Incorporating Visual Grounding In GCN For Zero-shot Learning Of Human Object Interaction Actions

Anonymous CVPR Workshop submission

Paper ID 59

Abstract

GCN-based zero-shot learning approaches commonly use fixed input graphs representing external knowledge that usually comes from language. However, such input graphs fail to incorporate the visual domain nuances. We introduce a method to ground the external knowledge graph visually. The method is demonstrated on a novel concept of grouping actions according to a shared notion and shown to be of superior performance in zero-shot action recognition on two challenging human manipulation action datasets, the EPIC Kitchens dataset, and the Charades dataset. We further show that visually grounding the knowledge graph enhances the performance of GCNs when an adversarial attack corrupts the input graph.

1. Introduction

Graph Neural networks are defacto in learning representations for graphical data. Graphs are useful when representing scenes [2,20], actions [13], and human-object interaction [4, 30] activities. Because of the ability to encompass the scene's structure with either attributes or symbols, knowledge graphs are a step toward explainable AI [29]. In this paper, we utilize knowledge graphs to solve one of the challenging problems of recognizing human-object interaction activities under a zero-shot learning setting.

Consider a simplified formulation with two graphs typically used in zero-shot settings: an input language graph and an output vision graph that we are interested in learning. In the language graph, each node represents an action obtained from word embeddings. The language graph's edge weights encode the semantic distance between actions. The output vision graph has the same amount of nodes as the language graph and corresponds to the visual embeddings. Message passing methods like graph convolution and graph attention have been used to learn the mapping between the known category nodes of the input graph and output graph to fill in information for the nodes in the output graph for which we don't have examples. In both the formulations of graph convolution networks (GCN) [18] and graph attention networks (GAT) [31], the structure of the input graph, along with its edges, is fixed.

One of the challenges of using GCN or GAT is that message passing relies only on the input graph semantics derived from language embeddings, completely being obscure to the semantics of the visual graph we are interested in predicting. However, we know that the word embeddings and visual embeddings need not be aligned semantically. Furthermore, word embeddings are prone to noise depending on the text corpora used to obtain them. The predicted visual graph from GCN carrying language semantics and the actual visual graph made using visual embeddings will differ owing to the domain gap between the language and the visual embedding. Moreover, we can't directly utilize an input graph made from visual embeddings as we only have information about the training class nodes in a zero-shot learning setting. This raises the question of the ideal input graph for zero-shot learning. Furthermore, using the above formulation to recognize human object interaction actions under a zero-shot learning setting has another challenge: the semantic gap between videos of unseen novel test classes and the seen training classes. In this paper, we focus on introducing solutions to these issues from the perspective of knowledge graphs.

We propose two methods to visually ground the language graph to address these challenges. In the first method, we modify the input language graph by changing the weights of the edges to reflect the visual semantics of the visual graph. Specifically, we modify the language adjacency matrix by adjusting its weights according to the weights from the vision graph. We use two different concepts to define shared concepts to group actions. We call this process as inhibitory and excitatory feedback for message passing in graph convolutions. We experimentally demonstrate that the modification of edge weights leads to improvements in the task of zero-shot action recognition on the Charades and EPIC-kitchens datasets.

In the second method, we integrate the visual graph in learning the GCN. Since the test classes are unknown, we first estimate the adjacency matrix of the visual graph for all the classes by using the adjacency matrix of the language graph. We then modify the GCN graph propagation rule to integrate the visual adjacency matrix. We experimentally demonstrate the usefulness of the visual adjacency matrix under normal conditions as well as when the language graph has been adversarially attacked.

In summary, our contributions are:

1. We highlight the limitation of using only the language graph in zero-shot learning, ignoring the semantics of the visual graph. We propose two methods to visually ground the language graph to tackle this.

2. Proposed methods are simple and easy to integrate with existing GCN message passing methods.

2. Related Work

2.1. Graph Neural Networks

Graph convolutional networks (GCN) have been introduced by Kipf and Welling [18] for the semi-supervised classification of graph data. The core of the GCN is the graph propagation rule, which intuitively does feature aggregation of neighboring nodes. The importance of a neighboring node in learning the feature of a node is given by the edge weight connecting the node and its neighboring node. The GCN of Kipf and Welling [18] uses an adjacency matrix to represent the edge weights and is fixed. Thus, its performance relies mostly on the accuracy of the adjacency matrix and to an extent the features of nodes. Graph Attention networks (GAT) [31], on the other hand, implicitly learn the importance of a node with its neighboring node through self-attention over the features of nodes. There have been several improvements [11, 14, 17, 19] over the original GCN and GAT, but the underlying graph propagation in networks still relies on the input graph. Elinas et al. [5] introduced structural learning of the GCN adjacency graph assuming a transductive setting. However, they assume that all training classes are independent of each other when conditioned on features and an adjacency matrix. For manipulation actions, we can't assume the classes are independent of each other. Many actions can be grouped based on attributes or shared concepts. Our method of using grouping of actions in GCNs shows that there is indeed benefit in assuming the interdependence of action classes. Furthermore, we build an inductive model to estimate the visual adjacency matrix as feedback from the visual graph. We do not assume any probabilistic model of the adjacency graph or the features. While updating the GCN, Ghosh et al. [10] add a triplet loss between positive and negative neighbors in the input language graph but still ignore the visual domain semantics to obtain the triplets. We differ from all previous methods by introducing another aspect to graph propagation. We introduce direct feedback from the output graph while doing graph propagation. The additional information from the output graph in the form of an adjacency matrix provides the relationship between the nodes in the output domain

2.2. GCNs For Zero-shot Action Recognition

Previous approaches for zero-shot action recognition include [7–9, 12, 16, 21, 23, 25, 32]. Many studies rely on attributes for zero-shot recognition. However none of the previous works define attributes that are effective in large challenging human object manipulation datasets like the Charades dataset and EPIC-Kitchens55 dataset. Jain *et al.* for their system Objects2action [12] use objects present in actions as attributes, and Liu *et al.* [21] use attributes for classification of whole body actions. The methods in [9, 10, 16] are graph based using word embedding only. Our approach falls under the umbrella of hybrid approaches as it combines the word embedding with visual feedback and the visual adjacency matrix. In our experiments (Sec. 4) we compare against Ghosh *et al.* [9], who fuse three graphs to represent actions, objects and verbs and perform knowledge transfer. In comparison, our knowledge graph is much simpler consisting of just one graph instead of a fusion of three and relies on simply calculating the cosine similarities between the word embeddings. To the best of our knowledge, none of the previous works consider addressing the domain gap between visual and language graphs. A few studies use excitatory and inhibitatory weight changes. [11] exploited multidimensional edge features. Jia *et al.* [14] introduced squeeze excitation of node channels completely in a feed-forward way of message passing by learning weights to excite the nodes. We differ from [11, 14] by incorporating excitation or inhibition based on grouping of actions.

3. Method

3.1. Architectural Overview

Here we describe the high level formulation of the zero-shot action recognition. This forms our GCN baseline method referred to in the experimental section 4. Let the number of training classes be S, and the number of test classes be U.

Figure 1 shows the architecture of the network at training time. The network consists of input language graph on the left, which represents all the actions in form of language vector embeddings, and a vision graph, on the right, whose nodes are the embeddings of actions from the visual domain.

The language graph nodes are initialized with the GloVe [24] word embeddings, and the value of the weights on the edges originally are the cosine similarity distances

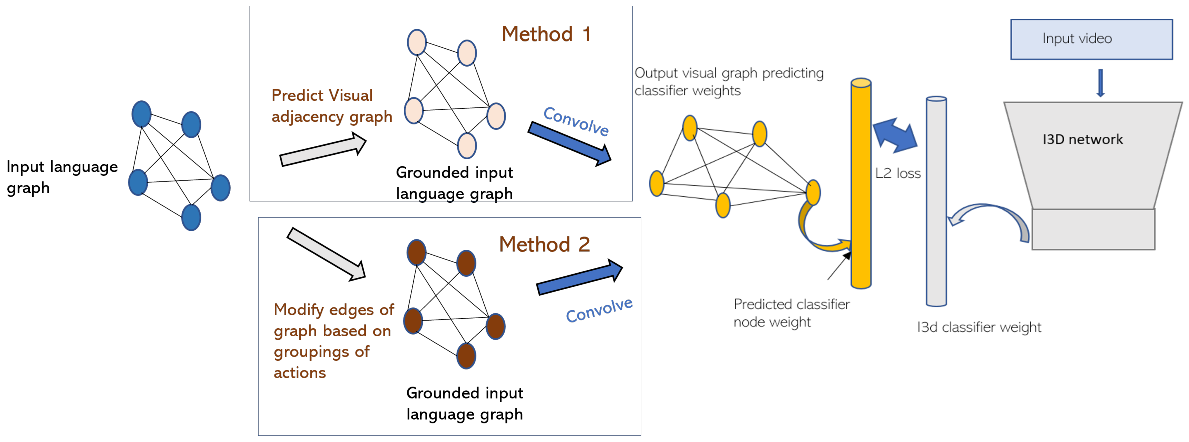
Input language graph
Method 1
Predict Visual adjacency graph
Grounded input language graph
Convolve
Method 2
Modify edges of graph based on groupings of actions
Grounded input language graph
Convolve
Output visual graph predicting classifier weights
L2 loss
Predicted classifier node weight
I3d classifier weight
Input video
I3D network

Figure 1. **Overview of our zero-shot action recognition approach:** The Input graph is visually grounded before being passed to GCN to predict the output visual graph. Visual grounding refers to either the weight adjustment method (Method2) or the visual adjacency graph method (Method1) described in sections 3.5 and 3.6, respectively.

between the GloVe vectors.

3.2. Graph Convolution Network

Consider a graph G with N nodes. Let its adjacency matrix be A of dimension $N \times N$. The graph propagation rule from the formulation of GCN by Kipf and Welling [18] is given by

$$H_{l+1} = f(A \times H_l \times W_l) \tag{1}$$

In equation 1 , H_{l+1} is the GCN output matrix at level $l+1$ of dimension $N \times k$, and H_l is the input feature matrix at level l of dimension $N \times d$. W_l is the GCN weight matrix of dimension $d \times k$ that we are learning. Here k is the output feature dimension. A is the adjacency matrix. f is a non-linear function which in our case is a leaky relu activation layer at the end of each convolution operation. We use the same normalization to the adjacency matrix as in [15]. After normalization of A the modified propagation rule is given by equation 2.

$$H_{l+1} = f(D^{-1} \times A \times H_l \times W_l), \tag{2}$$

with D denoting the diagonal matrix of the adjacency matrix.

3.3. Training

The graph convolutional network is trained to learn the nodes of the output vision graph. This is done as follows: In a pre-processing step, we train an I3d [1] classifier using the videos of action categories in the training set. We then utilize the final classifier layer weights of the I3d classifier to train the GCN. Let us denote $W_{classifier}$ of dimension $S \times k$ as the final classifier layer weight.

During training, the predicted embeddings $W_{predtrain}$ of the training nodes of the output vision graph are matched with the I3d classifier weights $W_{classifier}$ obtained earlier in the pre-processing step. We ensure that the node representing say for example "put " action class is matched with the I3d classifier layer weight representing the "put" action class. Here $W_{predtrain}$ of dimension $S \times k$ represents the visual embeddings of the training nodes of graph. Specifically $loss$,

$$loss = ||W_{predtrain} - W_{classifier}||^2 \tag{3}$$

is minimized between the predicted weights and the trained I3d classifier weights.

3.4. Testing

At test time, given a test video, the I3d features f_{test} of dimension k are extracted. f_{test} is compared against all the nodes of the output GCN representing testing classes to obtain the action category of the video. Let $W_{predtest}$ of dimension $U \times k$ represent the visual embeddings of U number of test class nodes. Using the symbol T to denote transpose of a matrix, i.e., f_{test}^T, the predicted class Y is derived from equation 4

$$Y = softmax(W_{predtest} * f_{test}^T) \tag{4}$$

3.5. Visual Grounding Using Grouping Of Actions

Since we don't have access to all the nodes of output graph at training time, we indirectly approach using the output graph's semantics. We form groups of actions using a cognitive concept that they share in common. Using super-categories and creating groups of actions has roots in defining an ontology of actions [33]. In our case, we define a higher-level representation of manipulation actions by considering the geometric transformation performed on the manipulated object, topological changes on the scene, or the type of object motion involved. The grouping of actions follows cognitive reasoning and can be extended to any human-manipulation activities dataset.

3.5.1 Defining the grouping of actions using shared cognitive concept

We group actions according to two criteria. First, according to the object's change of state that the manipulation action induces. Second, based on types of object motion involved.

Here we list the actions in the EPIC Kitchens dataset having that shared concept.

- Decrease in size: squeeze, press, crush, fold, knead
- Increase in size: open, stretch
- Add (to the scene): put, pour, insert, fill, add, apply, spray.
- Remove (from the scene): take, remove, empty, scoop, filter
- Separate: Cut, break, peel, divide.

We can also group actions

- complex motion: mix, shake
- translational motion: move, put, insert,take,remove.
- rotational motion: turn, scoop
- no-motion: grasp,hold.

Charades dataset, however, includes not just the manipulation actions but actions like “walk”, “sit”, “watch” for which object state changes don't directly apply. Another interesting feature of the Charades dataset is that action classes are a combination of both verbs and objects. Examples of some actions from the Charades dataset are as follows, “Putting something on a table”, “watching a book”, “watching a window”, ”smiling at a mirror”, ”watching something/someone/themselves in a mirror”. For the Charades dataset, we form groups based on the common object involved in an action. We hypothesize that actions involving the same objects tend to have closer RGB visual features based on a discussion in [27] that shows that errors are more for classifiers with actions with the same object and different verbs. Furthermore, our goal is not to create super categories of actions that classify all actions. But instead, come up with a logical way of grouping actions that can be easily generalized in any object manipulation dataset. While there are many ways of forming groups, we have seen from the experiments that our algorithm is resilient to criteria of grouping of actions.

3.5.2 Modifying the adjacency matrix based on grouping of actions

Here we describe how we utilize the grouping of actions to modify the adjacency matrix obtained from language embeddings. Let $A_{language}$ represent the adjacency matrix obtained from word embeddings. $A_{language}[i, j]$ represents the edge features between nodes i and j. If nodes i and j belong to the same group of actions, they are expected to be visually more coherent. Thus, we intend to excite the edges when nodes i and j belong to the same group of action and inhibit the edges when they are in a different group of actions. There are many ways to do it. In our experiments, we increase the weight of edge between nodes i and j by a constant value when they belong to the same category (excitation operation), and we decrease it by the same constant value when they are different (inhibition operation). Let us denote the modified adjacency matrix after this weight adjustment process as $A_{grounding}$. We set the mean of the row normalized language Adjacency matrix $A_{language}$ as the constant to be added or subtracted in excitation or inhibition. The intuition behind this is that this way the edge weights remain positive, and excitation is still mostly in the same order of original edge weight. Very high values would lead to instability, especially in the inhibition operation and very small values would not lead to any significant changes to the adjacency matrix. In the experimental section, we provide a sensitivity analysis of the choice of constant and how it impacts the performance.

After we obtain $A_{grounding}$ by modifying the original adjacency matrix $A_{language}$, we use $A_{grounding}$ for graph propagation in equation 2. Equation 5 describes the process of graph propagation.

$$H_{l+1} = f(D^{-1} \times A_{grounding} \times H_l \times W_l) \tag{5}$$

Modifying the distances of word embeddings by a constant based on the grouping of actions, may not be the optimal way to solve the domain shift between language embeddings and visual embeddings. However, it is very efficient in terms of ease of implementation and it incorporates the visual domain knowledge in the input adjacency matrix. Once the adjacency matrix $A_{language}$ is modified based on the grouping of actions in equation 5, we use the same propagation rule as given in the equation 2 to learn the classifier weights of test classes. Essentially, we are training the GCN only once in the entire process thereby reducing the complexity of incorporating visual feedback to the whole process.

3.6. Modifying Weights Via The Visual Adjacency Matrix

Here we introduce second method to learn the visual feedback required to incorporate the domain shift between

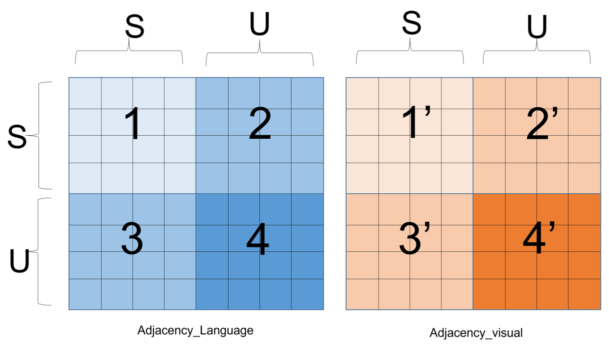

Figure 2. Learning the transformation from $A_{language}$ to A_{visual}

the language graph and visual graph. Let us refer to the adjacency matrix of all the visual embeddings as the visual adjacency matrix A_{visual}. If we utilize only A_{visual} for graph propagation, the updated GCN propagation rule becomes equation 6.

$$H_{l+1} = f(D^{-1} \times A_{visual} \times H_l \times W_l) \tag{6}$$

However A_{visual} is not available at training time. To obtain A_{visual} at testing time we learn a transformer function F shown in equation 7 that transforms the adjacency matrix from the language domain to the visual domain. In other words, the transformer function is learning how to predict the distances of nodes in the visual graph from the distances of nodes of the language graph. We use the GCN network to learn the visual embeddings of the test classes whereas the transformer function is used to estimate the distances between the visual embeddings of test classes.

$$F(A_{language}) = A_{visual}. \tag{7}$$

3.6.1 Learning Transformer Function F

Consider a language adjacency matrix $A_{language}$ with N nodes of dimension $N \times N$. The total number of classes in the dataset would also be equal to N. Let there be S training classes and U test classes in the dataset. We are interested in estimating a visual adjacency matrix A_{visual} with N nodes of dimension $N \times N$. $A_{language}$ can be easily obtained for all the N classes from it's word embeddings. However, for obtaining the A_{visual} we only have with us S classes during training. The remaining values have to be estimated. Training one regression model for the entire dataset didn't yield good results. Instead, we split up the regression into parts and learn the different parts separately. The performance of the system depends on how good we are able to learn the transformer function F.

Each row in the adjacency matrix represents the cosine similarities between it and the rest of the classes. We rearrange the adjacency matrix for illustration purposes to show how we learn the transformer function. We rearrange it such that the first S rows of the adjacency matrix represent that of training classes and the next U rows represent that of test classes. Similarly, the first S columns represent that of training classes and the next U columns represent that of test classes. Figure 2 illustrates this rearrangement for illustration purposes. In figure 2 , submatrix 1 of $A_{language}$ contains the cosine similarities between the S training classes of $A_{language}$. Submatrix 2 contains the cosine similarities between S training classes and U test classes of $A_{language}$. Submatrix 3 contains the cosine similarities between U testing classes and S training classes of $A_{language}$. Submatrix 4 contains the cosine similarities between the U test classes of $A_{language}$. Submatrices $1', 2', 3', 4'$ similarly are rearranged for A_{visual}. Submatrix $1'$ is available at training and our goal is to obtain to submatrices $2', 3', 4'$

To obtain $2'$, we train S number of regression models one for each row of $2'$ utilizing one row each of 1and $1'$ as training data such that $F(1) = 1'$. $3'$ is obtained as the transpose of $2'$ due to symmetry. $4'$ is obtained by a regression model that is trained on the entire submatrices 1 and $1'$ such that $F(1) = 1'$. Each of these regression models consists of a two-layered fully connected layer neural network having 10 nodes in each layer having relu activation layer and solved using LFBGS solver.

Once A_{visual} is obtained, we could use it instead of the language matrix. However, there are advantages for using both A_{visual} and $A_{language}$ for the sake of making the model robust. There are various ways of utilizing both A_{visual} and $A_{language}$. Ma et al. [22] developed a multi-dimensional GCN where A_{visual} and $A_{language}$ can be two channels of a larger A matrix. However, for the sake of simplicity, we use a single-dimensional GCN and propose a new modified GCN propagation rule in equation 8 which includes both A_{visual} and $A_{language}$.

$$H_{l+1} = f(D^{-1} \times (A_{language} + \beta \times A_{visual}) \times H_l \times W_l) \tag{8}$$

Here β is the weight factor that can be determined empirically for best performance.

4. Experiments

4.1. Datasets

Experiments are conducted using the **EPIC-Kitchens-55** dataset [3] and the **Charades** dataset [28]. We chose these datasets as they are some of the largest human manipulation action datasets and we can test our ideas of visual grounding through grouping of actions on them.

EPIC-Kitchens-55 is an egocentric activity dataset of people recording their activities in kitchens. It has over 125 verbs and 331 nouns. We selected the top 50 most frequent verbs provided by the authors of [3] for the experiments in this paper to avoid long tail problem. Among those, we used

22 verbs as training and 27 verbs as test classes for evaluation purposes. We eliminated the verb ”walk” because it is not a human manipulation action. For each verb, we manually formed the group of actions based on the geometric or topological change or the object motion involved as described earlier. The train and test splits and the groupings of actions will be released after acceptance.

Charades is a dataset made up of crowdsourced videos of activities in people's homes. The videos are on average 30 seconds long and each includes a sequence of multiple action classes. There are 157 action classes and for our experiments we used splits of 79 training and 78 test classes as reported in [9]. Descriptions of all videos along with the object classes are provided. For this dataset, because there are a significant amount of non manipulation actions and the action classes themselves are a combination of verbs and objects involved, we formed groups of actions based on the object involved. Intuitively, actions involving the same manipulated objects tend to have closer RGB visual features Since for this dataset the object manipulated in each video is provided, there was no need for further annotation.

Method	EPIC-Kitchens (22-27 split)
ESZSL [26]	7.33
DeViSE [6]	10.25
DEM [34]	8.66
knowledge graphs [9]	13.94
GCN baseline	15.21
Ours GCN-I	16.78
Ours GCN-IE	16.94
Ours GCN-E	**19.62**

Table 1. Evaluation of classification accuracy for novel test classes on EPIC-Kitchens Dataset.

Method	Charades
ESZSL [26]	17.21
Knowledge graphs [9]	18.21
GCN baseline	18.43
Ours GCN-I	18.16
Ours GCN-IE	19.16
Ours GCN-E	**19.35**
Ours GCN AV*	**19.19**
Ours GCN AV	**19.63**
GCN baseline with attack	15.846
GCN- V after attack	**17.11**

Table 2. Evaluation of classification accuracy for 79 novel test classes on Charades Dataset. We report mean average precision mAP.

4.2. Implementation Details

4.2.1 Baseline GCN

We used the I3d network [1] to obtain classifiers for the Epic-Kitchens-55 and the Charades dataset. EPIC-Kitchens-55 is a highly unbalanced dataset when it comes to the number of videos per class. This resulted in poor performance in zero-shot learning. Therefore we sampled the dataset such that approximately 50 samples per class were present in the training set. We initialized the weights of the I3d network with those pre-trained on Imagenet and finetuned the last layer of the I3d classifier for the EPIC-Kitchens dataset. For the Charades dataset, the I3d model classifier was trained end to end and we followed the insights from [9] to obtain the I3d classifier weights.

The trained I3d classifier weights were then used to train the GCN. We used a two-layer GCN having hidden layers of 1024 $\rightarrow$1024 dimensions to predict the I3d classifier weights. Empirically we found that two-layeres in the GCN gave the best results, better than deeper GCNs. We initialized the input graph nodes with 300-dimensional GloVe embeddings trained on the Wikipedia dataset. The input graph node dimension is 300, and the output graph node dimension is 1024, which is equal to the I3d feature dimension. The number of nodes of the graph for the EPIC-Kitchen dataset was 50, which is equal to the number of verbs we selected in our dataset. Although we removed ”walk” from the dataset we didn't remove its nodes in the graph as this didn't affect much the experiments. The number of nodes of the graph for the Charades dataset was 157, which is equal to the total number of classes in the dataset. We updated only the training nodes during the training. We used the Adam optimization algorithm with a learning rate of 0.0005 and momentum of 0.0001 for the Charades dataset, a learning rate of 0.0001, and momentum of 0.0001 for the EPIC-Kitchen dataset. The GCN was trained with L2 loss to predict the classifier weights for about 15000 epochs for the Charades dataset and for 400 epochs for the EPIC-Kitchens dataset. For all subsequent experiments and ablation studies we used the same architecture setting and network learning parameters. Only the way the adjacency matrix was computed differs.

4.3. Experimental details:

Table 1 and Table 2 summarize the results of zero-shot action recognition for different ways of forming the graph and training it on the EPIC-Kitchens and Charades dataset, respectively. We compare our method against a baseline GCN with no modification of the weights and other methods previously reported in the literature.

Table 2 shows the results of zero-short action recognition on the Charades dataset. The results of ESZSL [26] and [9] are from [9]. We report the Mean Average Precision of the test categories for the Charades dataset and report percentage accuracy for the test categories of the EPIC-Kitchens dataset. GCN-I stands for GCN with inhibition of edges and GCN-E stands for GCN with excitation of edges. GCN-IE stands for GCN with both excitation and inhibition. GCN-E was the best performing model for the Charades dataset, outperforming the baseline GCN, knowledge graphs [9], and ESZSL method [26]. GCN-IE which has both excitation and inhibition is the second best. Inhibition of edges is not leading to significant improvement in performance compared to excitation.

A similar trend is also seen for the zero-short action recognition results on the Epic-Kitchens dataset. We implemented ESZSL [26], DeViSE [6], DEM [34] baselines and compared against our methods as shown in Table 1. Our method GCN-E nearly has 10% improvements over previous zero-shot action recognition methods. We further report experiments of using the estimated A_{visual} adjacency matrix on the test splits of the Charades dataset in table 2. $GCNAV*$ represents GCN using $A_{language}$ and estimated A_{visual}. $GCNAV$ represents GCN using $A_{grounding}$ alongside estimated A_{visual}. Using the visual adjacency matrix on top of $A_{grounding}$ leads to the best performance on Charades, further validating the idea of visual feedback through the A_{visual}.

Method	Charades (40-39 split)
$GCN - L$	12.92
$GCN - V_{ideal}$	15.60
$GCN - V_{ideal}L$	**17.28**
$GCN - V_{estimated}$	12.06
$GCN - V_{estimated}L$	**13.46**

Table 3. Evaluation of classification accuracy for 39 novel test classes in ablation study on Charades Dataset. Results are in mAP.

Method	Set 1	Set 2	Set 3
GCN-I	17.21	16.78	16.79
GCN IE	16.48	16.94	16.3
GCN E	**18.72**	**19.62**	**18.61**

Table 4. Evaluation of classification accuracy for novel test classes on EPIC-Kitchens Dataset for different groupings of actions

4.3.1 How to set the "constant" parameter used in inhibition and excitation process?

As described earlier, the "constant" parameter can be set as the mean (over all elements) of the normalized $A_{language}$ matrix. Table 5 shows the sensitivity to this "constant" parameter on the performance of GCN-E and GCN-I for the EPIC-kitchens dataset. A smaller value than the mean of $A_{language}$ results in less accuracy improvements for GCN-E. For this experiment the mean of $A_{language}$ was 0.02. For GCN-I however a value slightly lower than the mean of $A_{language}$ results in best performance. We can tune the system to get the best values for the "constant" starting with the mean.

	$c = 0.02$	$c = 0.01$	$c = 0.005$
GCN-E	18.72	17.90	16.90
GCN-I	5.18	17.21	15.90

Table 5. Variation of classification accuracy on novel test classes on EPIC-kitchens dataset for different "constant" parameter used in excitation and inhibition process. In the table c refers to "constant".

	$\beta = 0.01$	$\beta = 0.02$
Test Acc	19.19	18.97

Table 6. Variation of classification accuracy for 79 novel test classes on Charades dataset for different β in equation 8 for GCNAV* setting.

4.3.2 Studying the effect of weight adjustment on different groupings of actions

We wanted to check if our proposed method is relatively agnostic to the choice of shared concepts used to group actions. We created three sets of different shared concepts and made groups of actions based on those sets (with a varying number of shared concepts). We performed the experiments on these three sets on the EPIC-kitchens dataset. Table 4 shows that our method of weight adjustment is agnostic to grouping of actions. Here we list the different shared concepts that were chosen for each set.

- Set1: Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate.
- Set2: Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate, complex motion
- Set3: Translational motion, Complex motion, Decrease in size, Increase in size, Add (to the scene), Remove (from the scene), Separate.

4.3.3 Effect of Visual Adjacency Matrix

To evaluate the impact of the visual adjacency matrix on the overall GCN propogation (equation 8), we performed some ablation experiments on the Charades dataset. For this, we only used the training set of 79 action classes in the charades dataset. Among these, we randomly chose 39

test classes and the remaining 40 were kept in the training. Then we performed zero-shot action recognition on the 39 test classes by using a different adjacency matrix in each case study. We analyze the following different cases of initial adjacency matrix:
1. GCN-L: This is a baseline GCN that uses the language adjacency matrix and the formulation represented in equation 2. We constructed a 79×79 dimensional language adjacency matrix $A_{language}$ and computed the test accuracy for 39 classes.
2. $GCN - V_{ideal}$: This is a GCN using the ground truth visual adjacency matrix. From this experiment, we could understand the upper limit of test class accuracy using the visual adjacency matrix A_{visual}. For this experiment, we computed the adjacency matrix using the entire 79 training classes in the original charades dataset. Instead of estimating the visual adjacency matrix, we used the ground truth I3d class weights and constructed A_{visual}. We computed the test accuracy for 39 classes and used the formulation in equation 6.
3. $GCN - Vestimated$: This is a GCN that uses the estimated visual adjacency matrix A_{visual} and the formulation in equation 6. We used 40 training classes and 39 test classes to learn A_{visual}. We set up a transformer network, as described in subsection 3.6.1 to obtain F. From F we obtained the visual adjacency matrix A_{visual}. We used the visual adjacency matrix A_{visual} in equation 6 to obtain the zero-shot action test accuracy for 39 classes.
4. $GCN - V_{ideal}L$: This is a GCN that uses both the ideal visual adjacency matrix A_{visual} and the original language adjacency matrix $A_{language}$. We used equation 8 to obtain the zero-shot action test accuracy for 39 classes. β in equation 8 was set to 0.2
5. $GCN - VestimatedL$: This is a GCN that uses only an estimated visual adjacency matrix A_{visual} and A_{visual} language adjacency matrix $A_{language}$. We used 40 training classes and 39 test classes. We set up the transformer network described in subsection 3.6.1 to obtain F. From F we obtained the visual adjacency matrix A_{visual}. We used the visual adjacency matrix in equation 8 to obtain the test accuracy for 39 classes. β used in equation 8 is 0.1

Table 3 reports the results of the ablation study. We report the Mean Average Precision for 39 test classes in the table for the Charades dataset. We will release the training and testing splits for this ablation study upon acceptance. From $GCN - V_{estimated}L$ results in table 3, we can see that using the estimated visual adjacency matrix A_{visual} has a lot of value on its own and performs as well as using it instead of $A_{language}$. Experiments on $GCN - V_{ideal}L$ were conducted to find the upper limit of test accuracy using A_{visual}. The experiments show that just the idea of using A_{visual} is beneficial provided we are able to learn a good transformer function F. However, for all practical purposes we have to rely on the estimated visual adjacency matrix A_{visual} using the transformer function F. Training the transformer function F is hard as we have limited data points for training and it is a complex function to learn. Learning a better F depends on accuracy of $A_{Languagetrain}$ and $A_{visualtrain}$. If both of these are unreliable, then the overall accuracy of the system will be poor, and this is one of the limitations of our approach.

Setting parameter β We performed experiments to determine sensitivity of parameter β in equation 8. Table 6 reports experiments for different parameters of β in Equation 8 for the Charades dataset experiments reported in Table 2 for GCNAV * setting. An initial value can be set like in any multitask learning.

4.3.4 Robustness to Noise in the Language Graph

While there is no direct negative social impact with our approaches, graphs are easily prone to adversarial attacks. We evaluate the performance of GCNs by creating perturbations on the input language graph. We created perturbation on the input language graph by poisoning the edges. Specifically, we poisoned the edges of those verbs belonging to same grouping of actions. We decreased the weight of these edges by 0.1. We then re-ran our algorithms on this setting. GCN after it was attacked had a 15.8 MAP on the charades test set when using the attacked adjacency matrix. $GCN - V$ uses the estimated visual adjacency matrix A_{visual} from the attacked adjacency matrix along with the attacked adjacency matrix. $GCN - V$ had a 17.11 MAP on test classes showing robustness when using the estimated visual adjacency matrix despite the original language matrix being attacked.

5. Conclusion

We introduced two methods to provide visual feedback in language graphs for zero-shot action recognition. The use of grouping of actions allowed us to adjust additively the weights of the adjacency matrix to reflect the semantic similarity of actions in visual space. This process of increasing and decreasing the graph edge weights is simple yet very effective and can be easily adopted in other weight adjusting mechanisms. We also introduced the use of the visual adjacency matrix for setting the weights in the graph, which is usually overlooked in zero-shot learning. Experimental results on the EPIC-Kitchens and the Charades data sets showed that the addition of the visual adjacency matrix and use of weight adjustment to modify the adjacency matrix is beneficial.

References

[1] 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. 3, 6

[2] Vincent S. Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. Scene graph prediction with limited labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The EPIC-KITCHENS dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018. 5

[4] Eadom Dessalene, Chinmaya Devaraj, Michael Maynord, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[5] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *arXiv preprint arXiv:1906.01852*, 2019. 2

[6] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. 2013. 6, 7

[7] Yanwei Fu, Timothy M Hospedales, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Transductive multi-view embedding for zero-shot recognition and annotation. In *European Conference on Computer Vision*, pages 584–599. Springer, 2014. 2

[8] Chuang Gan, Ming Lin, Yi Yang, Yueting Zhuang, and Alexander G Hauptmann. Exploring semantic inter-class relationships (SIR) for zero-shot action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 2

[9] Pallabi Ghosh, Nirat Saini, Larry S Davis, and Abhinav Shrivastava. All about knowledge graphs for actions. *arXiv preprint arXiv:2008.12432*, 2020. 2, 6, 7

[10] Pallabi Ghosh, Nirat Saini, Larry S Davis, and Abhinav Shrivastava. Learning graphs for knowledge transfer with limited labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11151–11161, 2021. 2

[11] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9211–9219, 2019. 2

[12] 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

[13] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020. 1

[14] Nan Jia, Xiaolin Tian, Yang Zhang, and Fengge Wang. Semi-supervised node classification with discriminable squeeze excitation graph convolutional networks. *IEEE Access*, 8:148226–148236, 2020. 2

[15] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11487–11496, 2019. 3

[16] Keizo Kato, Yin Li, and Abhinav Gupta. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–251, 2018. 2

[17] Muhammad Raza Khan and Joshua E Blumenstock. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 606–613, 2019. 2

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

[19] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019. 2

[20] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE international conference on computer vision*, pages 1261–1270, 2017. 1

[21] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR 2011*, pages 3337–3344. IEEE, 2011. 2

[22] Yao Ma, Suhang Wang, Chara C Aggarwal, Dawei Yin, and Jiliang Tang. Multi-dimensional graph convolutional networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 657–665. SIAM, 2019. 5

[23] 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. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 372–380, 2018. 2

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 2

[25] Marcus Rohrbach, Sandra Ebert, and Bernt Schiele. Transfer learning in a transductive setting. *Advances in Neural Information Processing Systems*, 26:46–54, 2013. 2

[26] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015. 6, 7

[27] Gunnar A Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? In *Proceedings of the IEEE international conference on computer vision*, pages 2137–2146, 2017. 4

[28] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 5

[29] Ilaria Tiddi and Stefan Schlobach. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence*, 302:103627, 2022. 1

[30] Oytun Ulutan, ASM Iftekhar, and Bangalore S Manjunath. Vsgnet: Spatial attention network for detecting human object interactions using graph convolutions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13617–13626, 2020. 1

[31] 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

[32] Qian Wang and Ke Chen. Zero-shot visual recognition via bidirectional latent embedding. *International Journal of Computer Vision*, 124(3):356–383, 2017. 2

[33] Florentin Wörgötter, Eren Erdal Aksoy, Norbert Krüger, Justus Piater, Ales Ude, and Minija Tamosiunaite. A simple ontology of manipulation actions based on hand-object relations. *IEEE Transactions on Autonomous Mental Development*, 5(2):117–134, 2013. 3

[34] 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. 6, 7