This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Learning Graphs for Knowledge Transfer with Limited Labels

Pallabi Ghosh Nirat Saini Larry S. Davis Abhinav Shrivastava
University of Maryland, College Park

Abstract

Fixed input graphs are a mainstay in approaches that utilize Graph Convolution Networks (GCNs) for knowledge transfer. The standard paradigm is to utilize relationships in the input graph to transfer information using GCNs from training to testing nodes in the graph; for example, the semi-supervised, zero-shot, and few-shot learning setups. We propose a generalized framework for learning and improving the input graph as part of the standard GCN-based learning setup. Moreover, we use additional constraints between similar and dissimilar neighbors for each node in the graph by applying triplet loss on the intermediate layer output. We present results of semi-supervised learning on Citeseer, Cora, and Pubmed benchmarking datasets, and zero/few-shot action recognition on UCF101 and HMDB51 datasets, significantly outperforming current approaches. We also present qualitative results visualizing the graph connections that our approach learns to update.

1. Introduction

Graph Convolution Network (GCN) based techniques have been used widely in transfer learning for tasks where labeled data is limited, *e.g.*, semi-supervised learning [24, 55] and zero-shot/few-shot learning [57, 9, 12] with zero or few samples from test classes. These approaches rely on an input graph that captures the relationships between the nodes in the graph. Given this input graph, a GCN is then used to propagate and assimilate information across the graph's nodes, obeying the relationships expressed in the graph connections. The goal of this framework is to transfer information from the training nodes to the test nodes. This is a fairly generic framework which has been adapted for a wide variety of tasks, with diverse node representations and input graphs. For example, for semi-supervised learning [24, 55], the knowledge is transferred from training samples to test samples; and the nodes represent each sample data point in the dataset and the input graph represents how these samples are related. Zero-shot learning [57, 9, 12] transfer knowledge from training classes to test classes; and the nodes represent semantic embeddings for classes (e.g., word2vec [34], sentence2vec [40]) and input graphs can come from a variety of sources (e.g., WordNet [35], NELL [2], NEIL [6]). Few-shot learning transfer knowledge between class or sample based nodes [12, 23].

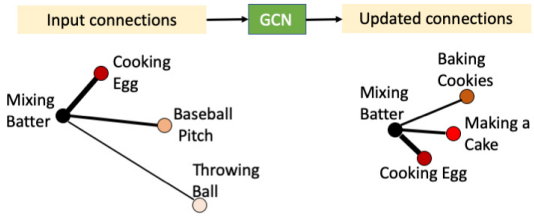

Figure 1: We use a GCN to update the input graph connections and show results for "Mixing Batter" class in zero-shot action recognition. Language based models associates "batter" to "baseball" which is rectified in the updated graph.

One of the key limitations of these GCN-based techniques discussed thus far is that the input graph structure, as captured by the adjacency matrix, is fixed. By design, the GCN-based approaches rely heavily on the input graphs, and noisy or low-quality graphs have an outsized impact on performance. In this work, we explore the adaptive learning of the input adjacency matrix over time, in conjunction with the rest of the GCN training; i.e., the losses used to train the underlying tasks (e.g., semi-supervised learning or zero-/few-shot learning) are also used to update the structure of the input adjacency matrix. We show empirically that our learned graph yields better results for downstream tasks. Our proposed approach is a straightforward algorithm to update the graph's structure by learning better node representations and using these to recompute the adjacency matrix. Note that we do not add any new network weights to learn. This is in stark contrast with other related graph learning works [21, 10], which have a separate dedicated network and special loss functions to update the adjacency matrix. Since the learned node representations, via a GCN, capture better correlations with respect to the downstream task, the resulting graph tends to be better than the input graph from an external source. One such update is illustrated in Figure 1, where we learn better connections for the class "Mixing batter". A language-based knowledge graph

Project : https://pallabig.github.io/LearningGraphsForGCN/

(KG) associates "batter" with the verb "batting" (shown as 'input'), and our approach rectifies this mistake across updates and results in more meaningful connections.

Operationalizing the straightforward approach described above has two key issues. First, updating a densely or fully-connected graph, in the absence of any other constraints, often tends to provide arbitrary updates to the structure, eventually leading to degenerate solutions (e.g., same weights for all edges). Second, if the graph connections are sparse (as is generally the case), there is no mechanism to learn to add or drop connections in the learned graph. Simple heuristics, such as fixed degree for each node can be a solution, but they tend to be sub-optimal as different nodes might have a different number of related nodes that they should be connected to. In addition, each downstream task can have domain-specific constraints on the degree of the nodes; e.g., for zero-shot action recognition, [12] observed that a fully-connected graph is detrimental to the performance and empirically determine the suitable degree. To address both the drawbacks discussed above, while obeying the domain-specific constraints, we propose to utilize a triplet loss formulation on the intermediate output nodes – i.e., the node features after our graph learning step but before the graph is passed to the GCN framework for the downstream task. Our formulation selects positive and negative neighbors for each node, and uses them to add constraints on its degree. Degenerate solutions are avoided by ensuring that negative neighbors are farther than the positive ones. Therefore, the graph learning step is trained using both the downstream task losses and the triplet loss.

In summary, our contributions are a simple learning approach that can update the input graphs for the GCN-based transfer learning framework and a triplet loss formulation that avoids degenerate solutions and allows the flexibility of degree-constraints. We demonstrate the effectiveness of our approach on semi-supervised, zero-shot, and few-shot learning setups. For semi-supervised learning, we use the generic framework [24] built on citation network datasets, like Cora, Citeseer, and Pubmed, with accompanying well-defined input graphs. For zero-shot/few-shot learning, we focus on the action recognition pipeline [12] with input KG built from sentence2vec [40] embeddings.

2. Related Works

2.1. Graph Networks

Graph networks have been used for a large number of applications like scene understanding [59, 62], segmentation [54], action recognition [13, 60] etc. Multiple works on graph neural networks and graph convolutional networks include [11, 47, 23, 8, 17, 24, 48, 64]. Spectral graph theory was introduced by Hammond *et al*. [15] and more recent works on spectral graph theory include those by Defferrard *et al*. [7] and Kipf and Welling [24]. Some of the other work in graph networks in the last decade include [20, 32, 67].

Semi-supervised learning Several works [18, 24, 25, 30, 43, 55] utilize a GCN-framework for semi-supervised learning. Such works often use citation networks datasets, like Citeseer, Cora, and PubMed [49, 39], and protein-protein interaction dataset [70] for experimentation on semi-supervised learning. Our approach utilizes the GCN framework proposed by Kipf and Welling [24] as our GCN operator and the citation network datasets as our input.

Graph Learning Networks Most related to our research are the recent works on graph learning networks for semi-supervised learning [10, 21], which proposed a new loss function to learn the edge weights in the graph. Instead of a separate network which outputs the edge weights, we take the intermediate output from the original GCN formulation and update the adjacency matrix directly. Our technique is more flexible, allowing for the update of node features and edge weights, and connections when necessary. Our method is also robust to complexity issues raised from increase in the length of the input node feature dimensions, unlike Jiang *et al*. [21]. Chen *et al*. [5] update the graph topology using a heuristics to prevent over-smoothing in GNNs. Kim *et al*. [23] applies a graph neural network model for learning the edge weights in the input graph for few-shot learning, predicting labels based on connectivity to other labeled nodes. In contrast, we build upon the GCN-framework for zero/few-shot learning, where nodes in the graph represent classes and not individual samples.

2.2. Zero/Few-shot Learning

Extensive research in the fields of zero/few-shot image classification include works by [4, 16, 26, 29, 37, 44, 45, 46, 50, 53, 56, 66]. One of these zero-shot techniques [57] uses a GCN on input KGs to transfer knowledge from seen to unseen classes. Building on this framework, [12] propose frameworks based on 3 different KGs for zero/few-shot action recognition. Due to its flexibility, use of different input graphs, and two downstream applications, we use the pipeline by [12] as our GCN-framework for experiments on zero/few-shot learning. Other research in the field of zero/few-shot action recognition include [1, 9, 14, 19, 28, 33, 36, 61, 68, 69], where [9] also uses a GCN-based system built on ConceptNet [52]. We will demonstrate that our approach outperforms the state-of-the-art approaches in this domain.

3. GCN-framework Overview

The GCN network for semi-supervised learning is a two layer network based on the spectral GCN form, introduced by [24] and given in equation 1.

$$H^{l+1} = g(H^l, A) = \sigma(D^{-1/2}AD^{-1/2}H^lW^l), \qquad (1)$$

In this equation, g is the GCN operation which takes as input H^l, output of the l^{th} layer, and A, the adjacency matrix

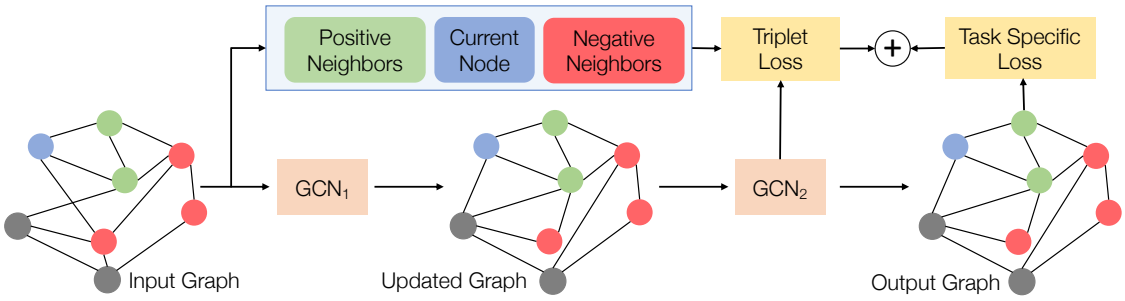

Figure 2: System overview for adaptive learning of graph connections. The input graph is passed through a GCN layer and this intermediate output is used to update the graph as well as calculate a triplet loss between the current node and the positive and negative sets. This output is then passed through another GCN network that generates outputs specific to the task at hand. The final output is used to calculate the task specific loss (like MSE loss for zero-shot learning).

with self connections. Here, D is the node degree matrix of A, W^l is the weight matrix of the l^{th} layer and σ is the activation function (e.g., ReLU) respectively. $D^{-1/2}AD^{-1/2}$ operation is called normalization of adjacency matrix from now on. We use the GCN framework for zero-shot learning proposed by [57]. A high-level overview is presented in the Algorithm 1 (black lines) and further details are provided in the supplementary.

4. Our Approach

The transfer of knowledge from training to test nodes relies heavily on the quality of the input graph. Better input inter-relationships among samples/classes lead to a better output of the GCN-based transfer learning framework. In the absence of an accompanying labelled graph (which are present in citation datasets), several studies have explored different types of KGs (e.g., [9, 12]). All GCN-based frameworks, with a few exceptions for both semi-supervised learning and zero/few-shot learning, use a fixed adjacency matrix throughout the GCN Network. However, as discussed earlier, being able to learn the adjacency matrix is both desirable and challenging. We first discuss our algorithm for updating the adjacency matrix adaptively and then present how we train this formulation.

4.1. Adaptively Updating the Adjacency Matrix

Let $\texttt{GCN}_1$ be the part of the original network that gives an intermediate output, and the rest of the original GCN be $\texttt{GCN}_2$. The output of $\texttt{GCN}_1$ is used to recalculate the adjacency matrix, where the edge weights are the cosine similarity of the output node values of $\texttt{GCN}_1$. Then, we use the new adjacency matrix as our input to $\texttt{GCN}_2$, starting from the next epoch.

More formally, let h_k^{l-1} be the output of the k^{th} node at the $(l-1)^{\text{th}}$ layer. This passes through the l^{th} convolution layer with weights W^l. Then, for each node, there is a weighted aggregation based on its neighbors, N_i, where the edge weights are represented by c_{ik} connecting nodes i and k. So the l^{th} layer output of the i^{th} node in $\texttt{GCN}_1$, h_i^l, is given by equation 2, where σ is the non-linearity (*e.g.* ReLU).

$$h_i^l = \sigma\left(\sum_{k \in N_i} c_{ik} h_k^{l-1} W^l\right), \tag{2}$$

Similarly, h_j^l is the output of the j^{th} node at the l^{th} layer. Then, the new edge weight connecting nodes i and j is given by the cosine similarity of h_i^l and h_j^l as shown in equation 3.

$$c_{ij} = \text{Normalize}\left(\frac{h_i^l \cdot h_j^l}{\| h_i^l \| \| h_j^l \|}\right), \tag{3}$$

Here Normalize is the adjacency matrix normalization used by [24]. We denote the original adjacency matrix with A^{in} and the updated one with A^{updated}. $\texttt{GCN}_1$ operates on A^{in}, whereas $\texttt{GCN}_2$ operates on A^{updated}. If we do not keep this constraint, an incorrect update to the adjacency matrix would lead to worse graph during the next update resulting in a domino effect. To aid with the optimization, we update A^{updated} every n epochs, so that $\texttt{GCN}_2$ can adapt to the new input graph. Finally, the graph adjacency is by taking a weighted average with the original input graph (update using equation 4).

$$A^{\text{updated}} = \lambda * A^{\text{updated}} + (1-\lambda) * A^{\text{in}}, \tag{4}$$

When we have good quality input graphs (*e.g.*, those in semi-supervised learning benchmarks, whose connections are based on dataset labels), we determine λ empirically. However, in cases where the input graphs are noisy (*e.g.*, those computed in [12] for action recognition), we often set $\lambda = 1$, *i.e.*, do not rely on input graph for $\texttt{GCN}_2$. Details for all setups are provided in Section 5.

4.2. Training using Triplet Loss

The original network described in Section 3 [57, 12, 24] uses classification loss and MSE (mean squared error) loss for training semi-supervised and zero-shot learning networks, respectively. To aid in updating the graph structure, we add a triplet loss. Therefore, the final framework is trained with a weighted sum of the triplet loss and the task-specific loss for increased supervision with a weight factor β. For the triplet loss, we need positive and negative sets

Algorithm 1 Overview of zero/few-shot learning

▷ black text is the algorithm from [57],
▷ blue text is our contribution.

Input: Input graph with node features (H^{feat}) and adjacency matrix (A^{in}), pretrained I3D network for test video feature extraction (f^{test}), and extraction of the final classifier layer weights for training classes (W^{cls}), # of epochs per update (n)

Output: Classification probability for all test classes (P^{test})

Networks: $\texttt{GCN}_1$ and $\texttt{GCN}_2$ are two GCN networks

procedure GCN TRAINING AND TESTING
  $A = A^{\text{in}}$
  $H^{\text{ref}} \leftarrow$ example reference node
  $P \leftarrow$ positive neighboring set for H^{ref} based on A^{in}
  $N \leftarrow$ negative neighboring set for H^{ref} based on A^{in}
  while not converged **do**
    $H^{\text{inter}} \leftarrow \texttt{GCN}_1\left(H^{\text{feat}}, A^{\text{in}}\right)$
    $H^{\text{out}} \leftarrow \texttt{GCN}_2\left(H^{\text{inter}}, A\right)$
    $H^{\text{train}} \leftarrow H^{\text{out}}$ for training classes,
    $H^{\text{P}} \leftarrow$ mean $\left(H^{\text{inter}}\right.$ for positive neighbors in $\left.P\right)$,
    $H^{\text{N}} \leftarrow$ mean $\left(H^{\text{inter}}\right.$ for negative neighbors in $\left.N\right)$
    $d^{\text{P}} = \|H^{\text{ref}} - H^{\text{P}}\|_2, d^{\text{N}} = \|H^{\text{ref}} - H^{\text{N}}\|_2$
    Loss $\leftarrow (1-\beta)L_{\text{MSE}} + \beta L_{\text{triplet}}$
    $= (1-\beta)\|W^{\text{cls}} - H^{\text{train}}\|_2 + \beta \max(d^{\text{P}} - d^{\text{N}} + \alpha, 0)$,
    where $\alpha =$ margin, $\beta =$ weighted loss parameter
    if $(\text{epoch} \bmod n) = 0$ **then**
      $A_{ij}^{\text{updated}} = \frac{H_i^{\text{inter}} \cdot H_j^{\text{inter}}}{\|H_i^{\text{inter}}\| \|H_j^{\text{inter}}\|}$,
      $A = \lambda \text{Normalize}\left(A^{\text{updated}}\right) + (1-\lambda)A^{\text{in}}$
      where $\lambda =$ weighted update parameter
  $H^{\text{out}^*} \leftarrow$ output for optimized network,
  $H^{\text{test}} \leftarrow H^{\text{out}^*}$ for testing classes,
  $P^{\text{test}} = f^{\text{test}}\left(H^{\text{test}}\right)^T$

for each node. For semi-supervised learning, each training node in the graph is a data sample with a class label. So we can use the soft-triple loss [42], which requires the number of clusters per class as a hyperparameter. We determine this empirically on the validation set and the values are provided in Section 5.

On the other hand, the positive and negative neighbors for the class nodes in zero/few-shot learning for actions need to be explicitly defined. We rely on the neighborhood of each class in the graph to initialize these sets as follows. For the positive set, we simply use the top-N(=2) neighbors closest to each node in the input KG. However, as is the case with triplet loss, defining negative set is more challenging. If we only use the farthest neighbors, the downstream task network, trained using MSE, already achieves good separation between positives and negatives and the triplet loss contribution is negligible. This implies that the triplet loss has no effect on training and the adjacency matrix can get arbitrary updates and lead to degenerate solutions. On the other hand, if the negative set is too close to the positive set, some nodes in the negative set may be constrictive and lead to large penalty which is detrimental to adjacency matrix updates. Therefore, we use the validation set to empirically select the ordinal range of the negative set classes (details in Section 6).

Finally, we take the average of the $\texttt{GCN}_1$ node outputs for the positive and negative sets to get the positive and negative vectors. Then, the triplet loss is zero when the distance between the positive vector and the current node is smaller than the distance between the current node and the negative vector by a certain margin α $(= 0.1)$. Mathematically, let H^{ref} be the output of the current reference node, and H^P and H^N be the averaged output vectors for the nodes in the positive and negative sets, respectively. Then, the distance between H^{ref} and H^P (or H^N) is represented as d^P and d^N (equation 5); and the triplet loss, L_{triplet} is calculated using equation 6.

$$d^P = \|H^{\text{ref}} - H^P\|_2, \quad d^N = \|H^{\text{ref}} - H^N\|_2, \tag{5}$$

$$L_{\text{triplet}} = \max(d^P - d^N + \alpha, 0), \tag{6}$$

5. Experimental Setup

Datasets. We use Citeseer, Cora, and Pubmed datasets [49, 39] for experiments on semi-supervised learning, where nodes are documents and edges are citations. There are 6 classes in Citeseer, 7 in Cora, and 3 in Pubmed. We use the same train, test, and validation splits as [24, 63]. For zero/few-shot action recognition, we use Kinetics [22] for pre-training our feature extraction model and as 'additional nodes' in the constructed graph (refer to [12] for details). We use UCF101 [51] and HMDB51 [27] as our evaluation datasets. Kinetics has 400 classes; UCF101 has 101 classes, out of which 23 are used for test and 78 for training; and HMDB51 has 51 classes, out of which 12 are used for test and 39 for training. These dataset splits are the same as the ones used by [12]. We make 10 random subsets of c classes among test classes and we average the performance on all 10 subsets for validation purposes. We then select the model with best performance on this validation set and report results on the entire test set. c is 20 for UCF101 dataset and 10 for HMDB51 dataset. More details about the datasets are in the supplementary.

Input graphs. Next we discuss the input graphs which we study in this work. For semi-supervised learning, we use the same graphs as used by [24], based on the Citeseer, Cora, and Pubmed datasets as discussed before. For zero/few-shot action recognition, we use the input KGs as used by [12]. We summarize these KGs below (and refer the reader to [9, 57] for discussion on other types of KGs for these tasks).

We use the action-KG (or KG1 from [12]) for zero-shot learning; referred to as $\texttt{A-KG}$ throughout this work. $\texttt{A-KG}$ nodes use sentence2vec [40] representations of action phrases, and the edge weight in the adjacency matrix is the

Table 1: We compare accuracy of our technique to various state-of-the-art techniques for semi-supervised learning using Cora, Citeseer, and Pubmed datasets (Higher is better). We provide our GCN baseline implementation in PyTorch (GCN*). The † in Pubmed for GLNN stands for downsampled input data. We get the best performance for both Citeseer and Pubmed datasets. For Cora, our absolute performance is worse compared to GLCN, but relative performance improvement from graph learning compared to respective GCN baselines is better.

Method	Cora	Citeseer	Pubmed
SemiEmb [58]	59.0%	59.6%	71.7%
DeepWalk [41]	67.2%	43.2%	65.3%
ICA [31]	75.1%	69.1%	73.9%
Planetoid [63]	75.7%	64.7%	77.2%
Chebyshev [7]	81.2%	69.8%	74.4%
GCN [24]	81.5%	70.3%	79.0%
MoNet [38]	81.7%	—	78.8%
GAT [55]	83.0%	72.5%	79.0%
GLNN [10]	83.4%	72.4%	76.7%†
GCN+GDC [25]	83.6%	73.4%	78.7%
H-GCN [18]	84.5%	72.8%	**79.8%**
GLCN [21]	**85.5%**	72.0%	78.3%
GCN*	80.0%	72.0%	77.8%
Ours	83.6%	**74.3%**	**79.8%**

cosine similarity between the corresponding node features. For few-shot learning, we use the visual feature-based KG (or KG3 from [12]); referred to as V−KG throughout. These visual features are extracted using an I3D [3] network for five random samples per class, and these features are averaged to generate node features for V−KG. Similar to A−KG, the edge weights are based on cosine similarity of node features. Finally, we also show results using a KG based on verbs and nouns (or KG2 from [12]), referred to as VN−KG. Both verbs and nouns are extracted from the action phrases, and their sentence2vec are used as the node features for two separate KGs. Edge connections are again based on cosine similarity of node features. Following [12], we also show results for combinations of these KGs under different settings and demonstrate that our approach can generalize to different input graph formulations.

Pipeline. For semi-supervised learning, we use a two-layer network where the intermediate output is used to update the graph connections. The learning rate is 0.005 for all three datasets. We empirically determined the number of cluster per class for soft-triple loss to be 2 for Pubmed and Cora and 10 for Citeseer. The rest of the hyperparameters for Soft-triple loss are the same as [42]. The λ parameter in equation 4 is 0.8 for all datasets.

For zero/few-shot action recognition, we use I3D [3] pre-trained on Kinetics and only finetune the last classifier layer on the downstream datasets (separately on UCF101 and HMDB51) until convergence. We use one layer for $\texttt{GCN}_1$ and five layers for $\texttt{GCN}_2$, where the last layer is for the fusion GCN for setups using multiple KGs. The learning rate is 0.001 for all experiments except few-shot learning for UCF101 where the learning rate is 0.00005. To calculate MSE loss, we use a weighted summation of the loss based on the specific dataset nodes (HMDB51 and UCF101) and Kinetics nodes. λ from equation 4 is 1.0 for all zero-shot/few-shot KGs except for HMDB51 A−KG +V−KG +VN−KG where it is 0.5 (determined empirically). For HMDB51 A−KG, we use the final output of $\texttt{GCN}_2$ to calculate A^{updated} (as opposed to $\texttt{GCN}_1$) and no triplet loss. More details on pipelines are in supplementary.

Table 2: Ablation comparing accuracy for Pubmed validation data for different values of weighted averaging between input and updated adjacency matrix, *i.e.*, λ from equation 4.

λ	1.0	0.8	0.6	0.4	0.2
Pubmed	76.2%	**80.6%**	79.8%	79.4%	79.0%

Table 3: Comparison of our results with Ghosh *et al*. [12] for UCF101 and HMDB51 datasets. We do better for all input KG configurations: A−KG, V−KG, and A−KG +VN−KG +V−KG. The metric is mean accuracy (Higher is better).

Input KG	UCF101		HMDB51	
	[12]	**Ours**	[12]	**Ours**
A−KG	49.14	**53.27**	38.01	**41.05**
V−KG	57.04	**60.57**	45.07	**48.07**
{A+VN+V}−KG	64.24	**65.49**	47.69	**49.17**

6. Quantitative Results

6.1. Semi-supervised Learning

We show results on semi-supervised learning for Cora, Citeseer, and Pubmed datasets in Table 1. We compare against multiple state-of-the-art methods, including graph learning methods like GLCN [21] and GLNN [10]. The GCN* is our implementation of GCN [24] in PyTorch environment with 256 intermediate channels and we get slightly differing baseline results. Since our approach builds on this baseline, we report these results as well, to do a direct comparison. Our approach outperforms all others on both Citeseer and Pubmed datasets. GLCN does best on the Cora dataset, but their result from GCN baseline is 82.9% and after graph learning their result is 85.5%, so relative performance gain is 2.6%. Our baseline GCN result is 80.0% and after graph learning our result is 83.6%, which implies a 3.6% relative gain.

Ablation analysis. We report results using different values of λ from equation 4 on Pubmed validation set in Table 2. We observe that $\lambda = 0.8$ achieves best performance and use this in all semi-supervised experiments. Additional experiments and results are provided in the supplementary.

Table 4: Improvements using triplet loss or updating adjacency matrix (on UCF101 V−KG) one at a time and then both together. Metric is mean accuracy (Higher is better).

KG (UCF101)	triplet loss	update A	mean accuracy
V−KG			57.04
V−KG	✓		58.57
V−KG		✓	59.39
V−KG	✓	✓	60.57

Table 5: Ablation showing performance of UCF101 A−KG with varying number of epochs per update of adjacency matrix. The metric used in mean accuracy (Higher is better).

# epoch per update	10	20	30	40	50
UCF101 A−KG	52.89	50.17	**54.41**	50.72	48.71

Table 6: Ablation showing performance of UCF101 A−KG with different negative set class index ranges for triplet loss. The metric used in mean accuracy (Higher is better).

Triplet loss '-'ive set	5-10	15-20	9-11	9-14	9-19
UCF101 A−KG	49.22	48.74	51.51	**54.41**	49.27

6.2. Zero-shot/Few-shot Action Recognition

We compare with the results of [12] for zero and few-shot action recognition in Table 3. These results are for both UCF101 and HMDB51, using three different input graph configurations: A−KG, V−KG, and A−KG +VN−KG +V−KG. For both UCF101 and HMDB51, the metric is mean accuracy, which averages the classwise accuracy over all classes. It can be observed that our approach of updating the graph structure during training significantly outperform [12].

Ablation analysis. We first analyze the contribution of our approach to update adjacency matrix A and the triplet loss formulation in Table 4 (on UCF101 using V−KG). We show that both contributions are effective individually and are complementary to each other. Next, we study the two hyperparameters associated with our proposals – (a) varying the number of epochs before updating A (n), and (b) different ordinal ranges for negative classes for the triplet loss. The results are presented in Table 5 and Table 6 respectively. For these, we use the mean of 10 runs of randomly chosen subsets of 20 test classes. We get the best performance at 30 epochs per update and negative set range of $[9, 14]$.

Comparison with State-of-the-art Zero-shot Learning. Finally, we compare against state-of-the-art approaches for zero-shot learning. Note that we cannot do a similar comparison for few-shot learning because we do not follow the episodic learning pipeline as the other papers. In particular, we compare against ESZSL [46], DEM [66], TS-GCN [9], GLCN [21], [12], [33], UR [69], Action2vec [14], and TARN [1]. Please refer to the supplementary for more details on these approaches. We evaluate on both UCF101 and HMDB51 datasets and report mean accuracy. We provide results for the entire test sets, for both UCF101 and HMDB51, in Table 7a and on the 20/81 split used by previous papers in Table 7b. For the latter, we randomly choose 20 classes from UCF101 test classes 10 times and average the output performance and report the average scores over all runs. We outperform the state-of-the-art techniques in all three cases, further emphasizing the importance of updating the graph structure for zero-shot approaches.

Table 7: Comparison with the state-of-the-art zero-shot action recognition results for both UCF101 and HMDB51 datasets. Metric used is mean accuracy (Higher is better). * implies our implementation of their algorithm. In (a) we compare over the entire test set. In (b) we randomly choose 20 classes from UCF101 test set 10 times and average the output to replicate the 20/81 split reported by previous works ([14, 1, 33, 69]).

Method	UCF101 23-78 split	HMDB51 12-39 split
[46]	35.27	34.16
[66]	34.26	35.26
[9]	44.5	-
[21]	49.96*	37.06*
[12]	50.13	40.77
Ours	**53.27**	**41.05**

(a)

Method	UCF101 20-81 split
[14]	36.5
[1]	42.7
[33]	51.2
[69]	53.8
Ours	**54.4**

(b)

7. Discussion

Class-wise performance. In Figure 3, we show class-wise performance comparison between our approach and [12] for UCF101 test classes with A−KG and V−KG as input. For zero-shot learning using A−KG, our technique outperforms the baseline for most classes (12 out of 23) like "Apply eye makeup", "Apply lipstick", "Billiards", "Nunchucks" and "Playing Daf". In some cases (7 out of 23), like "Still rings", "table tennis shot", "uneven bars", our updated graph performs worse. An explanation for the "Still rings" class is discussed later (in Figure 5). For few-shot learning using V−KG, we perform better on 12 and worse on 6, and similar to fixed input graphs on 5 classes.

Qualitative results for graph updates. Figure 4 shows the graph connections among 57 selected nodes for UCF101 and Kinetics based on A−KG. These nodes are the neighbors for the selected 8 test classes (class names shown in red). The edge weights are represented by the edge colors from the color bar. In this color bar, blue represents lowest edge weights and red represents the highest with green and yellow somewhere in the middle. The visualization on the left is for the input adjacency matrix, the center is after the first update at 30^{th} epoch, and the right is after the second update at 60^{th} epoch. There are many examples where the update improves the input KG, but due to space constraints, we only discuss one specific node here (for more

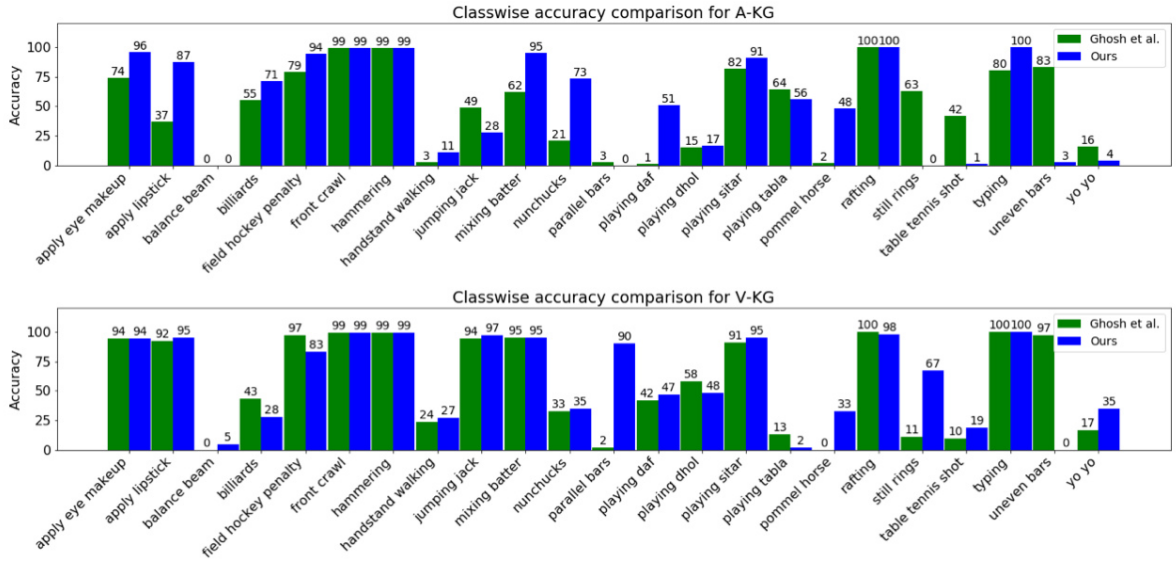

Figure 3: Class-wise comparison of accuracy for 23 UCF101 test classes using A−KG and V−KG as input for zero-shot and few-shot learning, respectively, between our approach (blue) and baseline [12] (green). In both cases, for majority of classes, we either beat or maintain the baseline performance. Best viewed in digital.

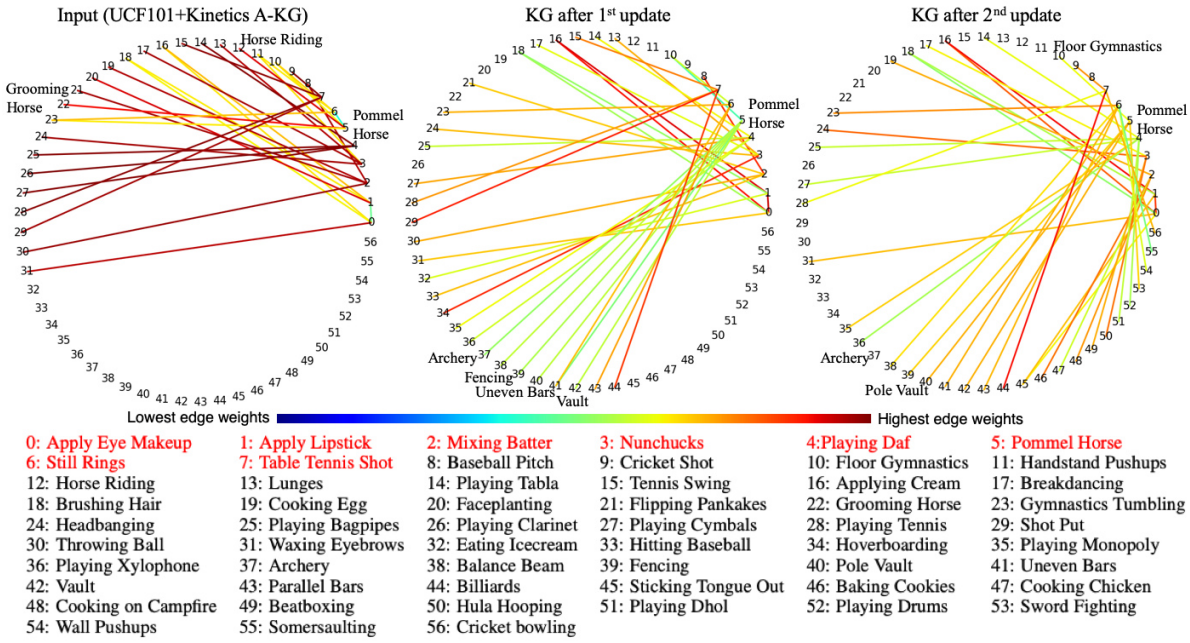

Figure 4: We plot the adjacency matrix connections for UCF101+Kinetics A−KG input and show the following two updates of the adjacency matrix. We plot only a sub-graph due to space limitations. We chose 8 test classes (class names shown in red) and display all their connections in the KG. The edge colors show the weight of the connection. There are multiple regions where we can see improvements after first and second update. Best viewed in digital.

examples, please refer to the supplementary). For "Pommel horse" (a gymnast action), we see multiple wrong connections in the input KG because this KG is based on word embeddings. Due to the presence of the term "horse" in the name, it associates "Pommel horse" with "Grooming horse" and "Horse riding". After the first update, these connections are removed, but it creates connections to classes like "Archery" and "Fencing" which are not correct. It has some correct connections, like ones to "Vault","Uneven bars"; but the weights are low due to normalization of various connections. After the second update, a lot of these connections (like "Archery") are removed and weight increases to connections like "Floor Gymnastics" and "Pole Vault". So, overall the KG improves after each update.

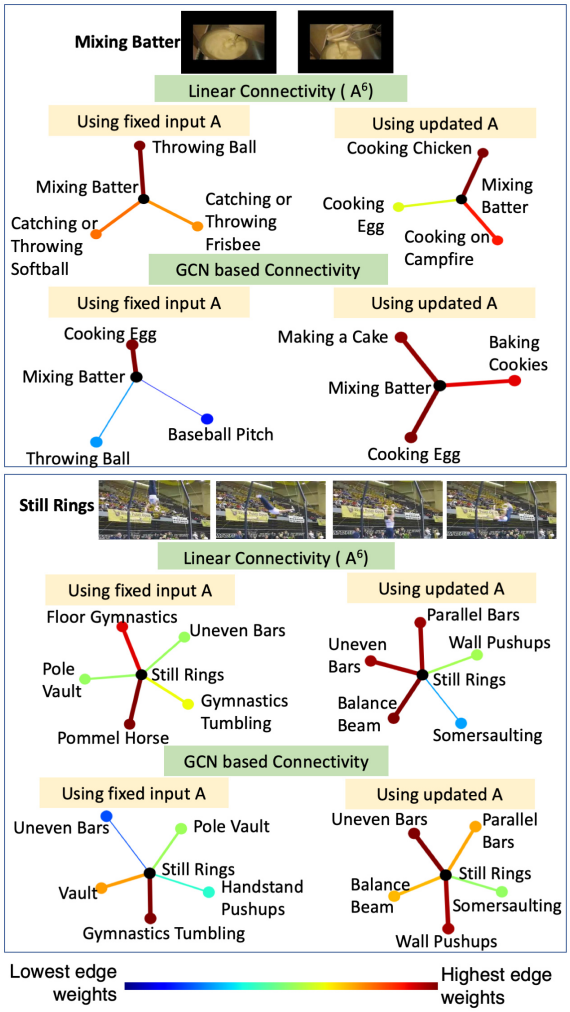

Figure 5: We show the connections of A^L where A is the adjacency matrix and L is the number of layers in the GCN (linear connectivity). We also show connections after passing through the non-linear GCN network (GCN-based connectivity) for "Mixing Batter" and "Still Rings" classes. For both, we show the top-K connections using fixed input A as well as updated A. The edge color (based on the color bar) and the width of the connections represent edge weights. (larger width $\propto$ higher weights). For "Mixing Batter" the performance becomes better while for 'Still Rings" the performance becomes worse after A is updated.

Visualizing important connections. Figure 5 shows important graph connections with respect to the GCN network. A GCN has multiple layers and each layer involves convolution, adjacency matrix multiplication, and non-linearity. The linear equivalent of this system is A^L where L is the number of layers in the GCN and A is the adjacency matrix. We display the top-N neighbors in A^L with A from input and updated adjacency matrix for two test classes: "Mixing batter" and "Still Rings" in Figure 5 labeled as linear connectivity. We also visualize the closest neighbors after the GCN operation when our method has updated the adjacency matrix. We propose a new visualization technique, inspired by [65] which blocks out portions of input images to understand ConvNets. If the GCN operation is represented by G and the input to the GCN is the KG K, the original output probability is given by $O = G(K) \times f_{\text{vid}}$, where f_{vid} is the feature vector of a video in class C. Next, we modify K to $K - n_i$ by removing connections to one input node n_i and the new output is given by $O_{\text{new}} = G(K - n_i) \times f_{\text{vid}}$. Then, the importance of connection between node n_i and the correct output class node C is given by equation 7, where the higher the change, the more important is the connection.

$$|O - O_{\text{new}}| = |\left(G(K) - G(K - n_i)\right) \times f_{\text{vid}}|, \quad (7)$$

We show GCN based connectivity, extracted using this approach, in Figure 5 for the two classes, "Mixing batter" and "Still Rings", using input and updated adjacency matrix. The edge color and widths represent importance or weight of the connection (higher width implies higher edge weight). The updated adjacency matrix based connectivity becomes better for "Mixing batter" and worse for "Still rings". For "Mixing batter" the word embedding based KG associates "batter" with "baseball" classes like "Throwing Ball", "Baseball Pitch", etc. whereas our updated KG correctly associates "Mixing batter" with cooking classes like "Cooking Egg" and "Making a cake". On the other hand for "Still rings" the original KG has "Pole Vault" and "Gymnastics tumbling" as some of the top neighbors whereas the updated KG has "Balance beam", "Uneven bars", and "Parallel bars" as the top neighbors. The problem is that these are more similar to the "Pommel horse" test class and so most "Still rings" videos are predicted as "Pommel horse" after the update. One possible solution to this problem in the few shot scenario is selective updates to the adjacency matrix where neighbors of only those nodes are updated which results in better performance.

8. Conclusion

We propose an approach to update adjacency matrices in GCN-based formulation adaptively using a triplet loss that can obey degree constrains in the graph. We analyze and qualitatively demonstrate how the graph connections change with updates. Inspired by prior work on ConvNets, we visualize importance of individual connections after the GCN operation as well as at the input. We outperform most state-of-the-art techniques on multiple benchmarking datasets for semi-supervised learning and zero/few-shot action recognition by a significant margin.

Acknowledgements This work was supported by the Air Force, via Small Business Technology Transfer (STTR) Phase I (FA865019P6014) and Phase II (FA864920C0010), and Defense Advanced Research Projects Agency (DARPA) SAIL-ON program (W911NF2020009).

References

[1] Mina Bishay, Georgios Zoumpourlis, and Ioannis Patras. Tarn: Temporal attentive relation network for few-shot and zero-shot action recognition, 2019. 2, 6

[2] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 1

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 5

[4] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016. 2

[5] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020. 2

[6] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416, 2013. 1

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016. 2, 5

[8] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015. 2

[9] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8303–8311, 2019. 1, 2, 3, 4, 6

[10] Xiang Gao, Wei Hu, and Zongming Guo. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020. 1, 2, 5

[11] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017. 2

[12] Pallabi Ghosh, Nirat Saini, Larry S. Davis, and Abhinav Shrivastava. All about knowledge graphs for actions, 2020. 1, 2, 3, 4, 5, 6, 7

[13] Pallabi Ghosh, Yi Yao, Larry Davis, and Ajay Divakaran. Stacked spatio-temporal graph convolutional networks for action segmentation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 576–585, 2020. 2

[14] Meera Hahn, Andrew Silva, and James M Rehg. Action2vec: A crossmodal embedding approach to action learning. *arXiv preprint arXiv:1901.00484*, 2019. 2, 6

[15] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. 2

[16] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017. 2

[17] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[18] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. Hierarchical graph convolutional networks for semi-supervised node classification. *arXiv preprint arXiv:1902.06667*, 2019. 2, 5

[19] Mihir Jain, Jan C van Gemert, Thomas Mensink, and Cees GM Snoek. Objects2action: Classifying and localizing actions without any video example. In *Proceedings of the IEEE international conference on computer vision*, pages 4588–4596, 2015. 2

[20] Feng Ji, Jielong Yang, Qiang Zhang, and Wee Peng Tay. Gfcn: A new graph convolutional network based on parallel flows. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3332–3336. IEEE, 2020. 2

[21] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11313–11320, 2019. 1, 2, 5, 6

[22] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 4

[23] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019. 1, 2

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 2, 3, 4, 5

[25] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, pages 13354–13366, 2019. 2, 5

[26] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2452–2460, 2015. 2

[27] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011. 4

[28] Sai Kumar Dwivedi, Vikram Gupta, Rahul Mitra, Shuaib Ahmed, and Arjun Jain. Protogan: Towards few shot learn-

ing for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 2

[29] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3):453–465, 2014. 2

[30] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*, 2018. 2

[31] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 496–503, 2003. 5

[32] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016. 2

[33] Pascal Mettes and Cees G. M. Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4453–4462, 2017. 2, 6

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 1

[35] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, (11):39–41, 1995. 1

[36] Ashish Mishra, Vinay Kumar Verma, M Shiva Krishna Reddy, S Arulkumar, Piyush Rai, and Anurag Mittal. A generative approach to zero-shot and few-shot action recognition. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar 2018. 2

[37] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. 2

[38] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017. 5

[39] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, 2012. 2, 4

[40] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL 2018-Conference of the North American Chapter of the Association for Computational Linguistics*, 2018. 1, 2, 4

[41] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014. 5

[42] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *IEEE International Conference on Computer Vision, ICCV 2019*, 2019. 4, 5

[43] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. Semi-supervised user geolocation via graph convolutional networks. *arXiv preprint arXiv:1804.08049*, 2018. 2

[44] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. 2

[45] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proceedings of 6th International Conference on Learning Representations ICLR*, 2018. 2

[46] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015. 2, 6

[47] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008. 2

[48] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018. 2

[49] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 2, 4

[50] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017. 2

[51] Khurram Soomro, Amir Roshan Zamir, and M Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012. 4

[52] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press, 2017. 2

[53] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. 2

[54] Gusi Te, Wei Hu, Zongming Guo, and Amin Zheng. Rgcnn: Regularized graph cnn for point cloud segmentation. *arXiv preprint arXiv:1806.02952*, 2018. 2

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 1, 2, 5

[56] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning, 2016. 2

[57] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018. 1, 2, 3, 4

[58] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pages 639–655. Springer, 2012. 5

[59] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419, 2017. 2

[60] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[61] Hongtao Yang, Xuming He, and Fatih Porikli. One-shot action localization by learning sequence matching network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[62] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018. 2

[63] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 4, 5

[64] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018. 2

[65] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 8

[66] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2021–2030, 2017. 2, 6

[67] Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3016–3025, 2019. 2

[68] Linchao Zhu and Yi Yang. Compound memory networks for few-shot video classification. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[69] Yi Zhu, Yang Long, Yu Guan, Shawn Newsam, and Ling Shao. Towards universal representation for unseen action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9436–9445, 2018. 2, 6

[70] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017. 2