Encoding Visual Information using Node Embeddings

Varun Nambikrishnan
Stanford University
Stanford, CA
varun14@stanford.edu

Karey Shi
Stanford Univeristy
Stanford, CA
kareyshi@stanford.edu

Karan Singhal
Stanford University
Stanford, CA
ksinghal@stanford.edu

Abstract

Node embedding techniques have been recently popularized for a variety of network analysis tasks, including node classification, link prediction, and clustering. We explore how well these methods perform on a graph containing Flickr interaction data, in which each node corresponds to an image, and images are linked by user interactions on the Flickr platform. Flickr images are a rich medium, as their pixel data provides a benchmark to compare node embedding techniques against. Specifically, we first explore how well node embedding techniques capture image similarity without access to image pixels by comparing against state-of-the-art convolutional neural networks (CNNs). We then investigate the performance of node embedding techniques in image classification, using features extracted from CNNs to produce a skyline. Finally, we explore whether the information encoded in the Flickr graph can augment CNN performance in image classification using RolX and feature concatenation. Our code can be found on Github [1].

1. Introduction

Currently, vast amounts of image information reside in online photo sharing communities, such as Flickr, Pinterest, or Instagram, just to name a few. These expansive and highly interactive communities maintain rich image sharing networks with large amounts of metadata describing the nature of interactions within these social contexts. With the existence of these modern data sources for images, the content and definition of these images on the Web now extends beyond its visual collection of pixels to incorporate the relational metadata present within its social network.

While deep learning methods, such as convolutional neural networks (CNNs), have soared in popularity due to the efficacy of capturing image information through examining pixel data, we aim to explore the extent to which the content of images can be modeled by the relational metadata and information present in a network of images in a social media context, such as those found on Flickr. In such networks, relational metadata describes a variety of interactions, such as which images were taken from the same location, which images belong to a certain gallery or group, which common tags are associated among images, and which images were captured by friends on the sharing platform.

In this project, our contributions fall under five key research questions, which we believe are substantial and novel enough to further the understanding of node embedding performance:

1. How much information about visual similarity among images can we glean from relational interaction metadata using node embedding methods?
2. Using these embeddings, can we perform well on downstream tasks, such as image classification?
3. How does performance on such tasks compare with that of state-of-the-art CNN methods?
4. Can we improve performance on these downstream tasks by combining node embeddings with features extracted from CNNs?
5. Can other techniques that leverage relational data (e.g. RolX) improve performance on these tasks?

By focusing on these questions, we build off of the intuition that the relationships discovered within the social-network metadata likely parallels similarities in visual content, and our project seeks to determine the extent to which node embeddings generated from relational and interaction metadata can represent and encompass useful information about images.

In the process of exploring these five key questions, we also explore a number of smaller research questions along the way. For example, we use three different node embedding methods in our work, providing insight into how these methods compare at encoding visual and non-visual information in the Flickr graph.

2. Related Work

There is previous work using the Flickr dataset by McAuley and Leskovec which explores the use of social-network metadata from online image-sharing platforms for downstream image labeling tasks [6]. These tasks include image label prediction, tag recomendation, and group recommendation, and in this work, four datasets of Flickr images and their corresponding metadata are used. This work examines the performance of three different models: an SVM using only low-level image features, an SVM containing low-level features and metadata features modeled independently, and a graphical model leveraging the relational metadata. The authors found that the graphical model outperforms methods that utilize only image-content-based features, and certain relational features (e.g. shared membership to a gallery, shared location) prove to be strong predictors for the considered classification tasks. This work was published before some node embedding techniques were developed and when SVMs were considered state of the art, and our aim is to validate whether newer graph techniques have the capacity to perform well against newer image classification methods, such as CNNs. Even if graphical models do not prove to be optimal anymore, there is still much to be explored in terms of the extent of information that they can capture about an image without utilizing the per pixel information that the image itself contains. It is also likely that node embeddings capture structural information orthogonal from the information stored in visual features, which can improve performance in downstream tasks in tandem with visual features.

Other works, such as those by Hamilton et al. and Goyal et al., have reviewed various node embedding techniques, and from their characterizations of these diverse techniques, we can examine the respective strengths and advantages of each approach [9][8]. These papers separate the space of node embedding approaches into three general categories: matrix factorization, random walk, and deep learning based methods.

In matrix factorization methods, information from the graph is recorded in a matrix, which is then factorized to produce embeddings. These methods involve a deterministic measure of node similarity, including inner-product methods, where the strength of the relationship between 2 nodes is proportional to the dot product of their embeddings. These inner-product methods include Graph Factorization (GF), GraRep, and HOPE.

Random walk methods define node similarity in a more flexible and stochastic manner, and such approaches include DeepWalk and node2vec. These shallow embedding approaches also use a decoder based on inner products, but instead of decoding a deterministic node similarity measure as done by the matrix factorization approaches, these methods optimize embeddings to encode the statistics of random walks.

Deep learning based approaches are diverse, but deep autoencoders have been used for dimensionality reductionthese approaches utilize auto encoder methods to synthesize information about a node's neighborhood. Examples of deep learning based node embedding approaches include Deep Neural Graph Representations (DNGR) and Structural Deep Network Embeddings (SDNE).

We also explore whether other techniques that leverage relational graph information can lead to better performance on a downstream task such as image classification. We explore one such method, RolX (Role eXtraction), which is an unsupervised method proposed by Henderson et. al to extract structural roles from networks by recursively expanding features of a node using the features of its neighbors [5]. While the node embedding techniques already take advantage of the graph's structure, we believe that RolX could improve the performance of CNN embeddings, as they don't naturally leverage relational data as node2vec and SDNE do.

3. Dataset and Graph Representation

We use the dataset of Flickr image relationships built by McAuley and Leskovec [6]. This dataset was aggregated from four popular datasets of images with human annotated groundtruth, and the image nodes have been augmented with social network metadata collected from Flickr. These four sources are the *PASCAL Visual Object Challenge* (PASCAL), the *MIR Flickr Retrieval Evaluation* (MIR), the *ImageCLEF Annotation Task* (CLEF), and the *NUS Web Image Dataset* (NUS) [3].

In addition to utilizing the entire cumulative dataset of image metadata from these four datasets (which, after filtering on amount of substantial metadata available, contains 105,938 images), we have decided to further focus on the NUS dataset, which, in its entirety, consists of 269,642 images, where Flickr sources are available for all images. We have acquired the raw image data for the NUS dataset, which was used for implementing our visual-content based CNN approach (of the 105,938 images for which we have complete metadata information, we have images for 89,251) [3]. For all the Flickr images in our dataset, the original metadata collected for each photo includes the photo title, description, location, timestamp, viewcount, upload date; user information, photo tags, and the groups/collections/galleries that the image belongs to; and comment threads associated with each photo.

The metadata is aggregated as node and edge features of a network representing the collection of images available. Each node represents an image, and each edge represents a relationship between two images, where an edge is drawn if any of the edge features are non-zero. Node features include the properties of the image, including the groups or categories that the image belongs to in its respective dataset. Edge features include seven properties as listed by McAuley and Leskovec: the number of common tags/groups/collections/galleries, and indicators for whether both photos were taken in the same location, taken by the same user, and taken by mutual contacts or friends. To get a better context of the size of the graph, for the entire curated Flickr image dataset where information of all pieces of metdata is available, we have a resulting 105,938 nodes (the number of images), and 2,316,948 edges. For the induced subgraph on our NUS images (for which we have pixel data), there are 89,251 nodes.

We also created other induced subgraphs to create image classification tasks from the larger dataset. We use these image classification tasks to measure how useful node embeddings are for image classification, both alone and in tandem with CNN features. We detail these induced subgraphs in Section 4.4.

4. Methods

At a high level, our approach involves computing image embeddings using both node embeddings on the Flickr images graph and CNN embeddings trained on the images themselves. We explore three node embedding techniques to serve as our exploration of graphical models, and we use CNNs to serve as a skyline for performance in downstream tasks, such as image classification. As an initial metric of evaluation for the quality of our embeddings, we use the cosine distances between images to establish a notion about the similarity of different embeddings (we discuss our method for doing so below). This metric allows us to determine how well node embeddings capture the visual similarity encoded by CNN image features, and it allows us to compare the relationships among our various node embedding techniques. We then use our produced embeddings for an image classification downstream task, to better understand how performance compares for embeddings that capture different types of information about the images.

4.1. Node Embedding Techniques

We define a node embedding on a graph $G = (V, E)$ as defined in Goyal, namely a mapping

$$f : v^i \mapsto y^i \in \mathbb{R}^d, \forall i \in [n]$$

such that $d \ll |V|$ and the function f preserves some proximity measure defined on graph G [6]. Essentially, we are trying to map each node to a low-dimensional feature vector while still preserving the connections between vertices. For our experiments, we chose $d = 128$, as it is a common choice in the literature. We choose one embedding technique from each of the three high-level categories described in Section 2: HOPE, node2vec, and SDNE.

4.1.1 HOPE embeddings

The first node embedding method we examine is *Higher Order Proximity preserved Embeddings*, or HOPE [7]. HOPE is a factorization based method, which represents the graph as a matrix and then uses some form of factorization to generate the embeddings. The objective function HOPE aims to compute:

$$\min \|S - Y_s Y_t^T\|_F^2$$

where S is the similarity matrix of the graph and Y_s, Y_t are the source and target node embeddings. Different similarity measures can be used, including Katz Index, Rooted Page Rank, Common Neighbors, and Adamic-Adar score. For our experiments we chose the Katz Index, which is a weighted summation over the path set of two vertices. We must also choose a decay parameter, β, which determines how fast the weight of a path decays as the length of path grows. For our experiments, we chose $\beta = .01$.

4.1.2 node2vec embeddings

Our second node embedding method is *node2vec*, a biased random walk procedure that uses a flexible notion of neighborhood that balances capturing a node's community and its structural role in the network [2]. We seek to maximize the following objective, which is based off of the skip-gram architecture in word2vec:

$$\max \sum_{u \in V} \log(P(N_s(u)|f(u)))$$

This objective maximizes the log probability of observing a network neighborhood $N_s(u)$ for a node u (generated by the sampling strategy s), conditioned on the feature representation f of that node. We control the sampling strategy using an in-out parameter q, which controls how biased the search is towards "inward" nodes vs. "outward" nodes in generating neighborhoods. These searches model BFS and DFS, respectively, and adjusting q allows us to interpolate between the two search strategies, providing us the aforementioned flexible notion of node neighborhoods. We also have a return parameter p that controls the likelihood of returning to a node immediately after visiting it in a walk. Setting this high encourages more exploration (no 2-hop redundancy) while setting it low keeps the walk close to the original node. Using random walks allows node2vec to be much more time/space efficient than a pure BFS/DFS strategy. For our experiments, we chose the number of walks per node u to be 10, the length of each walk to be 80, the return parameter p to be 1, and the in-out parameter q to be 5.

4.1.3 SDNE embeddings

The last embedding method we explore is *Structural Deep Network Embedding* or SDNE [10]. This is a deep learning based method that uses autoencoders to reduce dimensionality, as they have the ability to model highly non-linear structure in graphs. SDNE tries to preserve both first and second order proximities, where first order proximity is the pairwise proximity between nodes, and second order proximity is the proximity of two nodes' neighborhoods. SDNE consists two parts: an unsupervised component that reconstructs the neighborhood structure of each vertex, to preserve second order proximity, and a supervised component that uses the first-order proximity to restrict and refine the autoencoders' representation in latent space. We seek to jointly minimize these proximities with the following objective function:

$$\mathcal{L}_{mix} = \mathcal{L}_{2nd} + \mathcal{L}_{first} + \nu\mathcal{L}_{reg}$$

We have that $\mathcal{L}_{2nd}$ is the autoencoder loss function, minimizing the reconstruction of the nodes from their embeddings:

$$\mathcal{L}_{2nd} = \|(\hat{X} - X) \odot B\|$$

B is used to more heavily penalize the reconstruction error of non-zero elements, to get the autoencoder to avoid reconstructing only zero elements. $\mathcal{L}_{1st}$ penalizes similar vertexes that are mapped far away in the embedding space:

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{ij}\|y_i - y_j\|_2^2$$

Where s_ij is the similarity score between nodes i and j, as mentioned in the HOPE embedding section. For our experiments we used a 3 hidden layer autoencoder, and otherwise chose the standard parameters in the implementation provided by GEM and in Goyal et al. [8].

4.2. Convolutional Neural Network Embeddings

To provide a benchmark set of embeddings that capture visual information about images extremely well, we rely on convolutional neural networks (CNNs), which have risen to great popularity within the computer vision community for classifying images. CNNs have achieved extremely high accuracy on diverse image classification tasks, outperforming humans in some cases. Because CNNs progressively process information in images from low-level features to high-level features before making a final classification decision, we can extract the high-level features from the output of second to last layer of the CNN, i.e. the layer before the classification (fully-connected with softmax) layer. By passing each image through our CNNs, we can use the output of this second-to-last layer as a visual encoding of an image. These features are useful to serve as a skyline for performance on downstream tasks, as they encode the most information about the actual visual content of an image.

We use two of the ResNet architectures to produce visual features, ResNet-50 and ResNet-18 [4]. ResNets can be incredibly deep neural network architectures, with skip connections interleaved throughout the depth of the network to provide a fast-forwarded channel for information to pass down to later layers, which overcomes some disadvantages with extremely deep networks. These models are very popular in the computer vision community, and currently produce near state-of-the-art performance in image classification for the ImageNet dataset. We extract visual features by running Flickr images forward through the networks, extracting activations immediately before the fully-connected classification layer. Note that no training or fine-tuning of these networks on our datasets was done. For ResNet-18, this produces embeddings with dimensionality 512, and for ResNet-50, this produces embeddings with dimensionality 2048.

To produce the embeddings for the 89k images for which we had both pixel data and coverage in the dataset provided by McAuley and Leskovec on such deep networks, we set up a GPU instance on Google Cloud Platform, using an NVIDIA K80 GPU. Before sending images through the networks, we rescaled them to be 224x224 and normalized the mean and standard deviation of pixel values according to the mean and standard deviation of the pixel values described in the original paper. While the paper describes creating multiple crops and using other augmentation techniques (along with ensemble techniques), we do not do this.

4.3. Embedding Similarity

A key element of our approach is a technique to better understand how similar different sets of image embeddings are, which works across embedding techniques (which may embed nodes into different spaces) and even different dimensionalities of embeddings. The high-level idea is to take two different embeddings of the same set of nodes and treat them as models for determining "distance" between nodes. If we sample two nodes and ask the two models to make a prediction about how distant the nodes are, if the models make similar (correlated) predictions over many samples, then the models have a similar notion of node similarity, and the embeddings are similar. We can compute correlations using the Pearson correlation (to capture linear relationships between the similarities predicted by our models) and the Spearman (or rank) correlation (to capture non-linear relationships). We can also plot predicted distances with one embedding approach versus another, to produce a visual indicating how closely related model predictions are. With this high-level intuition, we present the full algorithm in Algorithm 1.

```
Data: 2 sets of embeddings for a shared group of
      nodes (emb1, emb2), number of samples
Result: A plot of predicted distances, Pearson and
        Spearman correlations
X = [], Y = [];
for i in range(num_samples) do
    left, right = random.sample(nodes, 2);
    dist1 = cosine_dist(emb1(left), emb1(right));
    dist2 = cosine_dist(emb2(left), emb2(right));
    X.append(dist1);
    Y.append(dist2);
end
plot(X, Y);
p_r = pearson(X, Y);
s_r = spearman(X, Y);
```

Algorithm 1: Computing embedding similarity for two embeddings for a shared set of nodes.

Note that our use of the Spearman correlation does not suffer from instability issues related to having repeated elements, as it is unlikely that the same exact pair of elements is sampled twice.

4.4. Image Classification

Given image embeddings produced from either node embedding techniques or CNNs, a feedforward neural network (or other techniques) can be used to predict classes for different images. Image classification is an interesting task in this context because it is not necessarily clear that embeddings that capture visual information the best will necessarily perform best at image classification (McAuley and Leskovec found that graphical methods performed better in some cases than simple visual methods).

Though we expect CNN features to perform better than the node embeddings we produce (because they have been highly optimized for image classification and pixels are a rich source of visual information), we suspect that the unique metadata that will likely be encoded in node embeddings will have some value, and it will be interesting to see how performance compares, especially across different subsets of the graph (do we do better where the graph is more dense, as in the "popular" subgraph described above?). Another interesting question to explore is whether we can achieve better performance by concatenating visual and graphical features.

To set up these image classification tasks, we first establish the exact classes of images, collect the relevant images for that task, induce a subgraph on those images, generate node embeddings using this subgraph, and train and test each of our node embeddings and ResNet embeddings using a neural network. The NUS dataset contains 81 labels in total, and each image in the dataset can be associated with one or more labels. We examine the distribution of the labels across the dataset and find that the labels are not close to being uniformly distributed across images, with multiple labels referring to overlapping concepts. Images can have up to 13 labels, and the most frequent label ('sky') contains 74190 images while the least frequent label ('map') contains only 60 images. We ultimately decide to construct classes by hand according to reasonable higher-level categories, aggregating together subcategories if necessary, and we evaluate performance across several image classification tasks with varying number of classes.

Our initial classification task is with 2 classes: 'person' and 'animal.' (Note: the 'animal' class is aggregated across multiple categories to help correct class imbalance, such as 'birds,' 'dog,' 'horses,' and 'cow'). We assign images a label of 'person' if it belongs to the 'person' category and not to any 'animal' subcategory. We assign images a label of 'animal' if it belongs to at least one 'animal' subcategory and not to any 'person' category. The induced subgraph has 32899 unique nodes (18948 person images and 13951 animal images) with 75742 edges.

We set up 2 other classification tasks with more classes involved for more opportunities of comparison, and especially to provide the ResNet with a more difficult task to see if our methods can supplement performance (it already achieves nearly perfect accuracy on a simple 2-class classification task). We construct a task with 3 classes: 'person,' 'animal,' 'plant.' The 'plant' class is aggregated in the same manner as the 'animal' class, and includes subcategories such as 'flowers,' 'tree,' and 'leaf.' This induced subgraph includes 42174 unique nodes (19720, 13277, 9177 images of category 'person,' 'animal,' and 'plant,' respectively) with 112229 edges. We also construct a task with 5 classes: 'person,' 'animal,' 'plant,' 'building,' and 'scenery.' The induced graph on images from these five classes has 26474 unique nodes with 97206 edges.

In addition to evaluating the performance of our node embeddings (HOPE, node2vec, SDNE) on these downstream tasks against our skyline performance with CNN embeddings, we also evaluate performance using augmented embeddings, namely CNN embeddings concatenated with node embeddings, and node embeddings concatenated with each other. We also examine the performance of incorporating relational information with our CNN embeddings through other techniques such as RolX, described in section 4.5.

4.5. Augmenting Embeddings with RolX

While embeddings are one way of utilizing graph structure, there are many other techniques to take advantage of relational data. One of these methods is RolX, which uses recursive feature extraction for a specified number of iterations to improve the initial feature representations of node [5]. In our case, the initial feature representation is simply the embedding for that node according to our node embedding techniques or CNNs. On each iteration, we iterate through each node and extend its embedding with the mean embedding of all its neighbors. We ran this for 1 and 2 iterations on all three node embeddings and the CNN embeddings. Since the size of the embeddings increases exponentially with the number of embeddings, doing RolX for more than a few iterations is unfeasible.

5. Results and Analysis

5.1. Hyperparameter Tuning

We performed grid search across the possible hyperparameters for each node embedding method, evaluating based on how similar each embedding was to the CNN embedding. Our intuition is that the ideal node embedding is able to capture the most visual information, for which the CNN embeddings serves as the skyline. For node2vec, we found that $q = .01$, $p = 1$ produced the highest correlation with the CNN embeddings, which aligns with random walks using a DFS-approach. This aligns with the intuition that images that are visually similar to one another would exhibit strong homophily-based relationships, and thus a macroscopic understanding of the network would be necessary to capture this organization. For SDNE, we found that choosing $B = 5$ produced the highest correlation, and for HOPE we found that $\beta = 1$ produced the highest correlation.

ResNet-18	1.000			
HOPE	0.007	1.000		
node2vec	0.051	0.437	1.000	
SDNE	0.046	0.383	0.709	1.000
	ResNet-18	HOPE	node2vec	SDNE

Table 1: Spearman correlation scores of cosine distances measured across various embedding methods.

5.2. Encoding Visual Similarity

Our first key task was examining the extent to which we can capture information about visual similarity using our various node embedding methods, which utilize only relational interaction metadata. We compute four sets of embeddings, one for each of the node embedding techniques we examine: CNN (our skyline, using ResNet-18), HOPE, node2vec, and SDNE. We sample 10,000 pairs of nodes of our popular subgraph, which consists of nodes with particularly high degree, and for each set of embeddings, we compute the cosine distance between each corresponding pair of node embeddings. This gives us four sets of cosine distances, and we evaluate both the Pearson and Spearman correlation scores for each pair of embedding method. Our Spearman correlation scores are reported in Table 1.

It is somewhat surprising to find that each of our node embedding techniques captures information that is mostly orthogonal to the information encoded by our CNN embeddings. This can be seen from the relatively low correlation scores between ResNet and the node embeddings. This suggests that the content gleaned from the relational data by HOPE, node2vec, and SDNE could be used to supplement ResNet in downstream tasks. Despite this lack of correlation with the visual features encoded by the CNN embeddings, the cosine-distances among node embeddings seem to be more highly correlated. This is reasonable to expect given that they are all encoding relational information, while the CNN embeddings are encoding an entirely different type of information (per-pixel visual information). Plots of some of the sampled similarities between embeddings on one of our subgraphs are contained in the Appendix.

5.3. Image Classification

After computing embedding similarity, we evaluate our node embeddings using our downstream task: image classification. From our examination of each embedding's capacity to encode visual similarity, we found that node2vec

	ResNet-18	node2vec	SDNE	HOPE
2-class	97.26	76.84	73.10	73.40
3-class	94.81	67.71	56.83	—
5-class	89.54	59.29	47.21	—

Table 2: Accuracies for image classification tasks across CNN and the three node embeddings.

had a higher correlation with the CNN embeddings than the other methods. As such, we chose focus more heavily on node2vec on our evaluation on downstream image classification, as it would provide the closest performance in comparison to the skyline of our CNN.

We use a simple Multi-layer Perceptron classifier to perform the downstream image classification tasks described in section 4.4. Our network consists of one hidden layer of 100 nodes with a ReLU activation. We use the Adam solver for optimization, and also use early stopping using a validation set 10% of the size of the training set. The classifier takes in image and node embeddings as input and outputs probabilities (using a softmax final layer) for the desired classes.

5.3.1 Standard Embeddings

Our results for the 2, 3, and 5-class image classification task can be found in table 2. We did not create node embeddings using HOPE for the 3-class and 5-class tasks because the induced subgraph was too large to compute. We note that node2vec achieves the highest performance out of the three node embedding techniques. It is also interesting to see that SDNE performance drops significantly as the classification task becomes more complex.

5.3.2 Analysis of Encoded Information

To further inspect how the node embedding techniques are using relational data to encode information about the images, we used t-SNE dimensionality reduction on our embeddings and plotted our embeddings for the 2-class classification task in 2D space. In Figure 1, we have the plot for the ResNet-18 embeddings on this subgraph, and in Figure 2, we have the plot for the node2vec embeddings. We can see that the ResNet-18 embeddings creates distinct clusters between the two image classes, whereas the node2vec embedding does not do as well, although there is still some separation between classes. This indicates that it is unlikely that changing our image classification technique (e.g. tuning hyperparameters) is unlikely to improve results significantly for the node2vec embedding.

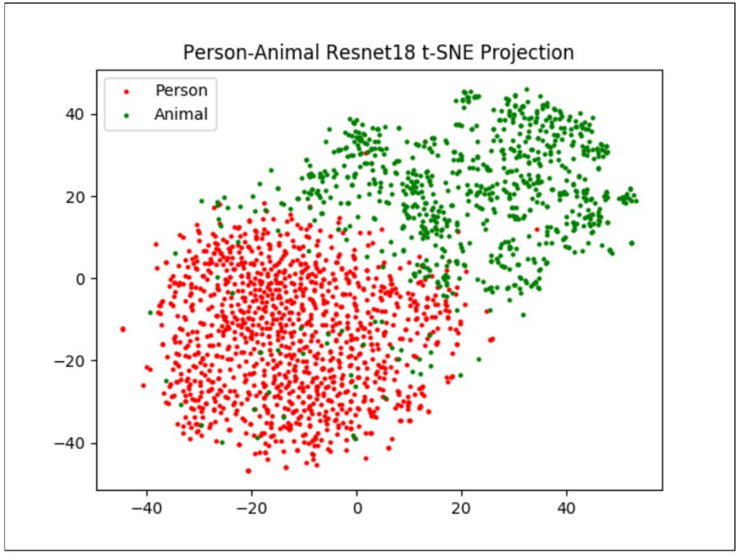


Figure 1: TSNE plot for ResNet-18 embeddings on person-animal subgraph.

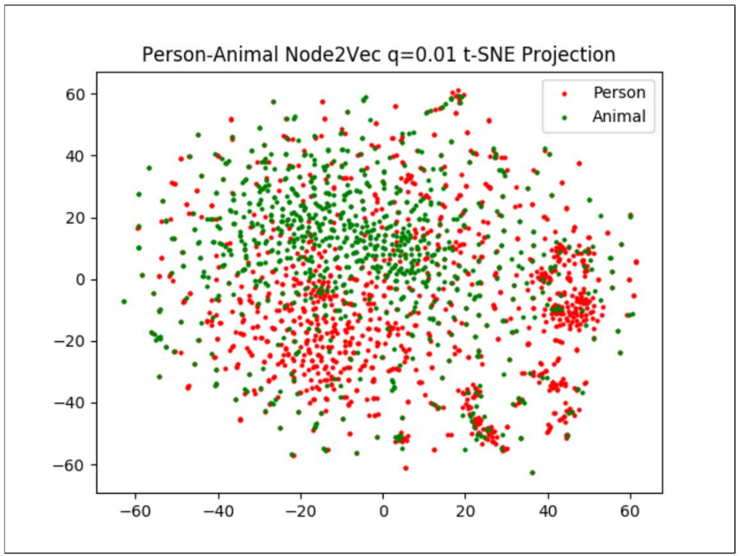


Figure 2: TSNE plot for node2vec embeddings on person-animal subgraph.

5.3.3 Concatenated Embeddings

We then examine whether concatenating different embeddings together can help boost performance on these downstream tasks. We found positive results when concatenating the ResNet embeddings with the node2vec embeddings, which show that the added relational information encoded by the node2vec embeddings can help performance on these downstream tasks as opposed to features solely derived from visual information. Our results for concatenation improvement can be found in Table 3

5.3.4 Embeddings with RolX

We also inspect the effects of applying the RolX on the CNN embeddings. We believe that this would provide

	ResNet-18	ResNet-18 + node2vec
2-class	97.26	97.33
3-class	94.81	95.07
5-class	89.54	89.43

Table 3: Comparison of accuracies for image classification tasks between ResNet-18 embeddings and ResNet-18 embeddings concatenated with node2vec embeddings.

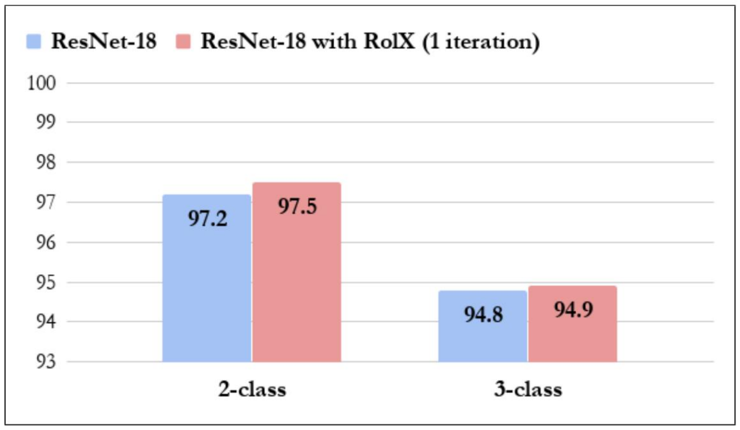


Figure 3: Accuracies for 2-class and 3-class tasks, highlighting the accuracy boost with embeddings produced by ResNet-18 with Rolx for 1 iteration.

another way to incorporate relational information while still taking advantage of the rich visual information encoded by the CNN embeddings. We perform RolX on the respective subgraphs induced for the first 2 tasks (2-class and 3-class), and our results are shown in Figure 3. We find that for the 2-class classification task, we were able to reduce the classification error by 10% with embeddings produced by ResNet-18 with RolX used for 1 iteration.

We additionally apply RolX to our node2vec embeddings to see if we can observe a similar boost in performance on these downstream tasks. In Figure 4, we can see that upon applying RolX for 2 iterations, we can achieve a nontrivial increase in performance. We did not observe as much of an increase with utilizing only 1 iteration of RolX.

5.4. Analysis of Graph Quality

The original Flickr graph was very large, which meant that it would take an extremely long time to run/tune our embeddings. Additionally, the labels were quite poor (lots of misclassified examples/ambigous classifications). To solve this issue, we chose to create 2,3, and 5-class subgraphs (with better labels) to run our node embeddings. However, this meant that we lost a lot of nodes/edges, meaning there was much less network structure from the node embeddings to learn from. These subgraphs were quite sparse, with very low average clus-

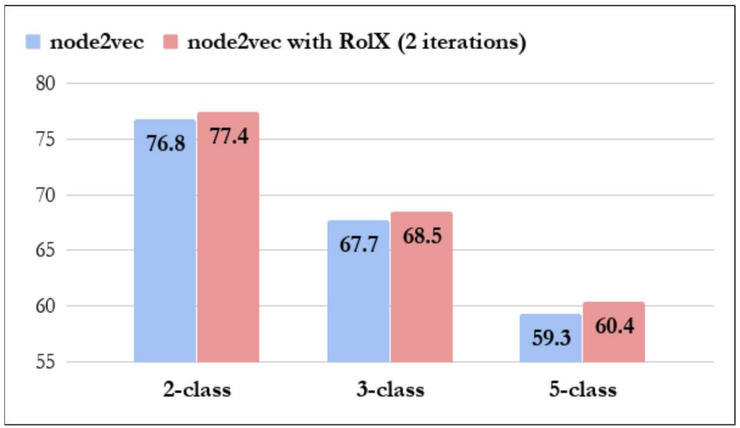


Figure 4: Accuracies for 2, 3, 5-class tasks, highlighting the accuracy boost with embeddings produced by node2vec with Rolx for 2 iterations.

tering coefficient, and thus were not optimal for running node embedding techniques. As such we ran experiments filtering which nodes we run classification on, by their node degree, to measure how well the node embeddings do as the information they have for a node increases. We see in Figure 5 that node2vec steadily produces a better classification accuracy as the degree of the node increases, while ResNet's accuracy does not improve with degree. This is promising, as it shows us that node2vec's performance scales well with the amount of information it can utilize. We also see in Figure 6, that concatenating the node2vec embedding with the ResNet embedding provides an increase over the ResNet embedding alone in the same experimental setup. This shows us that node2vec is capturing additional, relevant information for the image classification task, and shows us that relational information can complement visual information in these types of tasks. Lastly, we see in Figure 7 that concatenating SDNE and node2vec embeddings also provides an increase in accuracy, showing that the different methods of node embeddings are capturing slightly different content even though they are both trained on the same graph. In general, we see that concatenation leads to improvement, especially in sparse graphs, where we need to capture almost all the information provided to get a satisfactory embedding.

6. Conclusion and Future Work

Unsurprisingly, we find that CNN embeddings perform better than the node embeddings at encoding visual information, as CNNs are optimized for understanding the visual content of images. We also note that the node embeddings seem to capture different content than the CNN embeddings, as the correlation between the two types of embeddings is also quite low. However, we see that the

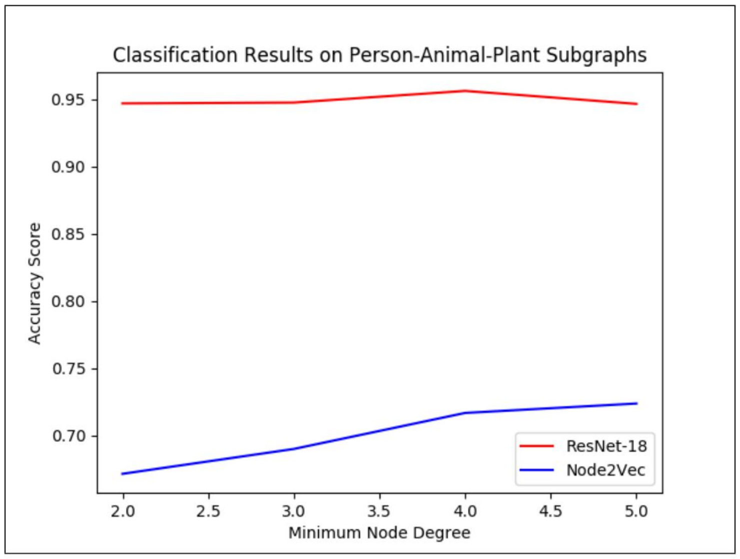


Figure 5: Improvement in performance for node2vec embeddings upon using a subset of nodes with higher degree to train and evaluate our classifier. Note that classification score stays relatively constant for our ResNet-18 embeddings.

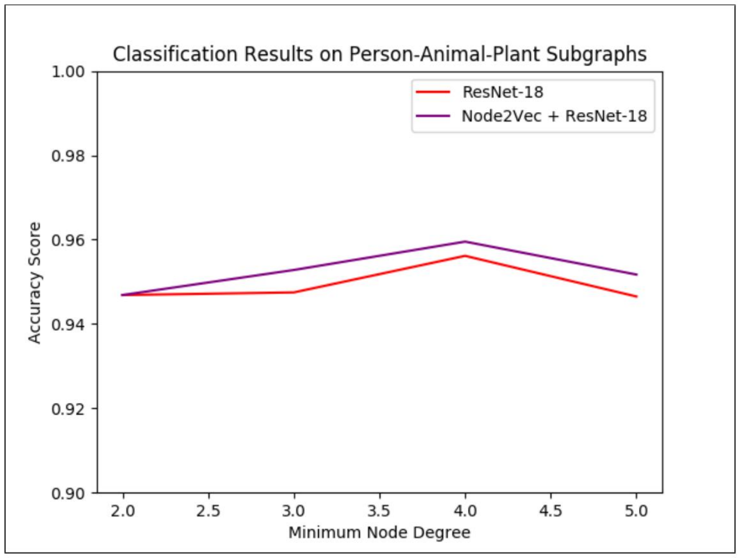


Figure 6: Improvement in performance for concatenated embeddings (ResNet-18 with node2vec) upon using subset of higher-degree nodes.

different types of node embeddings are well correlated with each other, which makes sense as they are synthesizing the same relational data. We also see that even though the content captured by node2vec is almost orthogonal to the CNN embeddings, we note that it performs reasonably well on our downstream image classification task, which indicates that it is capturing content useful for image classification, even though it may not be visual information. We also see that concatenating the node2vec embeddings with the CNN embeddings provides a small boost in classification accuracy, meaning the the information the node2vec embedding is relevant to our image

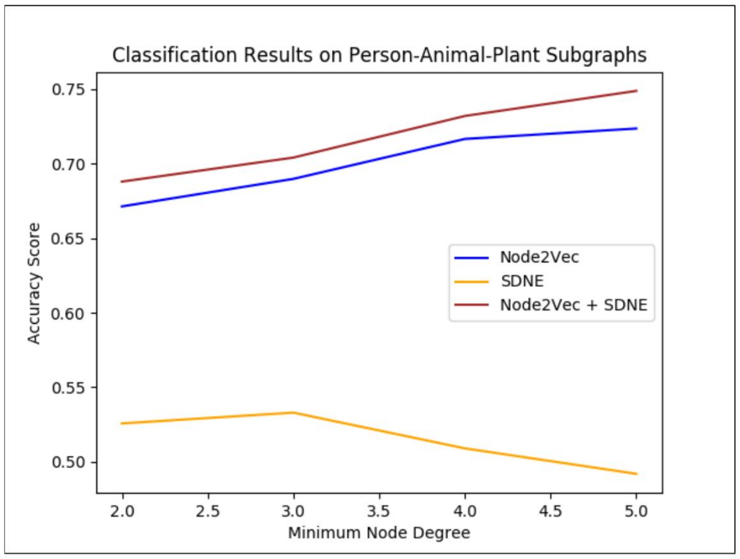


Figure 7: Improvement in performance when concatenating node2vec and SDNE upon using subset of higher-degree nodes. Note that SDNE alone does not improve in performance, but concatenation of the 2 embeddings does improve.

recognition task. This shows that even on relatively simple image classification tasks, relational data can be helpful in addition to the raw pixels. This point is advanced by the fact that using RolX embeddings improves accuracy over using plain embeddings.

We also notice that node2vec and SDNE differed quite dramatically in their classification, especially as the number of classes increases. SDNE fares significantly worse; however this is somewhat expected, as similar differences on the task of node classification in Goyal et. al's paper as well. It seems that the random walk approach may be better suited for classification tasks, while deep learning approaches such as SDNE are better for tasks such as link prediction. Training SDNE on a GPU may help bridge this gap.

We also found that the subgraphs from the original Flickr graph we created were quite sparse, as the edge:node ratio was usually between 2:1 and 4:1. This adversely impacted our node embeddings as it meant there was much less network structure than expected. We verified this by comparing the classification accuracy of the highest degree nodes (the nodes with the most graph structure to leverage) to the classification of lower degree nodes, and found that higher degree nodes had a much better classification accuracy. Thus, we believe that in a denser network, the node embedding methods would produce a more significant impact.

In future work, we would like to explore how node embeddings function on other image graphs, especially denser ones where the graph structure can be better utilized. Additionally, experimenting with other ways to classify nodes in a graph, such as iterative classification or loopy belief propagation, or a mixture of these methods could prove useful in image-related graphs. It would also be interesting to examine different ways to incorporate more nuanced information about the relationships between images to better inform our embeddings. We would also want to find more challenging classification tasks in which a ResNet would not be fully successful with, in which case adding relational data would be more significantly helpfulFurthermore, a survey paper on these types of experiments could help clarify in which situations certain classification methods are the most effective, which could be quite useful to the research community.

References

[1] Project source code. `https://github.com/karan1149/cs224w-project`.

[2] J. L. A. Grover. node2vec: Scalable feature learning for networks. 2016.

[3] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2012.

[6] J. McAuley and J. Leskovec. Image labeling on a network: Using social-network metadata for image classification. 2012.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[8] E. F. P. Goyal. Graph embedding techniques, applications, and performance: A survey. 2017.

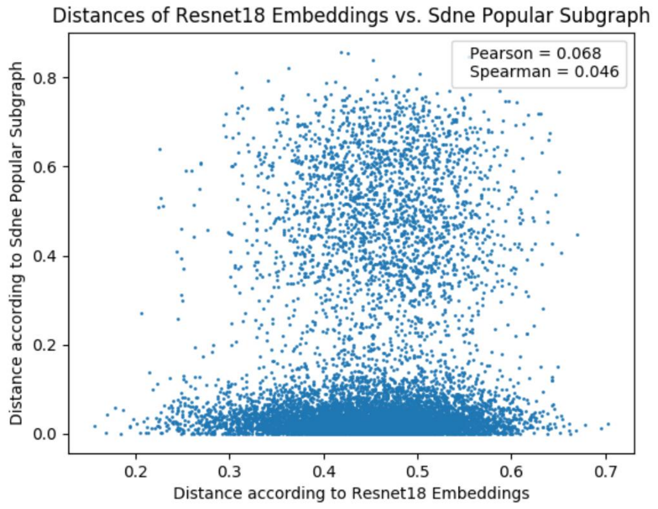
[9] J. L. W. Hamilton, R. Ying. Representation learning on graphs: Methods and applications. 2018.

[10] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

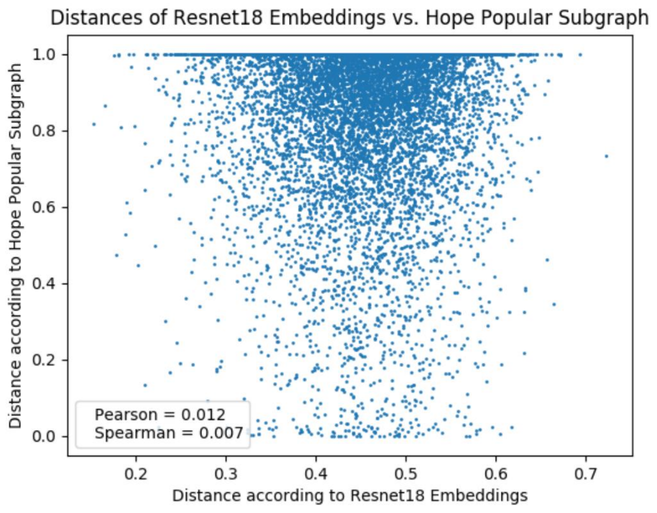
7. Appendix: Embedding Similarity Plots

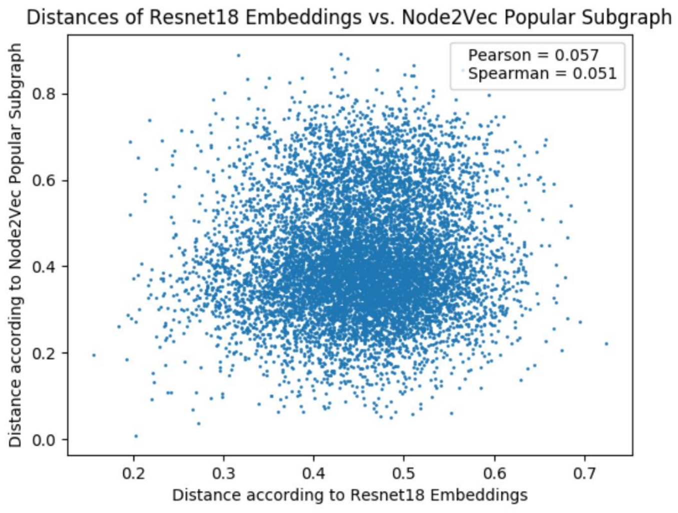
7.1. Comparison with ResNet-18

The following graphs show the relationships between predicted image distances according to different embeddings, as well as the corresponding Pearson and Spearman correlations.



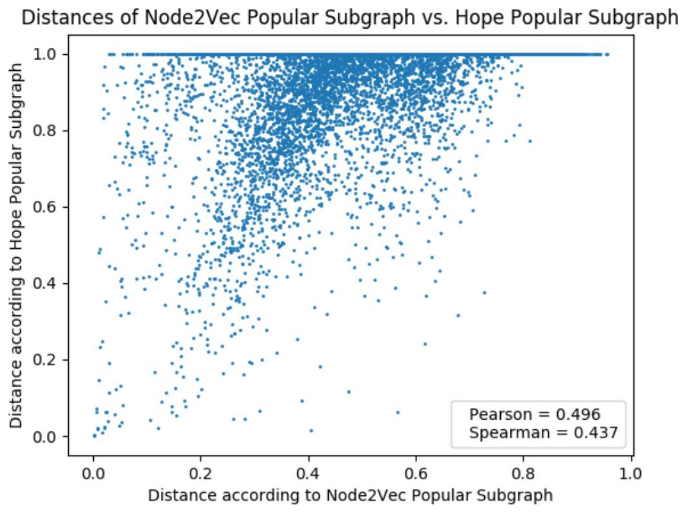
7.1.1 Node Embeddings on Popular Subgraph

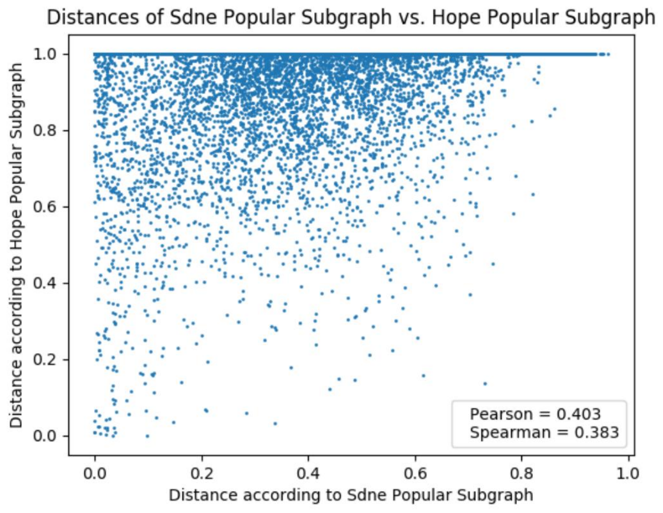


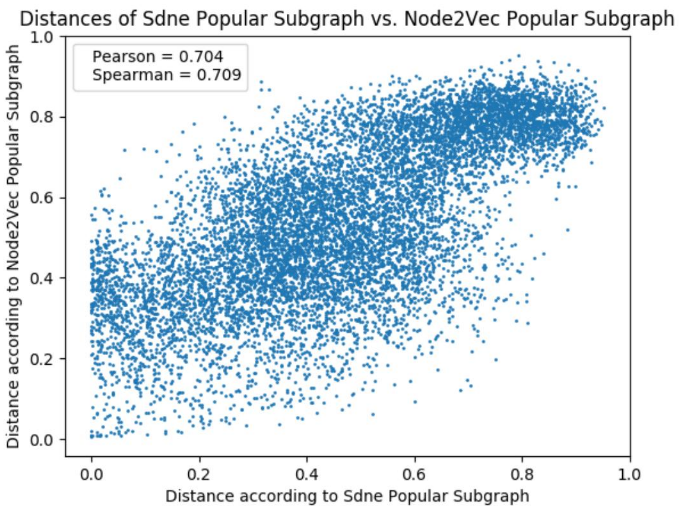


7.2. Comparison between Node Embedding Methods

7.2.1 Node Embeddings on Popular Subgraph






Distances of Sdne Popular Subgraph vs. Node2Vec Popular Subgraph
Pearson = 0.704
Spearman = 0.709
Distance according to Node2Vec Popular Subgraph
Distance according to Sdne Popular Subgraph
0.0
0.2
0.4
0.6
0.8
1.0