for the known classes and therefore corresponds to the M rows of the GCN output, which correspond to the training classes. M denotes the number of training classes and P 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

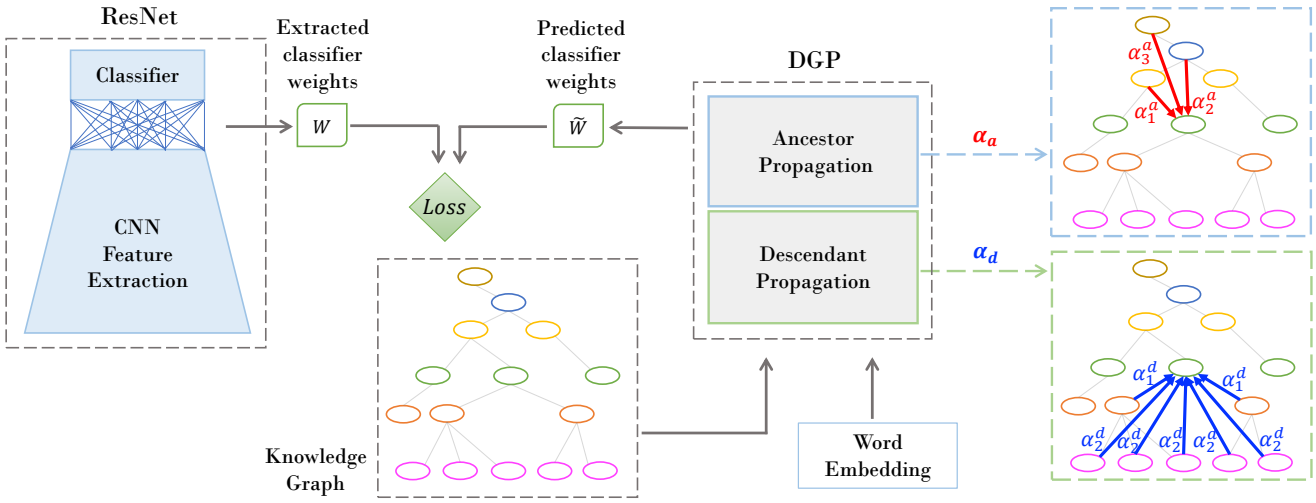


Figure 2: DGP is trained to predict classifier weights W for each node/class in a graph. The weights for the training classes are extracted from the final layer of a pre-trained ResNet. The graph is constructed from a knowledge graph and each node is represented by a vector that encodes semantic class information, in our experiments the classes word embedding. The network consists of two phases, a descendant phase where each node receives knowledge form its descendants and an ancestor phase, where it receives knowledge from its ancestors.

$W \in \mathbb{R}^{M \times P}$. During the inference phase, the features of new images are extracted from the CNN and the classifiers predicted by the GCN are used to classify the features.

However, the Laplacian smoothing operation in matrix form can be written as $(I - \gamma D^{-1} L)H$, as also noted in Li *et al*. [18]. Substituting the graph Laplacian with its definition $L = D - A$ the operation simplifies for $\gamma = 1$ (looking only at the immediate neighbors) to $D^{-1}AH$. This corresponds in parts to the graph convolution operation in Eq. 1. Thus, repeatedly applying Eq. 1 in a multi-layer GCN architecture will lead to repeated Laplacian smoothing, thus diluting the information. Empirical evidence is provided in the model analysis section (Sec. 4.4).

3.2. Dense Graph Propagation Module

Our DGP for zero-shot learning aims to use the hierarchical graph structure for the zero-shot learning task and avoids dilution of knowledge by intermediate nodes. This is achieved using a dense graph connectivity scheme consisting of two phases, namely descendant propagation and ancestor propagation. This two-phase approach further enables the model to learn separate relations between a node and its ancestors and a node and its descendants. Table 6 in the model analysis section provides empirical evidence for this choice. Unlike the GCN, 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 use two adjacency matrices: $A_a \in \mathbb{R}^{N \times N}$ denotes the connections from nodes to their ancestors, whereas A_d denotes the connections from nodes to their descendants. Note, as a given node is the descendant of its ancestors, the difference between the two adjacency matrices is a reversal of their edges $A_d = A_a^T$. Unlike 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 Fig. 1. The same propagation rule as in Eq. 1 is applied consecutively for the two connectivity patterns leading to the overall DGP propagation rule

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

Distance weighting scheme In order to allow DGP to weigh the contribution of various neighbors in the dense graph, we propose a weighting scheme that weighs a given node's 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 learned weights for the ancestor and the descendant 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 all 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 Eq. 3 becomes

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

where A_k^a and A_k^d denote the parts of the adjacency matrices that only contain the k-hop edges for the ancestor and descendant propagation phase, respectively. D_k^a and D_k^d are the corresponding degree matrices for A_k^a and A_k^d. As weights are shared across the graph, the proposed weighting scheme only adds $2 \times (K+1)$ parameters to the model, where K tends to be small ($K = 4$ in our experiments).

Our proposed weighting scheme is related to the attention mechanisms in graph convolutional neural networks [33]. However, unlike attention approaches, our weighting scheme adds only a negligible amount of parameters and does not add the potentially considerable memory overhead of attention approaches. Further, in our zero-shot learning setting, we observed a drop in performance when including the attention approach proposed in [33]. We hypothesize that this is due to the fact that a more complex model will be more prone to overfit given the limited amount of labeled data (sparsely labeled graph). Results are provided in the supplementary material.

3.3. Finetuning

Training is done in two stages, where the first stage trains the DGP to predict the last layer weights of a pre-trained CNN using Eq. 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 by optimizing the cross-entropy classification loss on the seen classes in a second stage. During this stage, the last layer weights are fixed to the predicted weights of the training classes in the DGP and only the feature representation is updated. This can be viewed as using the DGP as a constraint for the CNN, as we indirectly incorporate the graph information to constrain the CNN output space.

4. Experiments

We perform a comparative evaluation of the DGP against previous state-of-the-art on the ImageNet dataset [8], the largest commonly used dataset for zero-shot learning [2]. In our work, we follow the train/test split suggested by Frome *et al*. [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". 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 within two hops from the seen classes and consists of roughly 1.5K classes, while "3-hops" contains about 7.8K classes. "All" contains close to 21K classes. None of the classes are contained in the ImageNet 2012 dataset, which was used to pre-train the ResNet-50 model. Mirroring the experiment setup in [11, 24, 34] 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 the inference phase, the model does not have to be retrained. We denote the splits as "2-hops+1K", "3-hops+1K", "All+1K".

[2] Additional experiments have been performed on the AWA2 dataset and can be found in the supplementary material.

4.1. Training details

We use a ResNet-50 [12] model that has been pre-trained on the ImageNet 2012 dataset. Following Wang *et al*. [34], we use the GloVe text model [27] trained on the Wikipedia dataset as the feature representation of our concepts in the graph. The DGP model consists of two layers as illustrated in Eq. 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 Wang *et al*. [34], 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 [32] 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 values per phase K was set to 4 as additional weights had diminishing returns. The proposed DGP model is implemented in PyTorch [26] 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.

4.2. Comparing approaches

We compare our DGP to the following approaches: **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** [24] 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 embed-

Table 1: Top-k accuracy for the different models on the ImageNet dataset. Accuracy when only testing on unseen classes. Results indicated with *, †, and ‡ are taken from [4], [5], and [34], respectively.

Test set	Model	Hit@k (%) 1	2	5	10	20
2-hops	ConSE*	8.3	12.9	21.8	30.9	41.7
	SYNC*	10.5	17.7	28.6	40.1	52.0
	EXEM†	12.5	19.5	32.3	43.7	55.2
	GCNZ‡	19.8	33.3	53.2	65.4	74.6
	SGCN (ours)	26.2	40.4	60.2	71.9	81.0
	DGP (ours)	**26.6**	**40.7**	**60.3**	**72.3**	**81.3**
3-hops	ConSE*	2.6	4.1	7.3	11.1	16.4
	SYNC*	2.9	4.9	9.2	14.2	20.9
	EXEM†	3.6	5.9	10.7	16.1	23.1
	GCNZ‡	4.1	7.5	14.2	20.2	27.7
	SGCN (ours)	6.0	10.4	18.9	27.2	36.9
	DGP (ours)	**6.3**	**10.7**	**19.3**	**27.7**	**37.7**
All	ConSE*	1.3	2.1	3.8	5.8	8.7
	SYNC*	1.4	2.4	4.5	7.1	10.9
	EXEM†	1.8	2.9	5.3	8.2	12.2
	GCNZ‡	1.8	3.3	6.3	9.1	12.7
	SGCN (ours)	2.8	4.9	9.1	13.5	19.3
	DGP (ours)	**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. Results indicated with ††, ‡‡, and ‡ are taken from [11], [24], and [34], respectively.

Test set	Model	Hit@k (%) 1	2	5	10	20
2-hops+1K	DeViSE††	0.8	2.7	7.9	14.2	22.7
	ConSE‡‡	0.3	6.2	17.0	24.9	33.5
	ConSE‡	0.1	11.2	24.3	29.1	32.7
	GCNZ‡	9.7	20.4	42.6	57.0	68.2
	SGCN (ours)	**11.9**	**27.0**	**50.8**	65.1	75.9
	DGP (ours)	10.3	26.4	50.3	**65.2**	**76.0**
3-hops+1K	DeViSE††	0.5	1.4	3.4	5.9	9.7
	ConSE‡‡	0.2	2.2	5.9	9.7	14.3
	ConSE‡	0.2	3.2	7.3	10.0	12.2
	GCNZ‡	2.2	5.1	11.9	18.0	25.6
	SGCN (ours)	**3.2**	**7.1**	**16.1**	24.6	34.6
	DGP (ours)	2.9	**7.1**	**16.1**	**24.9**	**35.1**
All+1K	DeViSE††	0.3	0.8	1.9	3.2	5.3
	ConSE‡‡	0.2	1.2	3.0	5.0	7.5
	ConSE‡	0.1	1.5	3.5	4.9	6.2
	GCNZ‡	1.0	2.3	5.3	8.1	11.7
	SGCN (ours)	**1.5**	**3.4**	7.8	12.3	18.2
	DGP (ours)	1.4	**3.4**	**7.9**	**12.6**	**18.7**

ding 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 a convex combination of these phantom classes. **GCNZ** [34] represents the current state of the art and is the approach most related to our proposed DGP. A GCN is trained to predict last layer weights of a convolutional neural network.

Guided by experimental evidence (see our analysis in Table 5 in the model analysis section) and our intuition that extensive smoothing is a disadvantage for the weight regression in zero-shot learning, we add a single-hidden-layer GCN (**SGCN**) with non-symmetric normalization ($D^{-1}A$) (as defined in Eq. 1) as another baseline. Note, GCNZ made use of a symmetric normalization ($D^{-1/2}AD^{-1/2}$) but our experimental evaluation indicates that the difference is negligible. For the interested reader, an analysis of the effect of the changes between GCN and SGCN is included in the supplementary material. SGCN further yields a better baseline since our proposed DGP also utilizes the non-symmetric normalization. As DGP, our SGCN model makes use of the proposed two-stage finetuning approach.

4.3. Comparison to state-of-the-art methods

Quantitative results for the comparison on the ImageNet datasets are shown in Table 1. Compared to previous results such as ConSE [4], EXEM [5], and GCNZ [34] 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. DGP also achieves consistent improvements over the SGCN model. We observed that finetuning consistently improved performance for both models in all our experiments. Ablation studies that highlight the impact of finetuning and weighting of neighbors for the 2-hop scenario can be found in Table 3. DGP(-wf) is used to denote the accuracy that is achieved after training the DGP model without weighting (adding no weights in Eq. 4) and without finetuning. DGP(-w) and DGP(-f) are used to denote the results for DGP without weighting and DGP without finetuning, respectively. We further report the accuracy achieved by the SGCN model without finetuning (SGCN(-f)). We observe that the proposed weighting scheme, which allows distant neighbors to have less impact, is crucial for

ResNet: baboon, langur, koala, macaque, madagascar cat
GCNZ: phalanger, kangaroo, lemur, marsupial, **tree squirrel**
SGCN: phalanger, kangaroo,**tree squirrel**, lemur, tree wallaby
DGP: **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
SGCN: **penguin**, california sea lion, steller sea lion, south american sea lion, australian sea lion
DGP: **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
SGCN: dovetail plane, beading plane, jointer, circular plane, block plane
DGP: circular plane, dovetail plane, **opener**, jointer, router plane

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

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

the dense approach. Further, finetuning the model consistently leads to improved results.

Qualitative results of DGP and the SGCN are shown in Fig. 3. Example images from unseen test classes are displayed and we compare the results of our proposed DGP and the SGCN 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 the SGCN and DGP generally provide coherent top-5 results. All methods struggle to predict the *opener* and tend to predict some type of *plane* instead, however, DGP 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 supplementary material.

Testing including training classifiers. Following the example of [11, 24, 34], we also report the results when including both training labels and testing labels as potential labels during classification of the zero-shot examples. Results are shown in Table 2. For the baselines, we include two implementations of ConSE, one that uses AlexNet as a backbone [24] and one that uses ResNet-50 [34]. Compared to Table 1, we observe that the accuracy is considerably lower, but the SGCN and DGP still outperform the previous state-of-the-art approach GCNZ. SGCN outperforms DGP for low k in the Top-k accuracy measure especially for the 2-hops setting, while DGP outperforms SGCN for larger k. We observe that DGP tends to favor prediction to the closest training classes for its Top-1 prediction (see Table 4). However, this is not necessarily a drawback and is a well-known tradeoff [6] between performing well on the unseen classes and the seen classes, which are not considered in this setting. This tradeoff can be controlled by including a novelty detector, which predicts if an image comes from the seen or unseen classes as done in [31] and then assign it to the zero-shot classifier or a classifier trained on the seen classes. Another approach is calibrated stacking [6], which rescales the prediction scores of the known classes.

Table 3: Results of the ablation experiments on the 2-hops dataset. (-f), (-w), and (-wf) indicate models without finetuning, weighting and without both weighting and finetuning, respectively.

Test set	Model	Hit@k (%) 1	2	5	10	20
2-hops	SGCN(-f)	24.8	38.3	57.5	69.9	79.6
	DGP(-wf)	23.8	36.9	56.2	69.1	78.6
	DGP(-f)	24.6	37.8	56.9	69.6	79.3
	DGP(-w)	25.4	39.5	59.9	72.0	80.9
	SGCN (ours)	26.2	40.4	60.2	71.9	81.0
	DGP (ours)	**26.6**	**40.7**	**60.3**	**72.3**	**81.3**

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 4 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 both our methods outperform GCNZ.

4.4. Model analysis

Analysis of weighting scheme. To validate our intuition that weighting allows our approach to weigh distant neighbors less, we inspect the learned weights. For the first stage the weights are 0.244, 0.476, 0.162, 0.060, 0.058 and for the second (final) stage they are 0.493, 0.322, 0.097, 0.047, 0.041. Note, the first value corresponds to self-weighting, the second to the 1-hop neighbors, and so forth. It can be

Table 4: Performance on the seen ImageNet classes. ResNet represents ideal performance as it only predicts known classes. GCNZ is our reimplementation of [34].

Model	Hit@k (%)			
	1	2	5	10
ResNet	75.1	85.5	92.7	95.7
GCNZ	38.3	62.9	82.3	89.8
SGCN (ours)	49.1	68.7	**83.9**	**89.4**
DGP (ours)	**54.6**	**69.7**	83.8	89.1

Table 5: Results for 2-hops for SGCN without finetuning when increasing the depth.

#Layers	Hit@k (%)				
	1	2	5	10	20
1	**24.8**	**38.3**	**57.5**	**69.9**	**79.6**
2	24.2	37.7	57.4	69.2	78.1
3	23.9	37.5	57.1	68.4	77.2

Table 6: Results for 2-hops with/without separating the adjacency matrix into ancestors and descendants for DGP.

Model	Hit@k (%)				
	1	2	5	10	20
without	26.0	40.2	59.8	71.4	80.3
with	**26.6**	**40.7**	**60.3**	**72.3**	**81.3**

observed, that ancestors aggregate information mainly from their immediate descendants in the first phase and later distribute it to their descendants in the second phase. Further, we observe that distant neighbors have far less impact in the final stage. This means that the model learns to preserve the overall graph structure imposed by the knowledge graph, where importance is governed by the distance in the graph.

Analysis of number of layers. We perform an empirical evaluation to verify that our intuition is correct and that additional hidden layers indeed cause a drop in performance when employing a GCN. Table 5 illustrates the performance when adding additional layers to the GCN for the 2-hops experiment. These results are reported without finetuning the model. In order to perform this ablation study we fix all hidden layers to have a dimensionality of 2048 with 0.5 dropout. We want to stress that there is a fundamental difference in our experimental setting and the study in Wang *et al*. [34], as their ablation study does not only consider a different number of layers in the network but also a different number of neurons per layer at the same time.

Analysis of two-phase propagation. We further, perform an ablation study to analyze the benefit of a two-phase directed propagation rule where ancestors and descendants are considered individually. We compared this to two consecutive updates using the full adjacency matrix in the dense method and illustrate the results in Table 6. Consistent improvements are obtained using our proposed two-phase directed propagation rule.

Table 7: Mean and standard deviation for 3 runs. More stable as the number of class increases.

Test set	Model	Hit@k (%)	
		1	2
2-hops	SGCN	26.17±0.03	40.41±0.03
	DGP	26.67±0.09	40.74±0.04
All	SGCN	2.80±0.01	4.90±0.01
	DGP	2.95±0.00	5.05±0.02

Robustness of results. Table 7 shows the mean and standard deviation for 3 runs for the 2-hops and All datasets. The results are stable over multiple runs and it can clearly be observed that as the number of classes increases (2-hops to all), results become more stable.

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 exploits the structure of knowledge graphs, where entities only have few ancestors and descendants, 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}.

With regards to the number of parameters, the SGCN consists of 4,810,752 weights. DGP increases the number of trainable parameters by adding $2 \times (K + 1)$ additional weights. However, as $K = 4$ in our experiments, this difference in the number of parameters is negligible. Overall the number of trainable parameters is considerably lower than that in the GCNZ model (9,527,808 weights).

5. Conclusion

In contrast to previous approaches using graph convolutional neural networks for zero-shot learning, we illustrate that the task of zero-shot learning benefits from shallow networks. Further, to avoid the lack of information propagation between distant nodes in shallow models, we propose DGP, which exploits the hierarchical structure of the knowledge graph by adding a weighted dense connection scheme. 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 weighting mechanisms to further improve the performance of DGP compared to the SGCN. The inclusion of additional semantic information for settings where these are available for a subset of nodes is another future direction.

Acknowledgments: This work was partially funded by the Norwegian Research Council FRIPRO grant no. 239844.

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. 2, 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, 11

[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. 5, 6

[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. 7

[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. 2, 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, 11

[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, 2, 3

[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. 3

[16] V. Kumar Verma, G. Arora, A. Mishra, and P. Rai. Generalized zero-shot learning via synthesized examples. In *CVPR*, 2018. 11

[17] 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

[18] 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. 1, 4

[19] 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

[20] 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

[21] Y. Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3432–3438. AAAI Press, 2016. 1

[22] 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

[23] 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

[24] 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. 5, 6, 7, 11

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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, 7

[32] 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

[33] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *International Conference for Learning Representations*, 2018. 5, 11

[34] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 3, 5, 6, 7, 8, 11

[35] 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. 11

[36] H. Zhang and P. Koniusz. Zero-shot kernel learning. In *CVPR*, 2018. 11

6. Supplementary Materials

6.1. Additional Qualitative Examples

Figure 4 and 5 provide further qualitative results of our single-hidden-layer GCN (SGCN) and Dense Graph Propagation Module (DGP) compared to a standard ResNet and GCNZ, our reimplementation of [34].

6.2. Performance improvements between GCNZ and SGCN

Table 8 explains the performance difference between our SGCN, our reimplementation of GCNZ and the reported results in [34]. Note, unless otherwise stated training is performed for 3000 epochs. Non-symmetric normalization ($D^{-1}A$) is denoted as *non-sym* in the normalization column, while a symmetric normalization ($D^{-1/2}AD^{-1/2}$) is denoted as *sym*. No finetuning has been performed for SGCN in these results.

Table 8: Illustration of the improvements between the original results of GCNZ in [34], our reimplementation of GCNZ and our SGCN. GCNZ† corresponds to updated results from [34] (taken from https://github.com/JudyYe/zero-shot-gcn). GCNZ‡ is our reimplementation of [34].

Model	Norm	**Hit@k (%)**				
		1	2	5	10	20
GCNZ (300 epochs) [34]	sym	19.8	33.3	53.2	65.4	74.6
GCNZ† (300 epochs) [34]	sym	21.0	33.7	52.7	64.8	74.3
GCNZ‡ (300 epochs)	sym	21.4	34.7	54.3	67.5	77.6
GCNZ‡	sym	23.5	36.9	56.5	68.8	78.0
SGCN (ours)	sym	24.6	38.1	57.6	70.0	79.7
SGCN (ours)	non-sym	24.8	38.3	57.5	69.9	79.6

6.3. Performance on AWA2

AWA2 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 [35] 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.

Results for the AWA2 dataset are presented in Table 9. Note that our model differs 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 [34], 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 [34], 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. Note, all methods, except SYNC, use a pretrained network trained on the 1K ImageNet classes. GCNZ and our DGP do not make use of the attribute information supplied for AWA2, however, both methods use the ImageNet knowledge graph.

Table 9: Top-1 accuracy results for unseen classes on AWA2. Results for ConSE, Devise and SYNC obtained from [35].

Model	**ACC (%)**
ConSE [24]	44.5
Devise [11]	59.7
SYNC [4]	46.6
SE-GZSL [16]	69.2
Gaussian-Ort [36]	70.5
GCNZ [34]	70.7
DGP (ours)	**77.3**

6.4. Comparison to Graph Attention Networks

Table 10 illustrates the results for a 1-hidden-layer and 2-hidden-layer GCN with the attention mechanism proposed in GAT [33]. Note, performance degrades compared to a 1-hidden-layer GCN (i.e. SGCN(-f)). The hidden dimension is 2048 and training settings are the same as in the paper.

Table 10: Accuracy on ImageNet for a 1- and 2-hidden-layer GAT [33] compared to a 1-hidden-layer GCN without attention.

Test set	**Model**	**Hit@k (%)**				
		1	2	5	10	20
2-hops	GAT-1	24.1	37.5	57.2	69.7	79.4
	GAT-2	23.3	36.9	56.8	68.7	77.9
	GCN-1 (ours)	**24.8**	**38.3**	**57.5**	**69.9**	**79.6**

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

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

ResNet: lemon, orange, banana, spaghetti squash, fig
GCNZ: **bitter orange**, temple orange, citrus, sweet orange, edible fruit
SGCN: citrus, **bitter orange**, temple orange, sweet orange, edible fruit,
DGP: 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
SGCN: **blue**, hairstreak, copper, pierid, butterfly,
DGP: **blue**, hairstreak, copper, pierid, butterfly

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

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

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

ResNet: desktop computer, monitor, screen, computer keyboard, mouse
GCNZ: personal computer, portable computer, planner, computer, computer screen
SGCN: personal computer, computer, computer screen, **display**, television monitor
DGP: 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
SGCN: american bittern, **european bittern**, least bittern, plain turkey, great bustard
DGP: american bittern, least bittern, **european bittern**, heron, egret

ResNet: damselfly, dragonfly, lacewing, walking stick, grasshopper
GCNZ: **odonate**, neuropteron, hymenopterous insect, phasmid, brown lacewing
SGCN: **odonate**, neuropteron, brown lacewing, green lacewing, phasmid
DGP: **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
SGCN: **parrot**, lory, rainbow lorikeet, varied lorikeet, cockatoo
DGP: lory, **parrot**, cockatoo, rainbow lorikeet, varied lorikeet

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

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

ResNet: church, monastery, dome, bell cote, mosque
GCNZ: kirk, cathedral, abbey, basilica, cathedral
SGCN: abbey, cathedral, friary, basilica, cathedral
DGP: cathedral, abbey, cathedral, basilica, kirk

true label: **place_of_worship**