Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs

Xiaolong Wang* Yufei Ye* Abhinav Gupta
The Robotics Institute, Carnegie Mellon University

Abstract

We consider the problem of zero-shot recognition: learning a visual classifier for a category with zero training examples, just using the word embedding of the category and its relationship to other categories, which visual data are provided. The key to dealing with the unfamiliar or novel category is to transfer knowledge obtained from familiar classes to describe the unfamiliar class. In this paper, we build upon the recently introduced Graph Convolutional Network (GCN) and propose an approach that uses both semantic embeddings and the categorical relationships to predict the classifiers. Given a learned knowledge graph (KG), our approach takes as input semantic embeddings for each node (representing visual category). After a series of graph convolutions, we predict the visual classifier for each category. During training, the visual classifiers for a few categories are given to learn the GCN parameters. At test time, these filters are used to predict the visual classifiers of unseen categories. We show that our approach is robust to noise in the KG. More importantly, our approach provides significant improvement in performance compared to the current state-of-the-art results (from $2 \sim 3\%$ on some metrics to whopping 20% on a few).

1. Introduction

Consider the animal category "okapi". Even though we might have never heard of this category or seen visual examples in the past, we can still learn a good visual classifier based on the following description: "zebra-striped four legged animal with a brown torso and a deer-like face" (Test yourself on figure 1). On the other hand, our current recognition algorithms still operate in closed world conditions: that is, they can only recognize the categories they are trained with. Adding a new category requires collecting thousands of training examples and then retraining the classifiers. To tackle this problem, zero-shot learning is often used.

The key to dealing with the unfamiliar or novel category is to transfer knowledge obtained from familiar classes to describe the unfamiliar classes (generalization). There are two paradigms of transferring knowledge. The first paradigm is to use implicit knowledge representations, i.e. semantic embeddings. In this approach, one learns a vector representation of different categories using text data and then learns a mapping between the vector representation to visual classifier directly [34, 13]. However, these methods are limited by the generalization power of the semantic models and the mapping models themselves. It is also hard to learn semantic embeddings from structured information.

*Indicates equal contribution.

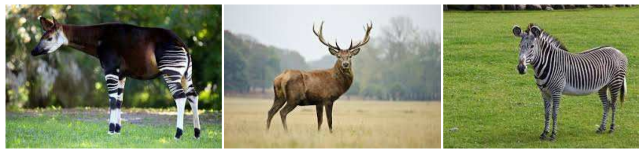
Figure 1. Can you find "okapi" in these images? Okapi is " zebra-striped four legged animal with a brown torso and a deer-like face". In this paper, we focus on the problem of zero-shot learning where visual classifiers are learned from semantic embeddings and relationships to other categories.

The alternative and less-explored paradigm for zero-shot learning is to use explicit knowledge bases or knowledge graphs. In this paradigm, one explicitly represents the knowledge as rules or relationships between objects. These relationships can then be used to learn zero-shot classifiers for new categories. The simplest example would be to learn visual classifiers of compositional categories. Given classifiers of primitive visual concepts as inputs, [33] applies a simple composition rule to generate classifiers for new complex concepts. However, in the general form, the relationships can be more complex than simple compositionality. An interesting question we want to explore is if we can use structured information and complex relationships to learn visual classifiers without seeing any examples.

In this paper, we propose to distill both the implicit knowledge representations (i.e. word embedding) and explicit relationships (i.e. knowledge graph) for learning visual classifiers of novel classes. We build a knowledge graph where each node corresponds to a semantic category. These nodes are linked via relationship edges. The input to each node of the graph is the vector representation (semantic embedding) of each category. We then use Graph Convolutional Network

(GCN) [22] to transfer information (message-passing) between different categories. Specifically, we train a 6-layer deep GCN that outputs the classifiers of different categories.

We focus on the task of image classification. We consider both of the test settings: (a) final test classes being only zero-shot classes (without training classes at test time); (b) at test time the labels can be either the seen or the unseen classes, namely "generalized zero-shot setting" [16, 6, 50]. We show surprisingly powerful results and huge improvements over classical baselines such as DeVise [13] , ConSE [34] ,and current state-of-the-art [5]. For example, on standard ImageNet with 2-hop setting, 43.7% of the images retrieved by [5] in top-10 are correct. Our approach retrieves 62.4% images correctly. **That is a whopping 18.7% improvement over the current state-of-the-art.** More interestingly, we show that our approach scales amazingly well and giving a significant improvement as we increase the size of the knowledge graph even if the graph is noisy.

2. Related Work

With recent success of large-scale recognition systems [45], the focus has now shifted to scaling these systems in terms of categories. As more realistic and practical settings are considered, the need for zero-shot recognition – training visual classifiers without any examples – has increased. Specifically, the problem of mapping text to visual classifiers is very interesting.

Early work on zero-shot learning used attributes [11, 24, 19] to represent categories as vector indicating presence/absence of attributes. This vector representation can then be mapped to learn visual classifiers. Instead of using manually defined attribute-class relationships, Rohrbach et al. [40, 38] mined these associations from different internet sources. Akata et al. [1] used attributes as side-information to learn a semantic embedding which helps in zero-shot recognition. Recently, there have been approaches such as [37] which trys to match Wikipedia text to images by modeling noise in the text description.

With the advancement of deep learning, most recent approaches can be mapped into two main research directions. The first approach is to use semantic embeddings (implicit representations). The core idea is to represent each category with learned vector representations that can be mapped to visual classifiers [48, 44, 13, 41, 25, 15, 14, 18, 23, 4, 5, 55, 54]. Socher et al. [44] proposed training two different neural networks for image and language in an unsupervised manner, and then learning a linear mapping between image representations and word embeddings. Motivated by this work, Frome et al. [13] proposed a system called DeViSE to train a mapping from image to word embeddings using a ConvNet and a transformation layer. By using the predicted embedding to perform nearest neighbor search, DeViSE scales up the zero-shot recognition to thousands of classes. Instead of training a ConvNet to predict the word embedding directly, Norouzi et al. [34] proposed another system named ConSE which constructs the image embedding by combining an existing image classification ConvNet and word embedding model. Recently, Changpinyo et al [4] proposed an approach to align semantic and visual manifolds via use of 'phantom' classes. They report state-of-the-art results on ImageNet dataset using this approach. One strong shortcoming of these approaches is they do not use any explicit relationships between classes but rather use semantic-embeddings to represent relationships.

The second popular way to distill the knowledge is to use knowledge graph (explicit knowledge representations). Researchers have proposed several approaches on how to use knowledge graphs for object recognition [12, 43, 30, 35, 39, 9, 8, 29, 49, 47, 26]. For example, Salakhutdinov et al. [43] used WordNet to share the representations among different object classifiers so that objects with few training examples can borrow statistical strength from related objects. On the other hand, the knowledge graph can also be used to model the mutual exclusion among different classes. Deng et al. [9] applied these exclusion rules as a constraint in the loss for training object classifiers (e.g. an object will not be a dog and a cat at the same time). They have also shown zero-shot applications by adding object-attribute relations into the graph. In contrast to these methods of using graph as constraints, our approach used the graph to directly generate novel object classifiers [33, 10, 2].

In our work, we propose to distill information both via semantic embeddings and knowledge graphs. Specifically, given a word embedding of an unseen category and the knowledge graph that encodes explicit relationships, our approach predicts the visual classifiers of unseen categories. To model the knowledge graph, our work builds upon the Graph Convolutional Networks [22]. It was originally proposed for semi-supervised learning in language processing. We extend it to our zero-short learning problem by changing the model architecture and training loss.

3. Approach

Our goal is to distill information from both implicit (word-embeddings) and explicit (knowledge-graph) representations for zero-shot recognition. But what is the right way to extract information? We build upon the recent work on Graph Convolutional Network (GCN) [22] to learn visual classifiers. In the following, we will first introduce how the GCN is applied in natural language processing for classification tasks, and then we will go into details about our approach: applying the GCN with a regression loss for zero-shot learning.

3.1. Preliminaries: Graph Convolutional Network

Graph Convolutional Network (GCN) was introduced in [22] to perform semi-supervised entity classification.

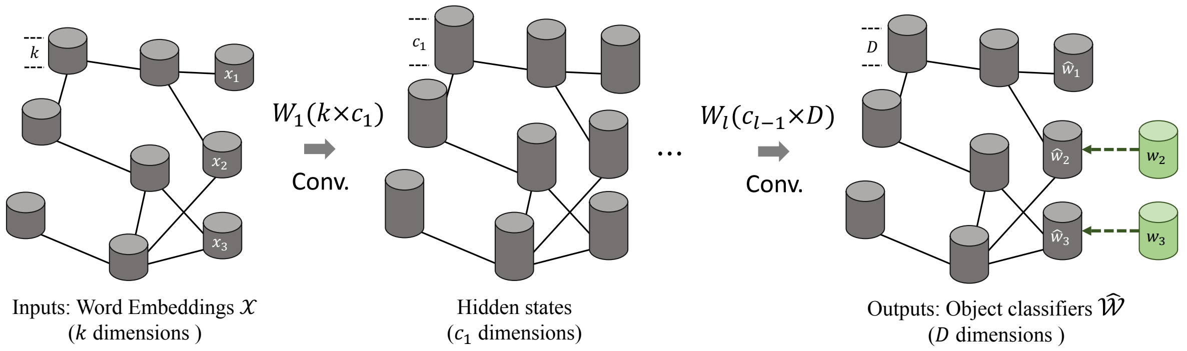

Figure 2. An example of our Graph Convolutional Network. It takes word embeddings as inputs and outputs the object classifiers. The supervision comes from the ground-truth classifiers w_2 and w_3 highlighted by green. During testing, we input the same word embeddings and obtain classifier for x_1 as $\hat{w}_1$. This classifier will be multiplied with the image features to produce classification scores.

Given object entities, represented by word embeddings or text features, the task is to perform classification. For example, entities such as "dog" and "cat" will be labeled as "mammal"; "chair" and "couch" will be labeled "furniture". We also assume that there is a graph where nodes are entities and the edges represent relationships between entities.

Formally, given a dataset with n entities $(X, Y) = \{(x_i, y_i)\}_{i=1}^n$ where x_i represents the word embedding for entity i and $y_i \in \{1, ..., C\}$ represents its label. In semi-supervised setting, we know the ground-truth labels for the first m entities. Our goal is to infer y_i for the remaining $n - m$ entities, which do not have labels, using the word embedding and the relationship graph. In the relationship graph, each node is an entity and two nodes are linked if they have a relationship in between.

We use a function $F(\cdot)$ to represent the Graph Convolutional Network. It takes all the entity word embeddings X as inputs at one time and outputs the SoftMax classification results for all of them as $F(X)$. For simplicity, we denote the output for the ith entity as $F_i(X)$, which is a C dimension SoftMax probability vector. In training time, we apply the SoftMax loss on the first m entities, which have labels as

$$\frac{1}{m} \sum_{i=1}^{m} L_{\text{softmax}}(F_i(X), y_i). \tag{1}$$

The weights of $F(\cdot)$ are trained via back-propagation with this loss. During testing time, we use the learned weights to obtain the labels for the $n - m$ entities with $F_i(X), i \in \{m + 1, ..., n\}$.

Unlike standard convolutions that operate on local region in an image, in GCN the convolutional operations compute the response at a node based on the neighboring nodes defined by the adjacency graph. Mathematically, the convolutional operations for each layer in the network $F(\cdot)$ is represented as

$$Z = \hat{A} X' W \tag{2}$$

where $\hat{A}$ is a normalized version of the binary adjacency matrix A of the graph, with $n \times n$ dimensions. X' is the input $n \times k$ feature matrix from the former layer. W is the weight matrix of the layer with dimension $k \times c$, where c is the output channel number. Therefore, the input to a convolutional layer is $n \times k$,and the output is a $n \times c$ matrix Z. These convolution operations can be stacked one after another. A non-linear operation (ReLU) is also applied after each convolutional layer before the features are forwarded to the next layer. For the final convolutional layer, the number of output channels is the number of label classes ($c = C$). For more details, please refer to [22].

3.2. GCN for Zero-shot Learning

Our model builds upon the Graph Convolutional Network. However, instead of entity classification, we apply it to the zero-shot recognition with a regression loss. The input of our framework is the set of categories and their corresponding semantic-embedding vectors (represented by $\mathcal{X} = \{x_i\}_{i=1}^n$). For the output, we want to predict the visual classifier for each input category (represented by $\mathcal{W} = \{w_i\}_{i=1}^n$).

Specifically, the visual classifier we want the GCN to predict is a logistic regression model on the fixed pre-trained ConvNet features. If the dimensionality of visual-feature vector is D, each classifier w_i for category i is also a D-dimensional vector. Thus the output of each node in the GCN is D dimensions, instead of C dimensions. In the zero-shot setting, we assume that the first m categories in the total n classes have enough visual examples to estimate their weight vectors. For the remaining $n - m$ categories, we want to estimate their corresponding weight vectors given their embedding vectors as inputs.

One way is to train a neural network (multi-layer perceptron) which takes x_i as an input and learns to predict w_i as an output. The parameters of the network can be estimated using m training pairs. However, generally m is small (in the order of a few hundreds) and therefore, we want to use the explicit structure of the visual world or the relationships between categories to constrain the problem. We represent these relationships as the knowledge-graph (KG). Each node in the KG represents a semantic category. Since we have a total of n categories, there are n nodes in the graph. Two nodes are linked to each other if there is a relationship between them. The graph structure is represented by the $n \times n$ adjacency matrix, A. Instead of building a bipartite graph as [22, 52], we replace all directed edges in the KG by undirected edges, which leads to a symmetric adjacency matrix.

As Fig. 2 shows, we use a 6-layer GCN where each layer l takes as input the feature representation from previous layer (Z_{l-1}) and outputs a new feature representation (Z_l). For the first layer the input is $\mathcal{X}$ which is an $n \times k$ matrix (k is the dimensionality of the word-embedding vector). For the final-layer the output feature-vector is $\hat{\mathcal{W}}$ which has the size of $n \times D$; D being the dimensionality of the classifier or visual feature vector.

Loss-function: For the first m categories, we have predicted classifier weights $\hat{\mathcal{W}}_{1...m}$ and ground-truth classifier weights learned from training images $\mathcal{W}_{1...m}$. We use the mean-square error as the loss function between the predicted and the ground truth classifiers.

$$\frac{1}{m}\sum_{i=1}^{m} L_{\text{mse}}(\hat{w}_i, w_i). \tag{3}$$

During training, we use the loss from the m seen categories to estimate the parameters for the GCN. Using the estimated parameters, we obtain the classifier weights for the zero-shot categories. At test time, we first extract the image feature representations via the pre-trained ConvNet and use these generated classifiers to perform classification on the extracted features.

3.3. Implementation Details

Our GCN is composed of 6 convolutional layers with output channel numbers as $2048 \rightarrow 2048 \rightarrow 1024 \rightarrow 1024 \rightarrow 512 \rightarrow D$, where D represents the dimension of the object classifier. Unlike the 2-layer network presented in [22], our network is much deeper. As shown in ablative studies, we find that making the network deep is essential in generating the classifier weights. For activation functions, instead of using ReLU after each convolutional layer, we apply LeakyReLU [27, 51] with the negative slope of 0.2. Empirically, we find that LeakyReLU leads to faster convergence for our regression problem.

While training our GCN, we perform L2-Normalization on the outputs of the networks and the ground-truth classifiers. During testing, the generated classifiers of unseen classes are also L2-Normalized. We find adding this constraint important, as it regularizes the weights of all the classifiers into similar magnitudes. In practice, we also find that the last layer classifiers of the ImageNet pre-trained networks are naturally normalized. That is, if we perform L2-Normalization on each of the last layer classifiers during testing, the performance on the ImageNet 2012 1K-class validation set changes marginally ($< 1\%$).

To obtain the word embeddings for GCN inputs, we use the GloVe text model [36] trained on the Wikipedia dataset, which leads to 300-d vectors. For the classes whose names contain multiple words, we match all the words in the trained model and find their embeddings. By averaging these word embeddings, we obtain the class embedding.

4. Experiment

We now perform experiments to showcase that our approach: (a) improves the state-of-the-art by a significant margin; (b) is robust to different pre-trained ConvNets and noise in the KG. We use two datasets in our experiments. The first dataset we use is constructed from publicly-available knowledge bases. The dataset consists of relationships and graph from Never-Ending Language Learning (NELL) [3] and images from Never-Ending Image Learning (NEIL) [8]. This is an ideal dataset for: (a) demonstrating that our approach is robust even with automatically learned (and noisy) KG; (b) ablative studies since the KG in this domain is rich, and we can perform ablations on KG as well.

Our final experiments are shown on the standard ImageNet dataset. We use the same settings as the baseline approaches [13, 34, 4] together with the WordNet [32] knowledge graph. We show that our approach surpasses the state-of-the-art methods by a significant margin.

4.1. Experiments on NELL and NEIL

Dataset settings. For this experiment, we construct a new knowledge graph based on the NELL [3] and NEIL [8] datasets. Specifically, the object nodes in NEIL correspond to the nodes in NELL. The NEIL dataset offers the sources of images and the NELL dataset offers the common sense knowledge rules. However, the NELL graph is incredibly large [1]: it contains roughly 1.7M types of object entities and around 2.4M edges representing the relationships between every two objects. Furthermore, since NELL is constructed automatically, there are noisy edges in the graph. Therefore, we create sub-graphs for our experiments.

The process of constructing this sub-graph is straightforward. We perform Breadth-first search (BFS) starting from the NEIL nodes. We discover paths with maximum length K hops such that the first and last node in the path are NEIL nodes. We add all the nodes and edges in these paths into our sub-graph. We set $K = 7$ during BFS because we discover a path longer than 7 hops will cause the connection between two objects noisy and unreasonable. For example, “jeep” can be connected to “deer” in a long path but they are hardly semantically related.

Note that each edge in NELL has a confidence value that is usually larger than 0.9. For our experiments, we create two

[1] http://rtw.ml.cmu.edu/

Dataset	All Nodes	NEIL Nodes (Train/Test)	Edges
High Value Edges	8819	431/88	40810
All Edges	14612	616/88	96772

Table 1. Dataset Statistics: Two different sizes of knowledge graphs in our experiment.

different versions of sub-graphs. The first smaller version is a graph with high value edges (larger than 0.999), and the second one used all the edges regardless of their confidence values. The statistics of the two sub-graphs are summarized in Table 1. For the larger sub-graph, we have 14K object nodes. Among these nodes, 704 of them have corresponding images in the NEIL database. We use 616 classes for training our GCN and leave 88 classes for testing. Note that these 88 testing classes are randomly selected among the classes that have no overlap with the 1000 classes in the standard ImageNet classification dataset. The smaller knowledge graph is around half the size of the larger one. We use the same 88 testing classes in both settings

Training details. For training the ConvNet on NEIL images, we use the 310K images associated with the 616 training classes. The evaluation is performed on the randomly selected 12K images associated with the 88 testing classes, i.e. all images from the training classes are excluded during testing. We fine-tune the ImageNet pre-trained VGGM [7] network architecture with relatively small $fc7$ outputs (128-dimension). Thus the object classifier dimension in $fc8$ is 128. For training our GCN, we use the ADAM [21] optimizer with learning rate 0.001 and weight decay 0.0005. We train our GCN for 300 epochs for every experiment.

Baseline method. We compare our method with one of the state-of-the-art methods, ConSE [34], which shows slightly better performance than DeViSE [13] in ImageNet. As a brief introduction, ConSE first feedforwards the test image into a ConvNet that is trained only on the training classes. With the output probabilities, ConSE selects top T predictions $\{p_i\}_{i=1}^T$ and the word embeddings $\{x_i\}_{i=1}^T$ [31] of these classes. It then generates a new word embedding by weighted averaging the T embeddings with the probability $\frac{1}{T}\sum_{i=1}^T p_i x_i$. This new embedding is applied to perform nearest neighbors in the word embeddings of the testing classes. The top retrieved classes are selected as the final result. We enumerate different values of T for evaluations.

Quantitative Results. We perform evaluations on the task of 88 unseen categories classification. Our metric is based on the percentage of correctly retrieved test data (out of top k retrievals) for a given zero-shot class. The results are shown in Table 2. We evaluate our method on two different sizes of knowledge graphs. We use "High Value Edges" to denote the knowledge graph constructed based on high confidence edges. "All Edges" represents the graph constructed with all the edges. We denote the baseline [34] as "ConSE(T)" where we set T to be 5, 10 and the number of training classes.

Test Set	Model	Hit@k (%) 1	2	5	10
High Value Edges	ConSE(5)	6.6	9.6	13.6	19.4
	ConSE(10)	7.0	9.8	14.2	20.1
	ConSE(431)	6.7	9.7	14.9	20.5
	Ours	**9.1**	**16.8**	**23.2**	**47.9**
All Edges	ConSE(5)	7.7	10.1	13.9	19.5
	ConSE(10)	7.7	10.4	14.7	20.5
	ConSE(616)	7.7	10.5	15.7	21.4
	Ours	**10.8**	**18.4**	**33.7**	**49.0**

Table 2. Top-k accuracy for different models in different settings.

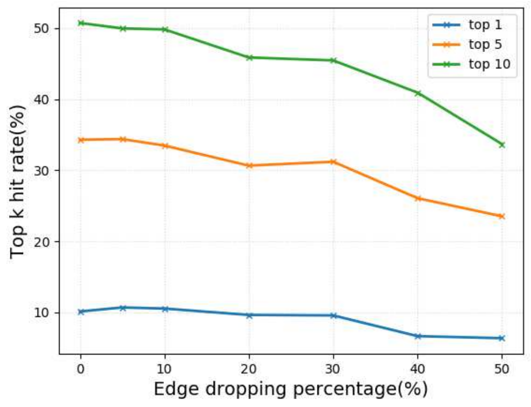

Figure 3. We randomly drop 5% to 50% of the edges in the "All Edges" graph and show the top-1, top-5 and top-10 accuracies.

Our method outperforms the ConSE baseline by a large margin. In the "All Edges" dataset, our method outperforms ConSE 3.6% in top-1 accuracy. **More impressively, the accuracy of our method is almost 2 times as that of ConSE in top-2 metric and even more than 2 times in top-5 and top-10 accuracies**. These results show that using knowledge graph with word embeddings in our method leads to much better result than the state-of-the-art results with word embeddings only.

From small to larger graph. In addition to improving performance in zero-shot recognition, our method obtains more performance gain as our graph size increases. As shown in Table 2, our method performs better by switching from the small to larger graph. Our approach has obtained $2 \sim 3\%$ improvements in all the metrics. On the other hand, there is little to no improvements in ConSE performance. It also shows that the KG does not need to be hand-crafted or cleaned. Our approach is able to robustly handle the errors in the graph structure.

Resilience to Missing Edges We explore how the performance of our model changes if we randomly drop 5% to 50% of the edges in the "All Edges" graph. As Fig. 3 shows, by dropping from 5% to 10% of edges, the performance of our model changes negligibly. This is mainly because the

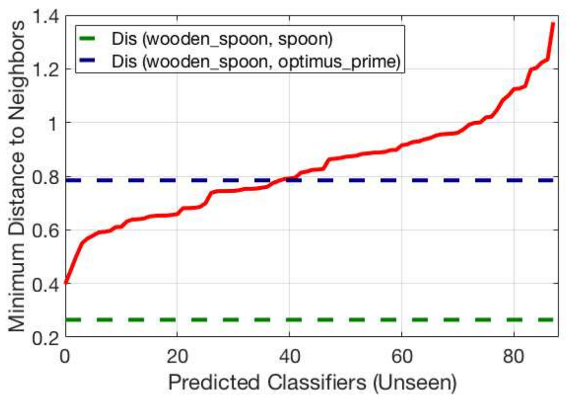

Figure 4. We compute the minimum Euclidean distances between predicted and training classifiers. The distances are plotted by sorting them from small to large.

knowledge graph can have redundant information with 14K nodes and 97K edges connecting them. This again implies that our model is robust to small noisy changes in the graph. As we start deleting more than 30% of the edges, the accuracies drop drastically. This indicates that the performance of our model is highly correlated to the size of the knowledge graph.

Random Graph? It is clear that our approach can handle noise in the graph. But does any random graph work? To demonstrate that the structure of the graph is still critical we also created some trivial graphs: (i) star model: we create a graph with one single root node and only have edges connecting object nodes to the root node; (ii) random graph: all nodes in the graph are randomly connected. Table 3 shows the results. It is clear that all the numbers are close to random guessing, which means a reasonable graph plays an important role and a random graph can have negative effects on the model.

Test Set	Trivial KG	Hit@k (%)			
		1	2	5	10
All Edges	Star Model	1.1	1.6	4.8	9.7
	Random Graph	1.0	2.2	5.6	11.3

Table 3. Top-k accuracy on trivial knowledge graphs we create.

How important is the depth of GCN? We show that making the Graph Convolutional Network deep is critical in our problem. We show the performance of using different numbers of layers for our model on the "All Edges" knowledge graph shown in Table 4. For the 2-layer model we use 512 hidden neurons, and the 4-layer model has output channel numbers as $2048 \rightarrow 1024 \rightarrow 512 \rightarrow 128$. We show that the performance keeps increasing as we make the model deeper from 2-layer to 6-layer. The reason is that increasing the times of convolutions is essentially increasing the times of message passing between nodes in the graph. However, we do not observe much gain by adding more layers above

Test Set	Model	Hit@k (%)			
		1	2	5	10
All Edges	Ours (2-layer)	5.3	8.7	15.5	24.3
	Ours (4-layer)	8.2	13.5	27.1	41.8
	Ours (6-layer)	**10.8**	**18.4**	**33.7**	**49.0**

Table 4. Top-k accuracy with different depths of our model.

the 6-layer model. One potential reason might be that the optimization becomes harder as the network goes deeper.

Is our network just copying classifiers as outputs? Even though we show our method is better than ConSE baseline, is it possible that it learns to selectively copy the nearby classifiers? To show our method is not learning this trivial solution, we compute the Euclidean distance between our generated classifiers and the training classifiers. More specifically, for a generated classifier, we compare it with the classifiers from the training classes that are at most 3-hops away. We calculate the minimum distance between each generated classifier and its neighbors. We sort the distances for all 88 classifiers and plot Fig. 4. As for reference, the distance between "wooden_spoon" and "spoon" classifiers in the training set is 0.26 and the distance between "wooden_spoon" and "optimus_prime" is 0.78. We can see that our predicted classifiers

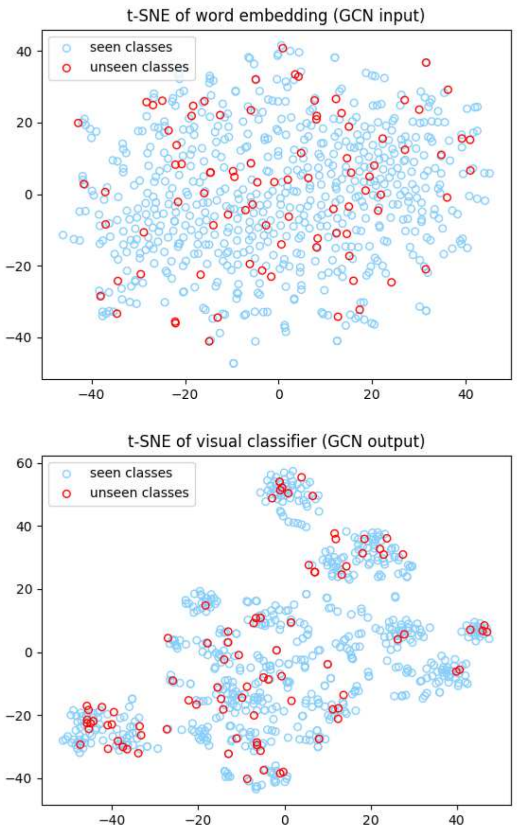

Figure 5. t-SNE visualizations for our word embeddings and GCN output visual classifiers in the "All Edges" dataset. The test classes are shown in red.

Test Set	Model	ConvNets	Hit@k (%)				
			1	2	5	10	20
2-hops	ConSE [4]	Inception-v1	8.3	12.9	21.8	30.9	41.7
	ConSE(us)	Inception-v1	12.4	18.4	25.3	28.5	31.8
	SYNC [4]	Inception-v1	10.5	17.7	28.6	40.1	52.0
	EXEM [5]	Inception-v1	12.5	19.5	32.3	43.7	55.2
	Ours	Inception-v1	18.5	31.3	50.1	62.4	72.0
	Ours	ResNet-50	**19.8**	**33.3**	**53.2**	**65.4**	**74.6**
3-hops	ConSE [4]	Inception-v1	2.6	4.1	7.3	11.1	16.4
	ConSE(us)	Inception-v1	3.2	4.9	7.6	9.7	11.4
	SYNC [4]	Inception-v1	2.9	4.9	9.2	14.2	20.9
	EXEM [5]	Inception-v1	3.6	5.9	10.7	16.1	23.1
	Ours	Inception-v1	3.8	6.9	13.1	18.8	26.0
	Ours	ResNet-50	**4.1**	**7.5**	**14.2**	**20.2**	**27.7**
All	ConSE [4]	Inception-v1	1.3	2.1	3.8	5.8	8.7
	ConSE(us)	Inception-v1	1.5	2.2	3.6	4.6	5.7
	SYNC [4]	Inception-v1	1.4	2.4	4.5	7.1	10.9
	EXEM [5]	Inception-v1	**1.8**	2.9	5.3	8.2	12.2
	Ours	Inception-v1	1.7	3.0	5.8	8.4	11.8
	Ours	ResNet-50	**1.8**	**3.3**	**6.3**	**9.1**	**12.7**

(a) Top-k accuracy for different models when testing on only unseen classes.

Test Set	Model	ConvNets	Hit@k (%)				
			1	2	5	10	20
2-hops (+1K)	DeViSE [13]	AlexNet	0.8	2.7	7.9	14.2	22.7
	ConSE [34]	AlexNet	0.3	6.2	17.0	24.9	33.5
	ConSE(us)	Inception-v1	0.2	7.8	18.1	22.8	26.4
	ConSE(us)	ResNet-50	0.1	11.2	24.3	29.1	32.7
	Ours	Inception-v1	7.9	18.6	39.4	53.8	65.3
	Ours	ResNet-50	**9.7**	**20.4**	**42.6**	**57.0**	**68.2**
3-hops (+1K)	DeViSE [13]	AlexNet	0.5	1.4	3.4	5.9	9.7
	ConSE [34]	AlexNet	0.2	2.2	5.9	9.7	14.3
	ConSE(us)	Inception-v1	0.2	2.8	6.5	8.9	10.9
	ConSE(us)	ResNet-50	0.2	3.2	7.3	10.0	12.2
	Ours	Inception-v1	1.9	4.6	10.9	16.7	24.0
	Ours	ResNet-50	**2.2**	**5.1**	**11.9**	**18.0**	**25.6**
All (+1K)	DeViSE [13]	AlexNet	0.3	0.8	1.9	3.2	5.3
	ConSE [34]	AlexNet	0.2	1.2	3.0	5.0	7.5
	ConSE(us)	Inception-v1	0.1	1.3	3.1	4.3	5.5
	ConSE(us)	ResNet-50	0.1	1.5	3.5	4.9	6.2
	Ours	Inception-v1	0.9	2.0	4.8	7.5	10.8
	Ours	ResNet-50	**1.0**	**2.3**	**5.3**	**8.1**	**11.7**

(b) Top-k accuracy for different models when testing on both seen and unseen classes (a more practical and generalized setting).

Table 5. Results on ImageNet. We test our model on 2 different settings over 3 different datasets.

are quite different from its neighbors.

Are the outputs only relying on the word embeddings? We perform t-SNE [28] visualizations to show that our output classifiers are not just derived from the word embeddings. We show the t-SNE [28] plots of both the word embeddings and the classifiers of the seen and unseen classes in the "All Edges" dataset. As Fig. 5 shows, we have very different clustering results between the word embeddings and the object classifiers, which indicates that our GCN is not just learning a direct projection from word embeddings to classifiers.

4.2. Experiments on WordNet and ImageNet

We now perform our experiments on a much larger-scale ImageNet [42] dataset. We adopt the same train/test split settings as [13, 34]. More specifically, we report our results on 3 different test datasets: "2-hops", "3-hops" and the whole "All" ImageNet set. These datasets are constructed according to how similar the classes are related to the classes in the ImageNet 2012 1K dataset. For example, "2-hops" dataset (around 1.5K classes) includes the classes from the ImageNet 2011 21K set which are semantically very similar to the ImageNet 2012 1K classes. "3-hops" dataset (around 7.8K classes) includes the classes that are within 3 hops of the ImageNet 2012 1K classes, and the "All" dataset includes all the labels in ImageNet 2011 21K. There are no common labels between the ImageNet 1K class and the classes in these 3-dataset. It is also obvious to see that as the number of class increases, the task becomes more challenging.

As for knowledge graph, we use the sub-graph of the WordNet [32], which includes around 30K object nodes[2].

Training details. Note that to perform testing on 3 different test sets, we only need to train one set of ConvNet and GCN. We use two different types of ConvNets as the base network for computing visual features: Inception-v1 [46] and ResNet-50 [17]. Both networks are pre-trained using the ImageNet 2012 1K dataset and no fine-tuning is required. For Inception-v1, the output feature of the second to the last layer has 1024 dimensions, which leads to $D = 1024$ object classifiers in the last layer. For ResNet-50, we have $D = 2048$. Except for the changes of output targets, other settings of training GCN remain the same as those of the previous experiments on NELL and NEIL. It is worthy to note that our GCN model is robust to different sizes of outputs. The model shows consistently better results as the representation (features) improves from Inception-v1 (68.7% top-1 accuracy in ImageNet 1K val set) to ResNet-50 (75.3%).

We evaluate our method with the same metric as the previous experiments: the percentage of hitting the ground-truth labels among the top k predictions. However, instead of only testing with the unseen object classifiers, we include both training and the predicted classifiers during testing, as suggested by [13, 34]. Note that in these two settings of experiments, we still perform testing on the same set of images associated with unseen classes only.

Testing without considering the training labels. We first perform experiments excluding the classifiers belonging to the training classes during testing. We report our results in Table. 5a. We compare our results to the recent state-of-the-art methods SYNC [4] and EXEM [5]. We show experiments with the same pre-trained ConvNets (Inception-v1) as [4, 5]. Due to unavailability of their word embeddings for all the nodes in KG, we use a different set of word embeddings (GloVe) ,which is publicly available.

[2]http://www.image-net.org/explore

Model	Word Embedding	Hit@k (%)				
		1	2	5	10	20
[53]	GloVe	7.8	11.5	17.2	21.2	25.6
Ours	GloVe	**18.5**	**31.3**	**50.1**	**62.4**	**72.0**
[53]	FastText	9.8	16.4	27.8	37.6	48.4
Ours	FastText	**18.7**	**30.8**	**49.6**	**62.0**	**71.5**
[53]	GoogleNews	13.0	20.6	33.5	44.1	55.2
Ours	GoogleNews	**18.3**	**31.6**	**51.1**	**63.4**	**73.0**

Table 6. Results with different word embeddings on ImageNet (2 hops), corresponding to the experiments in Table 5a.

Therefore, we first investigate if the change of word-embedding is crucial. We show this via the ConSE baseline. Our re-implementation of ConSE, shown as "ConSE(us)" in the table, uses the GloVe whereas the ConSE method implemented in [4, 5] uses their own word embedding. We see that both approaches have similar performance. Ours is slightly better in top-1 accuracy while the one in [4, 5] is better in top-20 accuracy. Thus, with respect to zero-shot learning, both word-embeddings seem equally powerful.

We then compare our results with SYNC [4] and EXEM [5]. With the same pre-trained ConvNet Inception-v1, our method outperforms almost all the other methods on all the datasets and metrics. On the "2-hops" dataset, our approach outperforms all methods with a large margin: around 6% on top-1 accuracy and 17% on top-5 accuracy. On the "3-hops" dataset, our approach is consistently better than EXEM [5] around $2 \sim 3\%$ from top-5 to top-20 metrics.

By replacing the Inception-v1 with the ResNet-50, we obtain another performance boost in all metrics. For the top-5 metric, our final model outperforms the state-of-the-art method EXEM [5] by a whooping 20.9% in the "2-hops" dataset, 3.5% in the "3-hops" dataset and 1% in the "All" dataset. Note that the gain is diminishing because the task increases in difficulty as the number of unseen classes increases.

Sensitivity to word embeddings. Is our method sensitive to word embeddings? What will happen if we use different word embeddings as inputs? We investigate 3 different word embeddings including GloVe [36] (which is used in the other experiments in the paper), FastText [20] and word2vec [31] trained with GoogleNews. As for comparisons, we have also implemented the method in [53] which trains a direct mapping from word embeddings to visual features without knowledge graphs. We use the Inception-v1 ConvNet to extract visual features. We show the results on ImageNet (with the 2-hops setting same as Table 5a). We can see that [53] highly relies on the quality of the word embeddings (top-5 results range from 17.2% to 33.5%). On the other hand, our top-5 results are stably around 50% and are much higher than [53]. With the GloVe word embeddings, our approach has **a relative improvement of almost 200%** over [53]. This again shows graph convolutions with knowledge graphs play a significant role in improving zero-shot recognition.

Test Image	ConSE (10)	Ours
	panthera tigris(train)	tigress (test)
	tiger cat (train)	**bengal tiger (test)**
	felis onca (train)	panthera tigris (train)
	leopard (train)	tiger cub (test)
	tiger shark (train)	tiger cat (train)
	rock beauty (train)	**butterfly fish (test)**
	ringlet (train)	rock beauty (train)
	flagpole (train)	damselfish (test)
	large slipper (test)	atoll (test)
	yellow slipper (train)	barrier reef (test)
	tractor (train)	tracked vehicle (test)
	reaper (train)	tractor (train)
	thresher (train)	propelled vehicle (test)
	trailer truck (train)	reaper (train)
	motortruck (test)	forklift (train)

Figure 6. Visualization of top 5 prediction results for 3 different images. The correct prediction results are highlighted by red bold characters. The unseen classes are marked with a red "test" in the bracket. Previously seen classes have a plain "train" in the bracket.

Testing with the training classifiers. Following the suggestions in [13, 34], a more practical setting for zero-shot recognition is to include both seen and unseen category classifiers during testing. We test our method in this generalized setting. Since there are very few baselines available for this setting of experiment, we can only compare the results with ConSE and DeViSE. We have also re-implemented the ConSE baselines with both Inception-v1 and ResNet-50 pre-trained networks. As Table 5b shows our method almost doubles the performance compared to the baselines on every metric and all 3-datasets. Moreover, we can still see the boost in of performance by switching the pre-trained Inception-v1 network to ResNet-50.

Visualizations. We finally perform visualizations using our model and ConSE with $T = 10$ in Fig. 6 (Top-5 prediction results). We can see that our method significantly outperforms ConSE(10) in these examples. Although ConSE(10) still gives reasonable results in most cases, the output labels are biased to be within the training labels. On the other hand, our method outputs the unseen classes as well.

5. Conclusion

We have presented an approach for zero-shot recognition using the semantic embeddings of a category and the knowledge graph that encodes the relationship of the novel category to familiar categories. Our work also shows that a knowledge graph provides supervision to learn meaningful classifiers on top of semantic embeddings. Our results indicate a significant improvement over current state-of-the-art.

Acknowledgement: This work was supported by ONR MURI N000141612007, Sloan, Okawa Fellowship to AG and NVIDIA Fellowship to XW. We would also like to thank Xinlei Chen, Senthil Purushwalkam, Zhilin Yang and Abulhair Saparov for many helpful discussions.

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label embedding for attribute-based classification. In *CVPR*, 2013. 2

[2] J. L. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting Deep Zero-Shot Convolutional Neural Networks using Textual Descriptions. In *ICCV*, 2015. 2

[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010. 4

[4] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized Classifiers for Zero-Shot Learning. In *CVPR*, 2016. 2, 4, 7, 8

[5] S. Changpinyo, W.-L. Chao, and F. Sha. Predicting Visual Exemplars of Unseen Classes for Zero-Shot Learning. In *ICCV*, 2017. 2, 7, 8

[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 *ECCV*, 2016. 2

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 5

[8] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. *ICCV*, 2013. 2, 4

[9] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-Scale Object Classification Using Label Relation Graphs. In *ECCV*, 2014. 2

[10] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a Classifier: Zero-Shot Learning Using Purely Textual Descriptions. In *ICCV*, 2013. 2

[11] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2

[12] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic Label Sharing for Learning with Many Categories. In *ECCV*, 2010. 2

[13] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 1, 2, 4, 5, 7, 8

[14] Y. Fu and L. Sigal. Semi-supervised Vocabulary-informed Learning. In *CVPR*, 2016. 2

[15] Z. Fu, T. Xiang, E. Kodirov, and S. Gong. Zero-Shot Object Recognition by Semantic Manifold Distance. In *CVPR*, 2015. 2

[16] B. Hariharan and R. Girshick. Low-shot Visual Recognition by Shrinking and Hallucinating Features. In *CoRR*, 2017. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[18] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016. 2

[19] D. Jayaraman and K. Grauman. Zero-shot recognition with unreliable attributes. In *NIPS*, pages 3464–3472, 2014. 2

[20] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016. 8

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[22] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017. 2, 3, 4

[23] E. Kodirov, T. Xiang, and S. Gong. Semantic Autoencoder for Zero-Shot Learning. In *CVPR*, 2017. 2

[24] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 2

[25] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-Based Classification for Zero-Shot Visual Object Categorization. In *TPAMI*, 2014. 2

[26] Y. Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. In *IJCAI*, 2016. 2

[27] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 4

[28] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 7

[29] K. Marino, R. Salakhutdinov, and A. Gupta. The More You Know: Using Knowledge Graphs for Image Classification. In *CVPR*, 2017. 2

[30] 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 *ECCV*, 2012. 2

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013. 5, 8

[32] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 4, 7

[33] I. Misra, A. Gupta, and M. Hebert. From Red Wine to Red Tomato: Composition with Context. In *CVPR*, 2017. 1, 2

[34] 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. In *ICLR*, 2014. 1, 2, 4, 5, 7, 8

[35] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot Learning with Semantic Output Codes. In *NIPS*, 2009. 2

[36] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014. 4, 8

[37] R. Qiao, L. Liu, C. Shen, and A. van den Hengel. Less is more: zero-shot learning from online textual documents with noise suppression. In *CVPR*, 2016. 2

[38] M. Rohrbach, S. Ebert, and B. Schiele. Transfer learning in a transductive setting. In *NIPS*, 2013. 2

[39] M. Rohrbach, M. Stark, and B. Schiele. Evaluating Knowledge Transfer and Zero-Shot Learning in a Large-Scale Setting. In *CVPR*, 2011. 2

[40] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What helps where - and why? semantic relatedness for knowledge transfer. In *CVPR*, 2010. 2

[41] B. Romera-Paredes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 2

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 7

[43] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to Share Visual Appearance for Multiclass Object Detection. In *CVPR*, 2011. 2

[44] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. Zero-Shot Learning Through Cross-Modal Transfer. In *ICLR*, 2013. 2

[45] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 2

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015. 7

[47] P. Wang, Q. Wu, C. Shen, A. van den Hengel, and A. Dick. FVQA: Fact-based Visual Question Answering. In *CoRR*, 2016. 2

[48] J. Weston, S. Bengio, and N. Usunier. Large Scale Image Annotation: Learning to Rank with Joint Word-Image Embeddings. In *ECML*, 2010. 2

[49] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016. 2

[50] Y. Xian, B. Schiele, and Z. Akata. Zero-Shot Learning - The Good, the Bad and the Ugly. In *CVPR*, 2017. 2

[51] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015. 4

[52] Z. Yang, W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016. 4

[53] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, 2017. 8

[54] Z. Zhang and V. Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015. 2

[55] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016. 2