ABSTRACT

Title of Dissertation: SCENE AND ACTION UNDERSTANDING USING CONTEXT AND KNOWLEDGE SHARING

Pallabi Ghosh
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Larry S. Davis
Professor Abhinav Shrivastava
Department of Computer Science

Complete scene understanding from video data involves spatio-temporal decision making over long sequences and utilization of world knowledge. We propose a method that captures edge connections between these spatio-temporal components or knowledge graphs through a graph convolution network (GCN). Our approach uses the GCN to fuse various information in the video like detected objects, human pose, scene information etc. for action segmentation. For certain functions like zero shot and few shot action recognition, we learn a classifier for unseen test classes through comparison with similar training classes. We provide information about similarity between two classes through an explicit relationship map i.e. the knowledge graph. We study different kinds of knowledge graphs based on action phrases, verbs or nouns and visual features to demonstrate how they perform with respect to each other. We build an integrated approach for zero-shot and few-shot learning. We also show further improvements through adaptive learning of the input knowledge graphs and using triplet loss along with the task specific loss while training.

We add results for semi-supervised learning as well to understand improvements from our graph learning technique.

For complete scene understanding, we also study depth completion using deep depth prior based on the deep image prior (DIP) technique. DIP shows that structure of convolutional neural networks (CNNs) induces a strong prior that favors natural images. Given color images and noisy or incomplete target depth maps, we optimize a randomly-initialized CNN model to reconstruct a depth map restored by virtue of using the CNN network structure as a prior combined with a view-constrained photo-consistency loss. This loss is computed using images from a geometrically calibrated camera from nearby viewpoints. It is based on test time optimization, so it is independent of training data distributions. We apply this deep depth prior for inpainting and refining incomplete and noisy depth maps within both binocular and multi-view stereo pipelines.

SCENE AND ACTION UNDERSTANDING USING CONTEXT AND KNOWLEDGE SHARING

by

Pallabi Ghosh

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Abhinav Shrivastava, Co-Advisor
Professor Behtash Babadi
Professor David Jacobs
Professor Soheil Feizi

Acknowledgments

I am thankful to everyone who have helped me in the course of my PhD. The last few years have been an amazing learning experience and I am very grateful to all the people who have made it possible.

First and foremost I would like to thank both my advisors, Dr. Larry S. Davis and Dr. Abhinav Shrivastava without whose guidance and support I would not have been able to complete this degree. Their expertise in both the actual research work as well as publishing said works have been invaluable. I would thank Dr. Behtash Babadi, Dr. David Jacobs and Dr. Soheil Feizi for being a part of my thesis committee and for helping me make my final manuscript better. I would also like to thank Dr. Tom Goldstein for being a co-author on my paper and helping direct it.

Next I would like to acknowledge the help from the staff members of the Computer Science department and ISSS who made all the administrative work and maintenance of my international student status seem so easy. I would like to thank Tom Hurst, Jennifer Story, Janice M. Perrone who helped me with all form submissions, reimbursements, extension of my student status etc. Without their help I would not have been able to spend as much time on my research work as I have.

My internship mentors and co-authors, Yi Yao and Ajay Divakaran at SRI International and Vibhav Vineet, Sudipta Sinha and Neel Joshi from Microsoft

Research have also guided me extensively throughout those research projects and I am extremely grateful for their advice.

My lab-mates, housemates and friends have also been extremely helpful in the course of my PhD. My co-authors Sohil Shah, Nirat Saini, Bor-Chun Chen and Vlad Morariu were instrumental in those works and labmates Kamal Gupta, Gaurav Shrivastava and Soumyadip Sengupta have helped whenever I was stuck at some point in my research. Also my friends Sudha Rao, Manaswi Saha, Kartik Nayak, Meethu Malu, Nidhi Shah and Gowthami Somepalli helped me in the ups and downs throughout my PhD career.

Last but not the least, I would like to thank my family members. Both my parents Nivedita Ghosh and Dr. Kiriti Bhusan Ghosh have guided me at various stages of my education and development. My sister Dr. Shrutakirti Ghosh has also been instrumental in my success as a student and my husband Dr. Bhaskar Ramasubramanian has stood by me in all my decisions.

I would like to acknowledge the financial support provided by DARPA MediFor program and Small Business Technology Transfer from the AirForce.

Finally I would like to thank all the UMD staff and faculty who made work possible even during the COVID19 pandemic and helped create a safe environment for us to work.

It is impossible to thank every person individually who have helped me during my PhD in this limited space. I am thankful for all their help.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In this thesis we show the utilization of context and knowledge bases for overall scene understanding. We improve action recognition and depth perception using various levels of supervision like fully supervised or zero-shot and few-shot action recognition, graph based semi-supervised learning and test time optimization for depth completion.

1.1 Action Segmentation

We construct a comprehensive spatio-temporal graph (STG) to jointly represent an action along with its associated actors, objects, and other contextual cues [6]. Specifically, graph nodes represent actions, actors, objects, and scenes; spatial edges represent spatial (e.g., next to, on top of, etc.) and functional relationships (e.g., attribution, role, etc.) between two nodes with importance weights; and temporal edges represent temporal and causal relationships. We exploit a variety of descriptors in order to capture these rich contextual cues. In literature, there exist various networks for situation recognition, object detection, scene classification, and semantic segmentation. The outputs of these networks provide embeddings that can serve as the node features of the proposed STGs. We show a sample STG

constructed from video data in Figure 1.1.

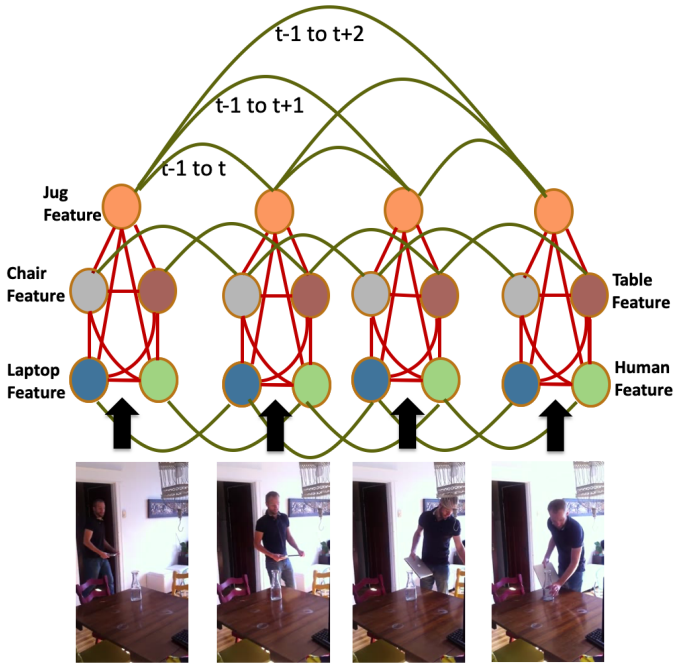

Figure 1.1: Constructing a spatio-temporal graph for video analysis. The nodes in the graph are features from different aspects of the scene like object descriptors for jug, chair, table and laptop and the human pose descriptor. We show the spatial and temporal connections (upto three time steps) in the graph.

We perform action segmentation on top of this STG via a stacked spatio-temporal graph convolution network (Stacked-STGCN). Our STGCN stems from the networks originally proposed for skeleton-based action recognition [5] and introduces two major advancements as our innovations. First, to accommodate various contextual cues, the nodes of our STG have a wide range of characteristics, leading to the need for using descriptors with varied length. Second, our STG allows arbitrary edge connections (even fully connected graph) to account for the large amount of graph deformation caused by missed detections, occlusions, and emerging/disappearing objects. These two advancements are achieved via enhanced designs with additional layers.

Another innovation we introduce is the extended use of stacked hourglass architecture on graph data. Stacked hourglass networks have been applied to grid-like data with regular connections (e.g., images using CNNs) and shown improved results for a number of tasks such as human pose estimation [7], facial landmark localization [8], etc. They allow repeated upsampling and downsampling of features and combine these features at different scales, leading to better performance. We propose to extend this encoder decoder architecture to graph data with irregular connections. Different from CNN, STGCN (or more general GCN) employs adjacency matrices to represent irregular connections among nodes. To address this fundamental difference, we adapt the hourglass networks by adding extra steps to down-sample the adjacency matrices at each encoder level to match the compressed dimensions of that level.

To summarize, the proposed Stacked-STGCN offers the following innovations:

- Joint inference over a rich set of contextual cues.
- Flexible graph configuration to support a wide range of descriptors with varied feature length and to account for large amounts of graph deformation over long video sequences.
- Stacked hourglass architecture specifically designed for graph data with irregular connections.

These innovations improved recognition and localization accuracy, robustness, and generalization performance for action segmentation over long video sequences.

1.2 Zero-shot and Few-shot Action Recognition

Action recognition has seen rapid progress in the past few years, including better datasets [9, 10] and stronger models [11, 12, 13, 14, 15, 16]. Despite this progress, it is not easy to train an action classifier for a new category. A potential solution is to leverage the knowledge from seen or familiar categories to recognize unseen or unfamiliar categories. This is the zero-shot learning paradigm, where we transfer or adapt classifiers of related, known, or seen categories to classify unseen ones (Figure 1.2). Similarly, for few-shot action recognition, instead of testing on completely unseen classes, we have only a few labeled samples from the test classes, which help in learning about the rest of the test samples.

Figure 1.2: Zero-shot action recognition based on similar training action classes. The classifier needs external knowledge to be able to group unseen class labelled "Playing field hockey" with seen classes labelled "Playing ice hockey" and "Playing soccer" and unseen class labelled "Playing guitar" with seen class labelled "Playing violin". It should not put "Playing basketball" with "Playing guitar" in-spite of the term "Playing" in both.

Zero-shot and few-shot learning methods have been studied widely for image classification. A recent research [17] builds a knowledge graph (KG) representing

relationships between seen and unseen classes and then trains a graph convolutional network (GCN) on this KG. This helps to transfer classifier knowledge from seen to unseen classes. Using the same technique for action recognition is hard since, unlike objects, it is unclear what is the best knowledge representation for actions. One of the reasons as observed in [18] is that verbs have a broader definition and conflicting meaning compared to nouns and we will be giving some examples in the next paragraph where action definitions can be confusing.

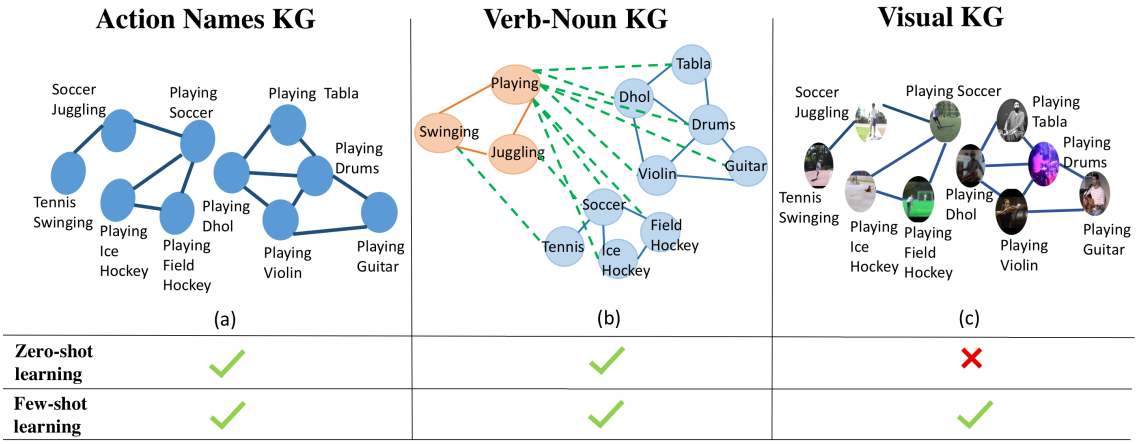

Figure 1.3: Different KGs for zero/few-shot action recognition. They are based on action class names, verbs and nouns related to these class names and visual features extracted from the videos belonging to each class.

In this thesis, we study the performance improvements by using different types of KGs for zero-shot and few-shot action recognition (Figure 1.3) [19]. The primary step in building a KG is generating a good implicit representation for each action class. In image classification, standard word embeddings (word2vec [20, 21, 22], GloVe [23], ConceptNet [24], etc.) capture the semantic knowledge associated with well-defined class names. However, for action classification, class names vary from single words ("sit", "stand", etc.) to phrases ("shooting ball (not playing baseball)")

and there are multiple definitions of the same (or similar) action class(es); like, "apply eye makeup" or "put on eye-liner". Such diversity is less pronounced in image classification tasks due to the simplicity of labels. Our first contribution is studying different implicit representations for action classes and showing the advantages of a sentence2vector model in capturing the semantics of word sequences for zero/few-shot action recognition.

Our second contribution is building an explicit relationship map from these implicit representations of action classes. In image classification, the explicit representations for transferring knowledge from seen to unseen categories are using attributes or external KGs. Several datasets provide labeled class-attribute pairs (e.g., AwA [25] , aYahoo [26], COCO-Attributes [27], MITstates [28], etc.). Similarly, many KGs have nodes that correspond to image classification classes (e.g., WordNet [29], NELL, and NEIL [30, 31]). In contrast, such sources are scarce for action classes. Wordnet contains verbs, therefore, it can be used to construct a KG for verbs, but we cannot have a KG with nodes representing the entire phrase (eg., "playing(verb) guitar(noun)") for an action class. Instead, there will be separate nodes for verbs and objects with defined inter-relationships. ConceptNet [24] has some phrases, but the list is not exhaustive and a lot of label names in our datasets are not present in ConceptNet. On the other hand, we build a KG with an explicit relationship of the multi-word action phrases in any dataset. We append dataset with action classes from other datasets and construct two KGs, one for noun, and other for verb either by splitting the action phrase in cases like "playing(verb) guitar(noun)" or using WordNet to get the nearest noun in cases like "cake"(noun) for

action class named "baking"(verb). Further, we build a KG for few-shot learning using mean features of training data-points per class. We use a combination of this KG with the two KGs defined previously and observe performance improvement.

Finally, majority of previous work on zero-shot action recognition use image-based learning models to estimate actions in videos. Recent advances in action recognition lead to the use of a network trained on a video dataset as the feature extractor. Such a system requires an improved evaluation paradigm, since the action classes in the training set cannot be in the test set. We manually check for commonalities between the training datasets (Kinetics) and testing datasets (UCF101, HMDB51, Charades), but could not resolve problems within Kinetics which is a huge dataset and can have videos common across multiple classes. We keep all Kinetics classes in training set and remove common classes from Kinetics with UCF101, HMDB51 and Charades from the test set. Our third contribution is the creation of this evaluation paradigm using UCF101, HMDB51, Charades, and Kinetics datasets.

To summarize, our main three contributions are:

- Better implicit representation of action phrases (which are word sequences) using sentence2vec.
- Comparative study of different KGs for action zero-shot/few-shot learning.
- Developing an improved evaluation paradigm for zero-shot/few-shot action recognition using networks trained on video datasets as feature extractors.

These 3 contributions together builds an integrated approach for both zero-shot and few-shot learning.

1.3 Learning Graphs for Knowledge Transfer

One of the key limitations of the 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., zero-/few-shot learning) are also used to update the structure of the input adjacency matrix. This is in stark contrast with other related graph learning works [32, 33], which have a separate dedicated network and special loss functions to update the adjacency matrix. As we demonstrate empirically, the benefit of using the downstream tasks' losses is that the learned graph using our approach is better suited for the downstream task. 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. 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 1.4, where we learn better connections for the class "Mixing batter". A language-based 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

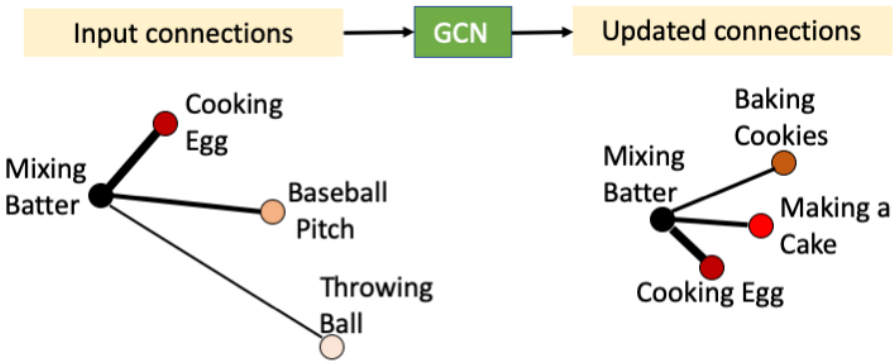

Figure 1.4: We use a GCN to update the input graph connections and show results for "Mixing Batter" class in zero-shot action recognition. Language based models associates "batter" to "baseball" which is rectified in the updated graph.

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 graph. Simple heuristics, such as fixed degree for each node, 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, we 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-based framework for the downstream task. Our formulation selects positive and negative neighbors for each node in the graph, and uses them to

add constraints on its degree while avoiding degenerate solutions 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 knowledge transfer or aggregation frameworks
- 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 [34] built on network datasets, like Cora, Citeseer, and Pubmed [35] with accompanying well-defined input graphs. The knowledge is transferred using GCN 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. For zero-shot/few-shot learning, we focus on our action recognition pipeline with input knowledge graph (KG) built from sentence2vec [36] embeddings.

1.4 Depth Completion

There are numerous approaches for estimating scene depth, such as using binocular [37] or multi-view [38, 39, 40] stereo, or directly measuring depth with depth cameras, eg. LIDAR, etc. These approaches suffer from artifacts, such as noise, inaccuracy, and incompleteness, due to various limitations. As depth es-

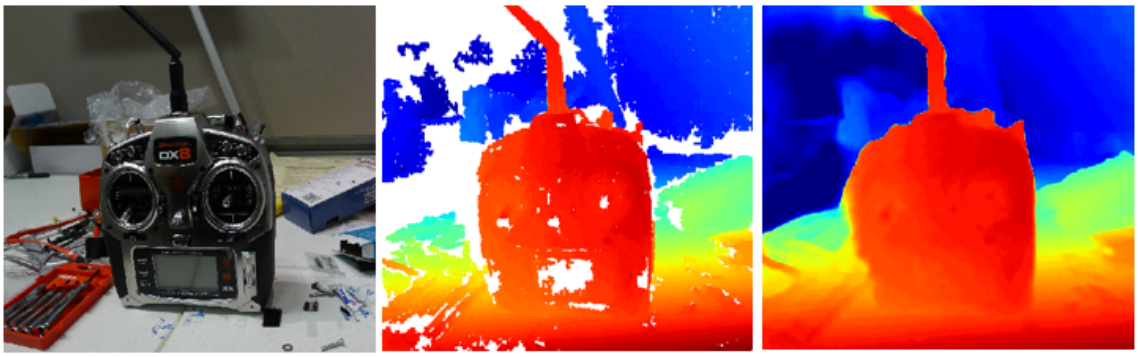

Figure 1.5: (a) Input image from one viewpoint (b) Target depth map computed with SGM using multiple neighboring images[4] (c) The refined depth map generated using our deep depth prior (DDP) technique. Depth map (c) has the holes from depth map (b) (shown in white) filled.

timation is an ill-posed problem, extensive research has been conducted to solve the problem using approximate inference and optimization techniques that employ appropriate priors and regularization [41, 42, 43, 44].

Supervised learning methods based on convolutional neural networks (CNNs) have shown promise in improving depth estimations, both in the binocular [45, 46, 47] and multi-view [48, 49, 50] stereo settings. However, these supervised methods rely on vast amounts of ground truth data to achieve proper generalization. While unsupervised learning approaches have been explored [51, 52, 53], their success appears modest compared to supervised methods.

In this thesis, we propose a new approach for improving depth measurements that is inspired by the recent work [54]. They demonstrated that the underlying structure of a encoder-decoder CNN induces a prior that favors natural images, a property they refer to as a "deep image prior" (DIP). The work on DIP shows that the parameters of a randomly initialized encoder-decoder CNN can be optimized to

map a high-dimensional noise vector to a single image. When the image is corrupted and the optimization is stopped at an appropriate point before overfitting sets in, the network outputs a noise-free image. The DIP has since been used as a regularizer in a number of vision tasks such as image denoising and inpainting [54, 55, 56].

We use DIP-based regularization for refining and inpainting noisy and incomplete depth maps obtained from a wide variety of sources [57]. Using a network similar to DIP, our approach generates a depth map by combining a depth reconstruction loss with a view-constrained photoconsistency loss. The latter loss term is computed by warping a color image into neighboring views using the generated depth map and then measuring the photometric discrepancy between the warped image and the original image.

In this sense, our technique resembles direct methods proposed for image registration problems, which all employ initialization and iterative optimization. However, instead of using handcrafted regularizers in the optimization objective, we use the deep image prior as the regularizer. While the role of regularization in end-to-end trainable CNN architectures is gaining interest [58, 59], our method is quite different, because there is no training and the network parameters are optimized from scratch on each set of test images. Figure 1.5 shows the inpainting results of applying our technique (DDP) on an input depth map with holes.

To the best of our knowledge, this is the first work to investigate deep image priors for completing depth images. We evaluate our approach using results from modern stereo pipelines and depth cameras and show that the refined depth maps are more accurate and complete, leading to more complete 3D models.

To summarize, our contributions are:

- A deep prior network for depth completion that allows test time optimization and is independent of training data distributions
- Using combination of reconstruction loss with view-constrained photo-consistency loss that works better than just the mean squared error loss used in DIP

Chapter 2: Related Works

2.1 Action Recognition and Segmentation

Action recognition is an example of one of the classic computer vision problems. In the early days, features like PCA-HOG, SIFT, dense trajectories, etc. were used in conjunction with optimization techniques like HMM, PCA, Markov models, SVM, etc. In 2014, Simonyan and Zisserman used spatial and temporal 2D CNNs [60]. That was followed by the development of 3D convolutions with combined spatial and temporal convolutional blocks. Since then a series of works following these two schemes, two-stream and 3D convolution, were studied including TSN [15], ST-ResNet [61], I3D [11], P3D [13], R(1+2)D [14], T3D [12], S3D [16], etc. Another popular type of deep neural networks used for action recognition is the Recurrent Neural Network (RNN) including Long Short-Term Memory networks (LSTM), which are designed to model sequential data. Particularly, RNNs/LSTMs operate on a sequence of per frame features and predict the action label for the whole video sequence (i.e., action recognition) or action of current frame/segment (i.e., action detection/segmentation). The structural-RNN (S-RNN) is one such method that uses RNNs on spatiotemporal graphs for action recognition [62]. The S-RNN relies on two independent RNNs, namely nodeRNN and edgeRNN, for it-

erative spatial and temporal inference. Recently, thanks to the rapid development in GNNs, graph-based representation became a popular option for action recognition, for instance skeleton-based activity recognition using STGCN [5], Graph Edge Convolution Networks [63] and Videos as space time region graphs [64]. In [64], GCN is applied to space-time graphs extracted from the whole video segment to output an accumulative descriptor, which is later combined with the aggregated frame-level features to generate action predictions. STGCN [5] was originally proposed for skeleton-based activity recognition. The nodes of the original STGCN are the skeletal joints, spatial connections depend on physical adjacency of these joints in the human body, and temporal edges connect joints of the same type (e.g., right wrist to right wrist) across one consecutive time step. STGCN on skeleton graph achieves state-of-the art recognition performance on multiple datasets. However, the STG is constructed based on human skeletons, which is indeed an oversimplified structure compared to the variety and complexity that our STG needs to handle in order to perform action segmentation with contextual cues and large graph deformation. Therefore, the original STGCN is not directly applicable. Instead, we use the original STGCN as our basis and introduce a significant amount of augmentation so that STGCN becomes generalizable to a wider variety of applications including action segmentation.

Action segmentation presents a more challenging problem than action recognition in the sense that it requires identifying a sequence of actions with semantic labels and temporally localized starting and ending points for each identified action. Conditional Random Fields (CRFs) are traditionally used for temporal inference [65].

Recently, there has been substantial research interest in leveraging RNNs including LSTM and Gated Recurrent Unit (GRU) [66, 67]. Lea *et al.*propose temporal convolutional networks (TCNs) [68], which lay the foundation for an additional line of work for action segmentation. Later, a number of variations of TCNs were also studied [69, 70, 71].

2.2 Learning on Graphs

Knowledge graphs(KGs) have been used in improving text based search engines [72, 73] and question answering [74, 75, 76, 77]. Automatic construction of a large KG and relationship learning has captured a lot of attention in the past [78, 79, 80, 81]. ConceptNet [24], FrameNet [82] and WordNet [29], are some of the existing large scale KGs.

KGs are also used to improve performance for image classification, representation learning, visual question answering and object detection techniques, [83, 84, 85, 86, 87, 88]. Hierarchical structure of KGs have contributed significantly in applications like learning similarity among classes [89] and for capturing hierarchical relationships between objects of distinct categories [90].

In recent years, there have been a number of research directions for applying neural networks on graphs. The original work by Scarselli *et al.*, referred to as the GNN, is an extension of the recursive neural networks and is used for sub-graph detection [91]. Later, GNNs were extended and a mapping function was introduced to project a graph and its nodes to an Euclidean space with a fixed dimension [92].

In 2016, Li *et al.*use gated recurrent units and better optimization techniques to develop the Gated Graph Neural Networks [93]. GNNs have been used in a number of different applications like situation recognition [94], human-object interaction [95], webpage ranking [91, 92], etc. The literature also mentions a number of techniques that apply convolutions on graphs. Duvenaud *et al.*were one of the first to develop convolution operations for graph propagation [96], whereas Atwood and Towsley developed their own technique independently [97]. Defferrard *et al.*use approximation in spectral domain [98] based on spectral graph introduced by Hammond *et al.* [99]. In [34], Kipf and Welling propose GCNs for semi-supervised classification based on similar spectral convolutions, but with further simplifications that result in higher speed and accuracy. Other GCN based semi-supervised learning work include [100, 101, 102, 103, 104]. Such works often use citation network datasets, like Citeseer, Cora, and PubMed [35], and protein-protein interaction dataset [105] for experimentation on semi-supervised learning. Some of our approaches utilize the GCN framework proposed by Kipf and Welling [34] as our GCN operator and the citation network datasets as our input.

Neural Graph Matching Networks was developed for few-shot learning in 3D action recognition [106]. Wang *et al.* [17] uses Graph Convolution network on KG for zero-shot image classification. The KG was formed with NELL(Never Ending Language Learning) [30], NEIL(Never Ending Image Learning) [31] and WordNet [29]. We use a similar model based on GCN for zero-shot actions. Dedre Gentner [18] shows how verbs and nouns have different levels of complexities and usually an action phrase comprises of both or just the verb. We explore different KGs, including

one with verbs and nouns only, to understand how these KGs improve performance for action recognition in zero-shot and few-shot learning setup.

Recently there has also been some work on graph learning networks for semi-supervised learning by [32, 33]. They develop a new loss to learn the edge weights in the graph. Instead of a separate network outputting the edge weights, we take the intermediate output of the original network and update the adjacency matrix. Our technique is more flexible, allowing for the update of node features as well as edge weights and connections when necessary. We also do not encounter computational complexity issues with increase in the length of the input node feature dimension unlike [32]. Kim *et al.*[107] 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 labelled nodes. In contrast, we build upon the GCN framework for zero/few-shot learning where nodes in the graph represents classes and not individual samples.

2.3 Zero-shot and Few-shot Learning

Zero-shot learning (ZSL) refers to the task of learning to predict on classes that are excluded from the training set [108]. Various studies do ZSL for image classification and object detection [25, 109, 110, 111, 112, 113, 114], as well as for action recognition [115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126]. The other zero-shot action papers, to the best of our knowledge, mostly are not GCN based, which has been proven to do better than traditional zero-shot techniques

for image classification [17]. While [81] is GCN based, their KG is very different from the one we use. They construct a single KG with actions and objects using ConceptNet [24], where nodes are connected based on word embedding. They use visual object features as a second channel to improve zero-shot learning. The number of objects in their graph is not dependent on the number of action classes. They show their best result when selecting 2000 most common visible objects in their dataset to get their object nodes, meaning they need access to the unlabelled test data (transductive). We use separate KGs for action, verb and noun and fuse them at the end with a fusion layer. Our verbs and nouns are dependent only on the action label and uses no visual information (inductive). Some other zero-shot learning research which we use as baselines include [127] and [128], where [127] uses a two layer network for learning relationships between features, attributes, and classes; while [128] uses the image feature space to map the language embedding, and not an intermediate space.

Few-shot learning (FSL) for image classification has been explored using meta-learning for learning distance of samples and decision boundary in the embedding space [129, 130, 131, 132], or by learning the optimization algorithm which can be generalized over different datasets [133, 134]. A benchmark for few shot image classification is created in [135]. For action recognition, studies propose embedding a video as a matrix [136, 137], using deep networks or generative models [138, 139] and using human-object interaction [140]. We tried GCN based FSL for action recognition, but our approach cannot be compared to many of these approaches due to two reasons – 1) Each paper uses a different dataset split, and our splits are

different as well because we use a pre-trained network from Kinetics in our pipeline; 2) We do not evaluate the episodic learning formulation like several other papers. Our aim is to build an integrated approach for zero and few-shot learning and also improve few-shot using the KG constructed for the zero-shot setting (relationship of class names, etc.) which, to the best of our knowledge, is not explored in the past.

2.4 Stereo Matching and Deep Stereo

Dense stereo matching is an extensively studied topic and there has been tremendous algorithmic progress both in the binocular setting [4, 45, 46, 51, 58] as well as in the multi-view setting [40, 49, 50, 141, 142], in conjunction with advances in benchmarking [143, 144, 145, 146, 147]. Traditionally, the best performing stereo methods were based on approximate MRF inference on pixel grids [41, 42, 43], where including suitable smoothness priors was considered quite crucial. However, such methods were usually computationally expensive. Hirschmuller proposed Semi-Global Matching (SGM) [4], a method that provides a trade-off between accuracy and efficiency by approximating a 2D MRF optimization problem with several 1D optimization problems. SGM has many recent extensions [148, 149, 150, 151, 152] and also works for multiple images [141]. Region growing methods have also shown promise and implicitly incorporate smoothness priors [40, 153, 154, 155]

In recent years, deep models for stereo have been proposed to compute better matching costs [45, 156, 157] or to directly regress disparity or depth [46, 47, 51, 158] and also for the multi-view setting [48, 49, 50]. Earlier on, end-to-end trainable CNN

models did not employ any form of explicit regularization, but recently hybrid CNN-CRF methods have advocated using appropriate regularization based on conditional random fields (CRFs) [58, 59]. In contrast with these works, as we do not perform learning by fitting to training data, our approach is more generalizable as it does not fall prey to the tendency of deep approaches to overfit to their training data.

2.5 Depth Map Refinement/Completion.

The fast bilateral solver [159] is an optimization technique for refining disparity or depth maps. However, the objective is fully handcrafted. Knoblereiter and Pock recently proposed a refinement scheme where the regularizer in the optimization objective is trained using ground truth disparity maps [160]. Their model learns to jointly reason about image color, stereo matching confidence and disparity. Voynov et al. [161] use a deep prior for depth super-resolution, but they do not have a multiview constraint, as we do, nor do they investigate refinement and hole-filling. Other recent disparity or depth map refinement techniques utilize trained CNN models [162]. Similarly depth map completion by Zhang et al. [2] use a learning based technique. They do single RGBD image depth completion whereas we use a multi-view photo-consistency loss for training our network. Also we show in out results that one main difference between their work and ours is that our result is not dependent on training data distributions. Depthcomp [3] also does depth completion and they use the semantic segmentation maps as prior knowledge.

2.6 Deep Prior for Color Images.

Beyond the previously discussed work of Ulyanov et al. [54], deep image priors have been extended for a number diverse applications – neural inverse rendering [163], mesh reconstruction from 3D points [55], and layer-based image decomposition [56]. Recently, Cheng et al. [164] pointed out important connections between DIP and Gaussian processes. Our approach is in a similar vein as these approaches, where we modify the DIP for depth maps by combing the usual reconstruction loss with a second term, the photoconsistency loss which ensures that when the reference image is warped into a neighboring view using our depth map, the discrepancy between the warped image and the original image is minimized.

Chapter 3: Stacked Spatio-Temporal Graph Convolutional Networks for Action Segmentation

In this chapter we describe out action segmentation pipeline using Stacked Spatio-Temporal Graph Convolutional Networks (Stacked-STGCNs). We provide the system overview in Figure 3.1. Each section in this figure in described in more detail.

3.1 Graph Convolutional Networks

Let a graph be defined as $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$ (see Figure 3.2). Vertex features of length d^0 are denoted as f_i for $i \in \{1, 2, \ldots, N\}$ where N is the total number of nodes. Edge weights are given as e_{ij} where $e_{ij} \geq 0$ and $i, j \in \{1, 2, \ldots, N\}$. The graph operation at the l^{th} layer is defined as:

$$\boldsymbol{H^{l+1}} = \boldsymbol{g(H^l, A)} = \boldsymbol{\sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^lW^l)} \tag{3.1}$$

where $\boldsymbol{W^l}$ and $\boldsymbol{H^l}$ are the $\boldsymbol{d^l \times d^{l+1}}$ weight matrix and $\boldsymbol{N \times d^l}$ input matrix of the $\boldsymbol{l^{th}}$ layer, respectively. $\boldsymbol{\hat{A} = I + A}$ where $\boldsymbol{A = [e_{i,j}]}$, $\boldsymbol{\hat{D}}$ is the diagonal node degree matrix of $\boldsymbol{\hat{A}}$, and $\boldsymbol{\sigma}$ represents a non-linear activation function (e.g., ReLU).

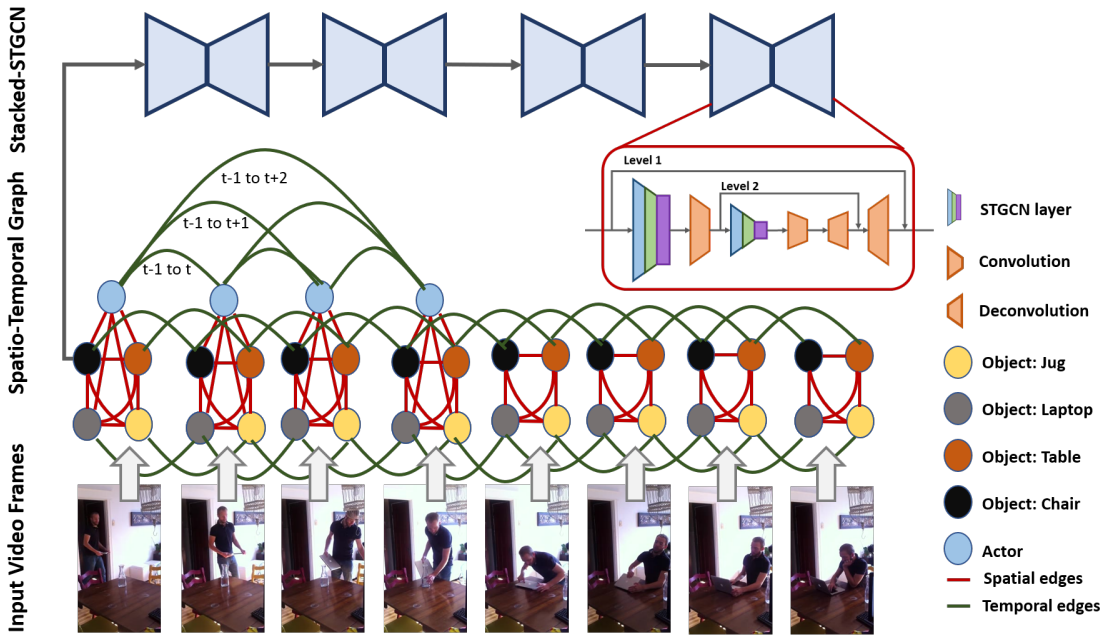

Figure 3.1: System overview. Different from the original STGCN based on human skeleton [5], our graph allows nodes of various types (such as actors, objects, and scenes) and with varied feature length. Our graph also supports flexible temporal connections (green lines) that can span multiple time steps, for example the connections among the actor nodes (blue nodes). Note that other nodes can have such temporal connections but are not depicted to avoid congested illustration. This spatio-temporal graph is fed into a stack of hourglass STGCN blocks to output a sequence of predicted actions observed in the video.

3.2 Spatio-Temporal GCNs

STGCN is originally designed for skeleton-based action recognition [5]. We apply STGCN for action segmentation of long video sequences using frame-based action graphs extracted via situation recognition [94]. To accommodate additional application requirements, our STG differs fundamentally in two aspects. First, the original STGCN is based on the human skeletal system with graph nodes corresponding to physical joints and spatial edges representing physical connectivity between these joints. Instead, we use human-object interactions to construct our spatial

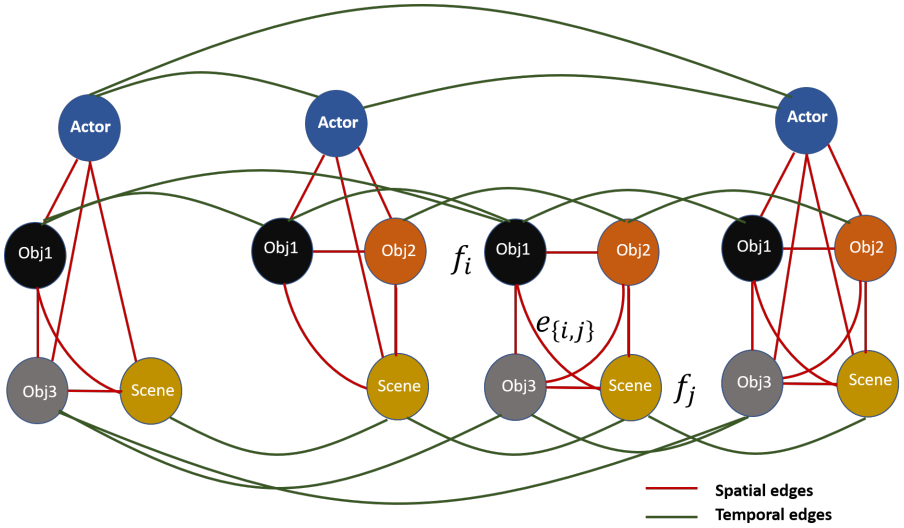

Figure 3.2: An illustration of spatio-temporal graphs. Each node v_i is represented by a feature vector denoted by f_i. The edge between node i and j has a weight $e_{i,j}$. These edge weights form the spatial and temporal adjacency matrices. Note that our spatio-temporal graph supports a large amount of deformation, such as missed detection (e.g., the actor node and the object 3 node) and emerging/disappearing nodes (e.g., the object 2 node).

graph where nodes represent actors, objects, scenes, and actions whereas edges represent their spatial (e.g., next to) and/or functional (e.g., role) relationships. Various descriptors can be extracted either as the channels or nodes of the spatial graph to encode comprehensive contextual information about the actions. For example, we can use pose feature to describe actor nodes, appearance features including attributes at high semantic levels for object nodes, and frame-level RGB/flow features for scene nodes.

Second, the original STGCN only connects physical joints of the same type across consecutive time stamps, which indeed reduces to a fixed and grid-like connectivity. As a result, the temporal GCN degrades to conventional convolution. To support flexible configurations and account for frequent graph deformation in

complex activities (e.g., missed detections, emerging/disappearing objects, heavy occlusions, etc.), our graph allows arbitrary temporal connections. For example, an object node present at time t_0 can be connected to an object node of the same type at time t_n with $n \geq 1$ in comparison to the original STGCN with $n = 1$.

Let A_s and A_t be the spatial and temporal adjacency matrices, respectively. Our proposed STGCN operation can be represented mathematically as follows:

$$\begin{aligned} \boldsymbol{H^{l+1}} &= \boldsymbol{g_t(H_s^l, A_t)} = \boldsymbol{\sigma(\hat{D}_t^{-1/2} \hat{A}_t \hat{D}_t^{-1/2} H_s^l W_t^l)} \\ \boldsymbol{H_s^l} &= \boldsymbol{g_s(H^l, A_s)} = \boldsymbol{\hat{D}_s^{-1/2} \hat{A}_s \hat{D}_s^{-1/2} H^l W_s^l} \end{aligned} \tag{3.2}$$

where W_s^l and W_t^l represents the spatial and temporal weight metrics of the l^{th} convolution layer, respectively. In comparison, the original STGCN reduces to

$$\boldsymbol{H^{l+1}} = \boldsymbol{g(H^l, A_s)} = \boldsymbol{\sigma(\hat{D}_s^{-1/2} \hat{A}_s \hat{D}_s^{-1/2} H^l W_s^l W_t^l)} \tag{3.3}$$

due to the fixed grid-like temporal connections.

Note that the original STGCN requires fixed feature length across all graph nodes, which may not hold for our applications where nodes of different types may require different feature vectors to characterize (e.g., features from Situation Recognition are of length 1024 while appearance features from Faster-RCNN [165] are of length 2048). To address the problem of varied feature length, one easy solution is to include an additional convolutional layer to convert features with varied length to fixed length (see Figure 3.3(a)). However, we argue that nodes of different types may require different length to embed different amounts of information. Converting

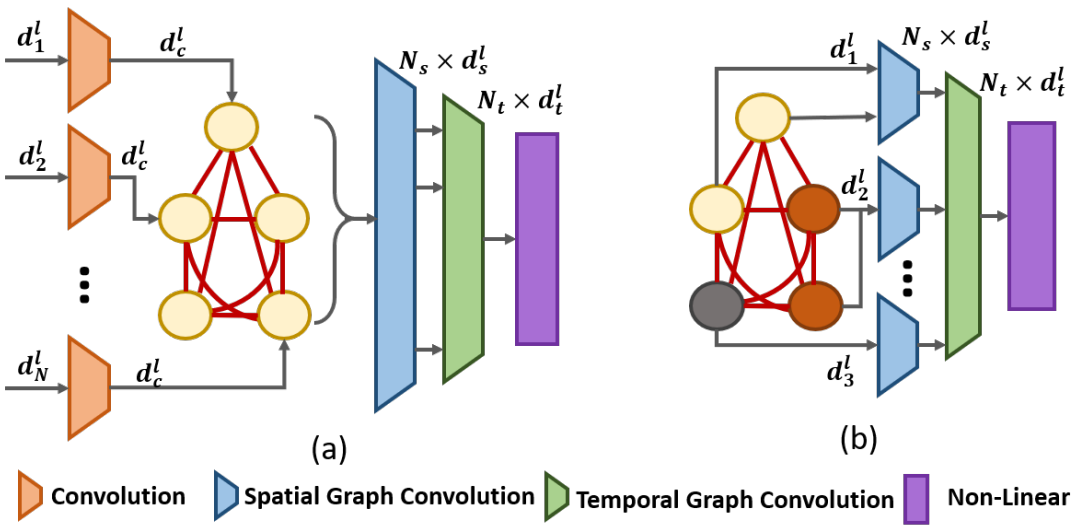

Figure 3.3: Illustration of two STGCN implementations to support graph nodes with varied feature length. (a) Additional convolution layers to convert node features with varied length to a fixed length. (b) Multiple spatial GCNs each for one cluster of nodes (nodes with the same color) with a similar feature length. These spatial GCNs convert features with varied length to a fixed length.

features to a fixed length may decrease the amount of information they can carry. Therefore, we group nodes into clusters based on their feature length and design multiple spatial GCNs, each corresponding to one of the node cluster. These spatial GCNs will convert features to a fixed length (see Figure 3.3(b)).

Notably, the S-RNN is developed for action recognition in [62] where node RNN and edge RNN are used iteratively to process graph-like input. In comparison, our model features a single graph network that can jointly process node features and edge connectivity in an interconnected manner. This, therefore, leads to improved performance and robustness.

3.3 Stacking of hourglass STGCN

Hourglass networks consist of a series of downsampling and upsampling operations with skip connections. They follow the principles of the information bottleneck approach to deep learning models [166] for improved performance. They have also been shown to work well for tasks such as human pose estimation [7], facial landmark localization [8], etc. In this work, we incorporate the hourglass architecture with STGCN so as to leverage the encoder-decoder structure for action segmentation with improved accuracy.

Our Stacked-STGCN extends and adapts the hourglass structure, commonly applied to data with regular grids (e.g., images), to data with irregular connections (e.g., graphs). This entails the development of new techniques: 1) non-symmetric encoding and decoding since feature pooling on graphs is only required in encoding stage and 2)adjustment of the dimensions of the spatial and temporal adjacency matrices accordingly. Our deliberate design of Stacked-STGCN stemming from 1) and 2) above tackle the difficulties in adapting the traditional hourglass to data with irregular connections and produce consistent performance improvement. To the best of our knowledge, extending/adapting the hourglass structure to spatiotemporal graphs at multiple spatial and temporal resolutions has not been attempted before.

Particularly, our GCN hourglass network contains a series of STGCN layer followed by a strided convolution layer as the basic building block for the encoding process. Conventional deconvolution layers comprise the basic unit for the decoding process to bring the spatial and temporal dimensions to the original size. Figure 3.4

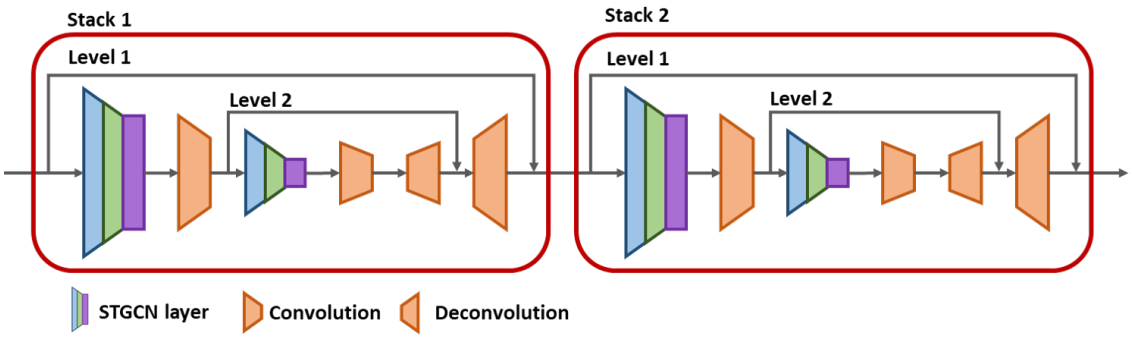

Figure 3.4: Illustration of stacked hourglass STGCN with two levels.

depicts an example with two levels.

Note that, at each layer of STGCN, the dimension of the spatial and temporal adjacency matrices, A_s and A_t, needs to be adjusted accordingly to reflect the downsampling operation. Take the illustrative example in Figure 3.4 for instance and assume that the adjacency matrices A_t and A_s are of size $N_t \times N_t$ and $N_s \times N_s$, respectively, at level 1 and that a stride of two is used. At level 2, both A_t and A_s are sub-sampled by two and their dimensions become $N_t/2 \times N_t/2$ and $N_s/2 \times N_s/2$, respectively. Due to the information compression enabled by the encoder-decoder structure, using hourglass networks leads to performance gain compared to using the same number of STGCN layers one after another.

3.4 Experiments

3.4.1 CAD120

Dataset. The CAD120 dataset is one of the more simplistic datasets available for activity recognition [167]. It provides RGBD Data for 120 videos on 4 subjects

as well as skeletal data. We use the 10 actions classes as our model labels including reaching, moving, pouring, eating, drinking, opening, placing, closing, scrubbing and null.

The CAD120 dataset splits each video into segments of the above mentioned actions. For each segment, it provides features for object nodes, skeleton features for actor nodes, and spatial weights for object-object and skeleton-object edges. Across segments, it also provides temporal weights for object-object and actor-actor edges. The object node feature captures information about the object's locations in the scene and the way it changes. The Openni's skeleton tracker [168] is applied to RGBD videos producing skeleton features for actor nodes. The spatial edge weights are based on the relative geometric features among the objects or between an object and the actor. The temporal edge weights capture the changes from one temporal segment to another.

Implementation. We exploited all the node features and edge weights provided by the CAD120 dataset. The skeleton feature of an actor node is of length 630 and the feature of an object node is of length 180. We pass each of these descriptors through convolution layers to convert them to a fixed length of 512. The initial learning rate is 0.00035 and the learning rate scheduler has a drop rate of 0.9 with a step size of 1. While experimentation, four fold cross-validation is carried out, where videos from 1 of the 4 people are used for testing and the videos from the rest three for training.

Results. For the CAD120 dataset, the node features and edge weights are provided by the dataset itself. The same set of features are used by S-RNN [62]

and Koppula et al [167, 169] who used spatio-temporal CRF to solve the problem. The S-RNN trains two separate RNN models, one for nodes (i.e., nodeRNN) and the other for edges (i.e., edgeRNN). The edgeRNN is a single layer LSTM of size 128 and the nodeRNN uses an LSTM of size 256. The actor nodeRNN outputs an action label at each time step.

Method	**F1-score (%)**
Koppula et al. [167, 169]	80.4
S-RNN w/o edge-RNN [62]	82.2
S-RNN [62]	83.2
S-RNN(multitask) [62]	82.4
Ours (STGCN)	**88.5**

Table 3.1: Performance comparison based on the F1 score using the CAD120 dataset. Our STGCN improves the F1 score over the best reported result (i.e., S-RNN) by approximately 5.0%.

In Table 3.1, we show some of the previous results, including the best reported one from S-RNN, as well as the result of our STGCN. The F1 score is used as the evaluation metric. Our STGCN outperforms the S-RNN by about 5.3% in F1 score. Instead of using two independent RNNs to model interactions among edges and nodes, our STGCN collectively performs joint inference over these inherently interconnected features. This, therefore, leads to the observed performance improvement.

In Figure 3.5, we can see a couple of errors in the second and third examples. The third prediction is 'opening' instead of 'moving' in the second example. The previous action is 'reaching' which is generally what precedes 'opening' when the actor is standing in front of a microwave and looking at it. So probably that is

the reason for the observed erroneous detection. Also the ninth frame is classified 'reaching' instead of 'moving'. If we look at the ninth frame and the eleventh frame, everything appears the same except for the blue cloth in the actor's hand. Our STGCN failed to capture such subtle changes and therefore predicted the wrong action label.

Figure 3.5: Action segmentation results of our Stacked-STGCN on the CAD120 dataset. Green: correct detection and red: erroneous detection.

3.4.2 Charades

Dataset. Charades is a recent real-world activity recognition/segmentation dataset including 9848 videos with 157 action classes, 38 object classes, and 33 verb classes [170, 171]. It contains both RGB and flow streams at a frame rate of 24fps. It poses a multi-label, multi-class problem in the sense that at each time step there can be more than one action label. The dataset provides ground-truth object and verb labels as well as FC7 feautres for every 4^{th} frame obtained from a two-stream network trained on Charades. The entire dataset is split into 7985 training videos

Description
Scene Features N1. FC7 layer output of VGG network trained on RGB frames
Motion Features N2. FC7 layer output of VGG network trained on flow frames
Segment Features N3. I3D pre-final layer output trained on RGB frames N4. I3D pre-final layer output trained on flow frames
Actor Features N5.GNN-based Situation Recognition trained on the ImSitu dataset
Object Features N6. Top 5 object detection features from Faster-RCNN

Table 3.2: Features for the Charades dataset.

and 1863 testing videos.

Implementation. For the Charades dataset, we explored two types of features, one based on VGG [172] and the other based on I3D [11], for the scene nodes in our spatio-temporal graph. Further, we used the GNN-based situation recognition technique [94] trained on the ImSitu dataset [173] to generate the verb feature for the actor nodes. The top five object features of the Faster-RCNN network [165] trained on MSCOCO are used as descriptors of the object nodes. In total, the spatial dimension of our STG is eight. The VGG features are of length 4096, the verb features 1024, and the object features 2048. Each of these channels are individually processed using graph convolution layers to convert them to a fixed length (e.g., we used 512). Table 3.2 summarizes these features.

In this experiment, spatial nodes are fully connected and temporal edges allow connections across three time steps, i.e., at the t^{th} step there are edges from t, to

$t+1$ and $t+2$ and $t+3$. The spatial edges between nodes are given a much smaller weight than self connections. We used a stack of three hourglass STGCN blocks. Before applying the normalized adjacency matrix, the input is also normalized by subtracting the mean. The output of the final Stacked-STGCN block is spatially pooled and passes through a fully connected layer to generate the probability scores of all possible classes. Since the Charades is a multi-label, multi-class dataset, the binary cross-entropy loss was used. We used an initial learning rate of 0.001 and a learning rate scheduler with a step size of 10 and a drop rate of 0.999.

To further improve action segmentation performance on Charades, we have also used a trained I3D model on Charades to generate descriptors for the scene nodes replacing the VGG features. These feature descriptors are of length 1024. Since I3D already represents short-term temporal dependencies, one block of hourglass STGCN is sufficient for capturing long-term temporal dependencies. The initial learning rate was 0.0005 and the learning rate scheduler was fixed at a drop rate of 0.995 at a step size of 10.

During training, we chose our maximum temporal dimension to be 50. If the length of a video segment is less than 50, we zero-pad the rest of the positions. But these positions are not used for loss or score computation. If the length of a video segment is greater than 50, we randomly select a starting point and use the 50 consecutive frames as the input to our graph.

At test time, we used a sliding window of length 50. Based on overlapping ratios, we applied a weighted average over these windowed scores to produce the final score. We used an overlap of 40 time steps. Following instructions in the Charades

dataset, we selected 25 equally spaced points from the available time steps in the video, to generate the final score vectors.

Ablation Studies. As to the Charades dataset, the mean average precision (mAP) is used as the evaluation metric. For fair comparison, we have used the scripts provided by the Charades dataset to generate mAP scores. We examined the performance of Stacked-STGCN using two types of descriptors for the scene nodes, namely frame-based VGG features and segment-based I3D features (see Table 3.2). We summarize our ablation studies in Table 3.3

(A1)	All Features; Baseline	7.67
(A2)	All Features; STGCN	9.22
(A3)	VGG-RGB; STGCN; 1 time step	6.33
(A4)	VGG-RGB; STGCN	6.54
(A5)	All Features; Stacked-STGCN; 1 time step	10.93
(A6)	VGG-RGB; Stacked-STGCN;	7.91
(A6)	VGG-RGB+VGG-Flow; Stacked-STGCN	10.94
(A7)	**All Features; Stacked-STGCN**	**11.73**

Table 3.3: Comparison of our Stacked-STGCN (A7) with baseline (A1), STGCN without hourglass (A2), different temporal connections (A3-A5), and different input features (A6). Input features include VGG-RGB for scene, VGG-Flow for motion, Situation Recognition for action, and Faster RCNN for object.

We first examine the performance improvement introduced by structured inference of contextual information represented in spatio-temporal graphs. We implemented a baseline method (A1) in Table 3.3 which employs a Fully Connected layer for joint inference of multiple types of features. We compare our Stacked-STGCN (A7) with this baseline (A1) and demonstrate an improvement of 4.06% .

We also compare our Stacked-STGCN (A7) with an implementation without the hourglass structure (A2) and demonstrate an improvement of 2.51% in Table 3.3.

For fair comparison of this experiment, we design a network (A2) with the same number of convolutional layers as the encoder of our Stacked-STGCN. To maintain the same temporal resolution, these convolution layers have a stride of one, compared to a stride of two in the Stacked-STGCN.

We further implement a network that closely resembles the original STGCN: 1) nodes are represented by the same type of features (i.e.,VGG-RGB); 2) pure graph convolutional operations (i.e., without hourglass); and 3) temporal connections across one time step. Comparing to this vanilla implementation (A3), our Stacked-STGCN (A7) produces an improvement of 5.40% in Table 3.3.

Next, we conduct a study on the performance of Stacked-STGCN with different input features. With one, two and four types of features, the performances are 7.91, 10.94, and 11.73, respectively, in Table 3.3 (A6, A7). This steady improvement is due to more context gained from enriched input features.

Finally, we study the performance of our Stacked-STGCN with different temporal connections. Comparing (A7) vs. (A5) in Table 3.3, temporal connections with three time steps demonstrate an improvement of 0.80%. With a simpler network (i.e., without hourglass), we observe an improvement of 0.21%, (A4) vs. (A3). The optimal number of time steps can vary depending on networks and applications. The empirical optimal number for our Stacked-STGCN on Charades is three.

Comparison with State-of-the-art. In Table 3.4, the performance of Stacked-STGCN is compared with a baseline, which uses two-stream VGG or I3D features directly for per frame action label prediction, an LSTM-based method, and the Super-Events approach proposed in [1]. Our Stacked-STGCN yields an approx-

imate 2.41% and 3.20% improvement in mAP using VGG features only and all four types of features, respectively. Using I3D features, our Stacked-STGCN ranks the second.

Method	**VGG mAP**	**I3D mAP**
Baseline [1]	6.56	17.22
LSTM [1]	7.85	18.12
Super-Events [1]	8.53	**19.41**
Stacked-STGCN (VGG only)	**10.94**	
Stacked-STGCN (all features)	**11.73**	
Stacked-STGCN (I3D)		19.09

Table 3.4: Performance comparison based on mAP between our Stacked-STGCN and the best reported results published in [1] using the Charades dataset. Our Stacked-STGCN yields an approximate 2.41% and 3.20% improvement in mAP using VGG features only and all four types of features, respectively.

Method	**mAP**
Random [174]	2.42
RGB [174]	7.89
Predictive-corrective [175]	8.90
Two-Stream [174]	8.94
Two-Stream + LSTM [174]	9.60
Sigurdsson etal. standard [174]	9.69
Sigurdsson etal. post-processing [174]	12.80
R-C3D [176]	12.70
I3D [11]	17.22
I3D +LSTM [1]	18.10
I3D+Temporal Pyramid [1]	18.20
I3D + Super-events [1]	**19.41**
I3D +Stacked-STGCN (Ours)	**19.09**

Table 3.5: Performance comparison based on mAP with previous works using the Charades dataset.

In Table 3.5, we compare the performance of Stacked-STGCN against some selected works on Charades. We can see that our Stacked-STGCN outperforms all the methods except for the I3D+super-events [1], which employs an attention

mechanism to learn proper temporal span per class. We believe that incorporating such attention mechanism could further improve the performance of our Stacked-STGCN. Furthermore, our method provides a principled way of structured inference over heterogeneous features, which most of the listed methods are incapable of. Another set of results on Charades is from the workshop held in conjunction with CVPR 2017. The results in that competition appear better. However, as mentioned in [1], that competition used a test set that is different from the validation set we used for performance evaluation. Besides those techniques could have used both the training and validation sets for training.

Chapter 4: All About Knowledge Graphs for Actions

After successfully using GCNs for action segmentation we also apply it for zero/few-shot action recognition. This chapter contains a detailed description of our system from Figure 4.1. We use the GCN layer described in equation 3.1 from [34] and the pipeline from zero shot learning technique by Wang et al. [17]. It consists of training and testing phases as described next.

Training: Initially, a model pre-trained on Kinetics is fine-tuned using training classes of UCF101, HMDB51, or Charades, followed by the extraction of the final classifier layer weights to be used for training the GCN. The constructed KG, along with the adjacency matrix, are inputs to the GCN. The output of each node of the GCN has the same dimensions as the trained classifier layer filter size (1024 in our case). The GCN is trained such that its output for the training classes matches the classifier layer weights of the trained I3D model. The loss used is the mean squared error (MSE) loss.

So if there are C_{train} number of training classes, C_{test} number of test classes and the output feature dimension of each class is d, then the output of the GCN, H_{GCN}, is of size $(C_{\text{train}}+C_{\text{test}})\times d$. From H_{GCN}, the output dimensions corresponding to the training nodes are selected, denoted by H_{GCNTrain} with size $C_{\text{train}}\times d$. This feature is

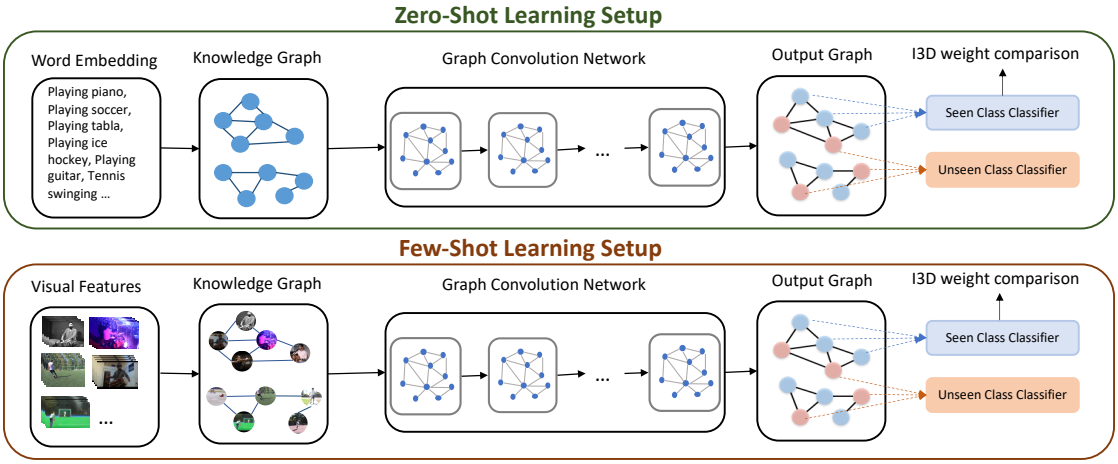

Figure 4.1: System overview: We use knowledge graphs based on word embeddings (action class names, and associated verbs and nouns) and visual features for action recognition. With the word embeddings based knowledge graph, we propose a zero-shot learning approach and with visual features based knowledge graph we propose a few-shot learning approach.

of the same dimension as the weights of the I3D classifier layer trained or fine-tuned on the training classes of the dataset, W_{cls}. The MSE loss that is back-propagated is given by $\|H_{\text{GCNTrain}} - W_{\text{cls}}\|_2$.

Testing: During test time, the penultimate layer of the I3D model is used to extract the features of the test images f_{test} with dimensions $N \times d$. The output of the test nodes of the GCN with dimension $C_{\text{test}} \times d$ is extracted from H_{GCN}, denoted by by H_{GCNTest}. The output class probabilities for the test images (P_{test}) are obtained as $P_{\text{test}} = f_{\text{test}} H^T_{\text{GCNTest}}$.

4.1 Proposed Knowledge Graphs for Actions

In this section, we describe the construction of different knowledge graphs (KGs) for actions. Wang *et al.*[17] use Wordnet [29] and NELL [30] embeddings to

construct the KG for zero-shot learning (ZSL) on image classification. Compared to [17], our action label classes are sentences or phrases instead of words, which is why using wordnet or word2vec doesn't provide distributive and coherent embeddings for action labels. Moreover, getting semantically correlated embedding space for words and visual features for a good KG is another challenge. We describe these challenges and how we tackle them while constructing three different versions of KGs for actions.

A-KG: The first KG is based on word descriptors of action class names, hence called action KG or `A-KG`. Since our action classes are composed of multiple words like a sentence or phrase, averaging word2vec embedding for all words in the sentence does not provide a cohesive embedding space. We discuss the experimental results for word2vec embeddings in Section 4.4. To overcome this challenge, we use the sentence2vec model described in [36], which is an unsupervised learning method to learn embeddings for whole sentences. We use the unigrams model trained on Wikipedia to generate our sentence embeddings.

The node features in `A-KG` are the sentence embeddings. The nodes from Kinetics action classes are added in `A-KG` corresponding to each dataset (UCF101, HMDB51, and Charades). This is inspired by [115, 116], where they show distinct advantages of adding classes and images from other datasets in ZSL. Although we cannot directly add images due to the way our model is constructed, we add new activity classes from the Kinetics dataset to increase the size of our KGs. Appending 400 Kinetics classes to UCF101 results in a total of 501 nodes in the `A-KG`. Similarly

appending the nodes to HBDB51 and Charades results in a total of 451 nodes and 557 nodes respectively. We show more results on performance comparison with and without adding Kinetics nodes in Section 4.4.

With the sentence2vector node features, we construct the `A-KG` where node i is connected to another node j in the combined dataset based on edge weights A_{ij} from cosine similarity of node features. Here, A is the adjacency matrix for `A-KG`. We sort the edges weights in descending order to get the top N closest neighbors per node. N is a hyperparameter that is determined experimentally and is dependent on the dataset. It is 5 for HMDB51 and UCF101 and 20 for Charades. j being one of the top N neighbors of i does not mean that the vice versa is true as well. To make the adjacency matrix symmetric, we fill A_{ji} with the same value as A_{ij}, so the number of connections to each node $>= N$.

VN-KG: The second graph is constructed with verbs and nouns associated with each action class, hence it is called verb-noun KG or `VN-KG`. This graph is inspired by multiple works on zero-shot action using human object interaction where the detected objects in the scene are used to draw the relationships between seen and unseen action classes [81, 119]. In [81] object detection is carried out in the visual domain as well and then mapped to word domain for ZSL. We do not do mapping for objects features from visual to word. Instead, we just take the output of verb and noun graphs (`VN-KG`), and pass it through the fusion layer to get the visual action classifier weights.

To construct `VN-KG`, we use a standard language lemmatizer [177] to break up

a phrase describing an action and convert the word to its root form. Then, we use a part-of-speech (pos) [178] tagger to label the word as a noun or a verb. Still, a lot of action class names do not have a noun in the phrase, for example "beatboxing". For such classes the pos tagger gives a noun label of "unknown" and if Wordnet can return a noun that is related to that word, we replace the "unknown" by the noun. For action classes like "archery", which does not have a specific verb associated with it, we replace the verb with "doing". For node features, we compute sentence2vec embeddings as above for verbs and nouns. Hence, we get a set of graphs with only verbs and only nouns. These also have same number of nodes as `A-KG`. Moreover, these graphs are used and categorized together as `VN-KG`, since individually they provide partial information about action class (either verb or noun). `A-KG` and `VN-KG` can be used to define ZSL setup.

V-KG: The third graph is developed to see relative performance improvements by incorporating only a few labelled images per test class. We use averaged visual features as nodes in this graph hence it is called visual feature based graph or `V-KG`. In the visual feature space, we see implicit clustering of similar actions, which is sometimes not captured in word embedding space. For example, "pommel horse" and "horse walking" are considered similar in word embedding space, but these are very different activities which is captured in visual embedding space shown for dataset UCF101 in Figure 4.2. We randomly pick 5 videos from each test class and use I3D to generate video features as described in Section 4.2.2. Then taking the mean of these features, we get the graph node descriptors and take their cosine similarity to

generate the adjacency matrix as we do for `A-KG` and `VN-KG`. This generates a graph based on visual features. `V-KG` is used to replicate few-shot learning (FSL) setup using KGs, since we use 5 visual features of each test classes to construct the nodes. In FSL setup, we can combine `V-KG` with `A-KG` and `VN-KG` to improve results.

4.2 Experimental Setup

4.2.1 Datasets

We use following four datasets, where Kinetics is just for pre-training the model, and rest are used for experiments:

Kinetics [10]: Kinetics is a large dataset with 400 classes and about $3*10^6$ videos. We do not actually need access to Kinetics videos, but the class names and an I3D model pre-trained on Kinetics available in [11]. Since we use Kinetics for pre-training I3D and data augmentation while training the GCN, we cannot keep common classes between Kinetics and UCF101 or HMDB51 or Charades in the test set while doing ZSL. So, we use classes in UCF101, HMDB51 and Charades that are also present in Kinetics, as training set.

UCF101 [179]: UCF101 has 13320 videos from 101 classes. After removing common classes with Kinetics, we get 23 classes with 3004 videos in test set for UCF101 and the remaining 78 classes are used for training. Some test class labels do not have semantically correlated neighbors. So, we appended these class names with extra words, for example "front crawl" in UCF101 becomes "front crawl swimming". We

discuss class-wise accuracy for test classes in Figure 4.4.

HMDB51 [180]: HMDB51 has 6849 videos from 51 classes. Similar to UCF101, we remove common classes with Kinetics, and get 12 classes with 1541 videos for HMDB51's test set and remaining 39 classes for training. Additionally, to encourage correlation with action classes in Kinetics, we convert the class labels to continuous tenses. For example, classes like "eat", uses sentence2vector embedding corresponding to "eating".

Charades [181]: Charades has 9848 videos from 157 classes and is also a multilabel dataset, meaning each video can have multiple action labels. Charades has noun and verb labels associated with each action class, which we use directly without labelling ourselves. After removing all videos which have at least one common label with Kinetics, we are left with 110 possible test classes. Each video can have both training and test labels in Charades. We cannot separate the training and test videos but just the classes. We split the classes into 50-50 train-test split meaning there are 79 and 78 train and test classes respectively. The 78 test classes are from the 110 classes not in common with Kinetics. All videos with at least one training class are kept in training set and we remove test class labels from them. The rest of the videos are test videos and training class labels are removed from them.

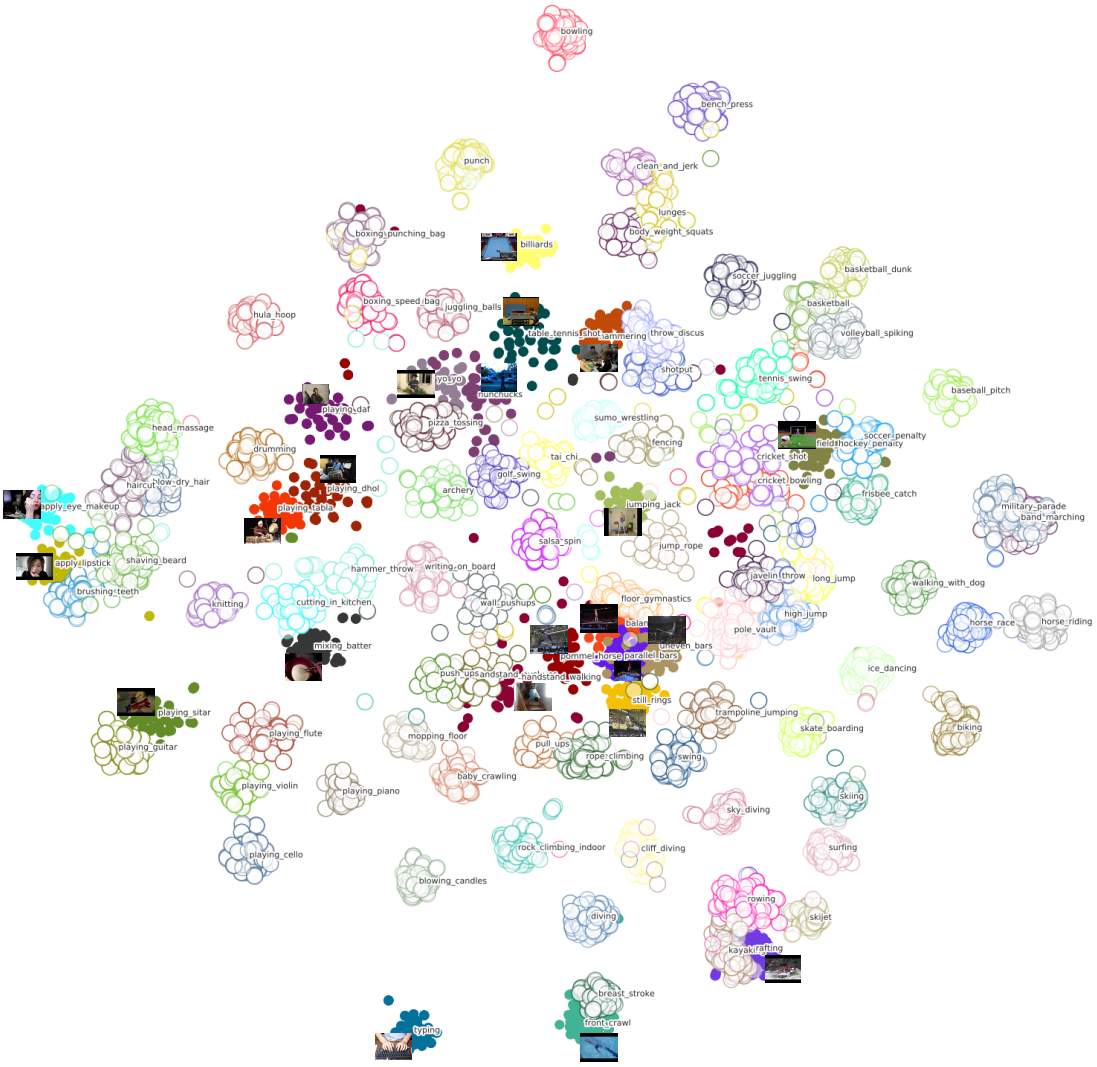

Figure 4.2: t-SNE visualization showing feature distribution of UCF101 video dataset. Sample images are added for our test classes. (Best viewed in digital format)

4.2.2 Feature Extraction

To extract *video features*, we use initial model of I3D trained on Kinetics data and fine-tune the last layer on the training classes of either UCF101 or HMDB51. For Charades, just fine-tuning the last layer did not yield good classification performance, so we fine-tune the whole network. This means while training, we cannot

compute loss on the Kinetics nodes in the KG for Charades. Even after fine-tuning the complete network for Charades we did not achieve significant performance for ZSL; so we use inverse of cross-correlation of training features added to a weighted identity matrix of the same size. This is followed by multiplication with the multiplied result of training features with training labels. This is used as the last layer weight to train GCN as inspired by [127]. We visualize the video feature space distribution of the UCF101 classes in Figure 4.2 with some example images for the test classes. As we can see in Figure 4.2, similar classes are grouped together forming clusters.

4.2.3 Our Pipeline

Our GCN consists of 6 layers with filter dimensions of $600 \rightarrow 512 \rightarrow 1024 \rightarrow 1024 \rightarrow 1024 \rightarrow 1024$. We choose 6 layers empirically. Our hypothesis is that lower depth might reduce field of view, necessary for information transfer, whereas higher depths might result in over-smoothing. The convolution kernel is of size 1. For training/fine-tuning both I3D model and the GCN model, we use ADAM optimizer with initial learning rate of 0.001. A stepwise scheduler with a drop rate of 0.99 after every 100 epochs is used for I3D training. For GCN, stepwise scheduler drop rate is 0.999 after every 100 epochs. Classwise mean accuracy is used as the evaluation metric for UCF101 and HMDB51 and mean average precision (mAP) scores for Charades. Most of the training parameters are the same for FSL setup as well, except we use a smaller learning rate of 0.00005 for UCF101.

To fuse the outputs of the different KGs, we concatenate along the channel dimension and then pass them through a GCN layer. This fusion GCN layer uses adjacency matrix of A-KG and V-KG for zero-shot and few-shot respectively. For A-KG+VN-KG in UCF101 and Charades, this fusion technique did not give good performance. So, we use the weighted sum of the outputs of A-KG and VN-KG with weights of 0.9 for A-KG and 0.05 each for the verb and noun outputs from VN-KG.

4.3 Results

Dataset	A-KG	VN-KG	A-KG+VN-KG
UCF101	49.14	45.47	**50.13**
HMDB51	38.01	31.57	**40.77**
Charades	15.81	12.48	**18.21**

Table 4.1: ZSL results for all 3 datasets where we compare performances of A-KG, VN-KG and a combination of the two. A-KG+VN-KG always does the best. For UCF101 and HMDB51, the results are in mean accuracy whereas for Charades, we report mean average precision (mAP).

The results for ZSL on all 23 test classes for UCF101, 12 test classes for HMDB51 and 78 test classes for Charades are in Table 4.1. These results are based on KGs A-KG and VN-KG and combination of both. The combination of A-KG and VN-KG graph is done through the fusion process as described in Section 4.2.3. Since all datasets have many action classes without any nouns, only VN-KG does not give good performance, but the combination of A-KG+VN-KG works well.

Method	UCF101		HMDB51		Charades
	23-78 split	50-51 split	12-39 split	25-26 split	78-79 split
ESZSL [127]	35.27	15.0	34.16	18.5	17.21
DEM [128]	34.26	-	35.26	-	-
Objects2Action [119]	-	30.3	-	15.6	-
CEWGAN [125]	-	26.9	-	30.2	-
TS-GCN [81]	44.5	34.2	-	23.2	-
Ours	**50.13**	-	**40.19**	-	**18.21**

Table 4.2: ZSL results for all 3 datasets. The baselines are ESZSL, DEM, Objects2Action, CEWGAN and TS-GCN. For UCF101 and HMDB51, the results are in mean accuracy whereas for Charades, we report mean average precision (mAP) since it is multi-label dataset.

We also provide the comparison with state-of-the-art in Table 4.2. For our data split, we have compared our results with three previous works carried out under similar ZSL settings, ESZSL [127], DEM [128] and TS-GCN [81]. We could not apply DEM baseline results for Charades, since it is a multi-label dataset. Also, TS-GCN only released code for the transductive setup for UCF101. We have implemented the inductive version and compared to it. We have also added some of the recent results for ZSL. Either their splits are different, or they do not provide code, or an essential part of their framework is missing. However, note that recent work of [81] outperforms these other approaches on their splits and we outperform [81] on our splits.

Dataset	Baseline	`V-KG`	`V-KG`+`A-KG`	`V-KG`+`VN-KG`	`V-KG`+`A-KG`+`VN-KG`
UCF101	52.7	57.04	62.10	59.92	**64.24**
HMDB	30.2	45.07	45.67	47.61	**47.69**

Table 4.3: FSL results for the UCF101 and HMDB51 datasets. The baseline is nearest neighbor, given 5 videos for each test set. The combination of `A-KG`, `VN-KG` and `V-KG` does the best in both cases.

We report results for combining `V-KG` with `A-KG` and `VN-KG` in Table 4.3. Since

we are using `V-KG`, these experiments can be considered as few shot learning. To create a baseline, we used the nearest neighbor search to get the class label for test videos. Based on the 5 labelled videos provided, we calculate the mean feature for each class and then we use cosine distances between the rest of the test videos and these class centers to sort them into corresponding classes. We use the same train-test class splits for UCF101 and HMDB51 as used in ZSL. For both UCF101 and HMDB51, we get best results if we use all 3 KGs. We do not conduct this experiment for Charades since each video has multiple labels, hence each video data point will update multiple class centers resulting in overlapping class distribution.

4.4 Analysis

Word embeddings for action labels: For constructing node features from action labels, we used the word2vec embeddings trained on Google News [20, 21, 22]. For all words in each class name, the word2vec embeddings were averaged to give a resultant embedding for the whole phrase, which serves as features of the nodes in the KG. In Figure 4.3(b), we show the word2vec embedding space of node "Pommel Horse" and its nearest neighbor class nodes.

Averaging word2Vec embedding for all words in action class label phrase works in some cases, but it cannot always capture the meaning or correct relationships between the action classes. Hence, for a class like "riding or walking with horse" in Kinetics dataset, the embedding for each word is located far apart from each other as displayed in Figure 4.3(b). The mean of these individual words does not lie close

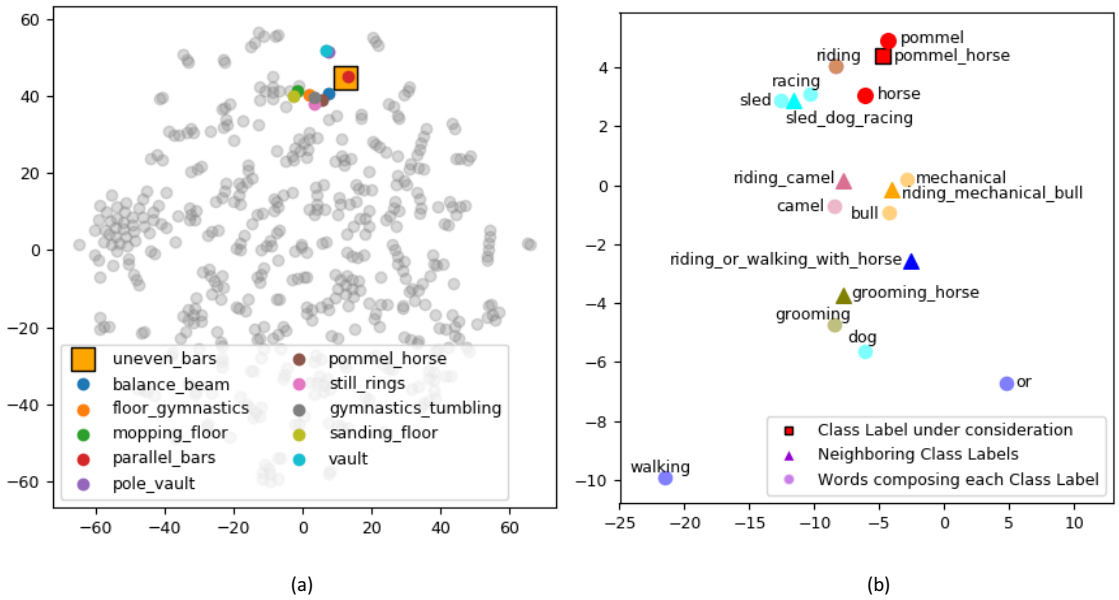

Figure 4.3: (a) Sentence2Vec embedding space for Kinetics and UCF101 classes. The class "uneven bars" and its neighbors are highlighted. (b) Class "Pommel horse" and its neighboring classes in Kinetics dataset using word2vec embedding. The embeddings of each individual word forming the phrase is also displayed. (Best viewed in digital format)

to related words in the embedding space and hence does not capture meaningful information.

Method	Mean Accuracy
Word2Vec	38.02
Sentence2vec	49.14

Table 4.4: Performance comparison between word2vec embedding and sentence2vec embedding based models. Both the models are trained on graphs consisting of class nodes from Kinetics and UCF101 (`A-KG`) with losses on both. Performance metric used is mean accuracy.

To solve this problem we use sentence2vec model from [36], which captures the semantic meaning of sequences of words. Using this embedding space, the closest word match to a class like "uneven bars" is "gymnastics tumbling". The word embedding space for all the classes in UCF101 and Kinetics are displayed in

Figure 4.3(a). The word "Uneven bars" along with its neighbors are emphasized. We run experiments with both word2vec embeddings trained on Google News [20, 21, 22] and Sentence2Vec embeddings based on unigram model trained on Wikipedia [36]. The results on UCF101 `A-KG` are shown in Table 4.4. These results show significant improvement by using setence2vec over word2vec.

Knowledge Graph	Nodes for Loss Computation	Mean Accuracy
UCF only	UCF	27.72
UCF+Kinetics	UCF	32.85
UCF+Kinetics	**UCF+Kinetics**	**49.14**
HMDB only	HMDB	31.09
HMDB+Kinetics	HMDB	29.22
HMDB+Kinetics	**HMDB+Kinetics**	**38.01**

Table 4.5: Experiments with 3 different knowledge graph constructions. The variations are due to using only UCF101/HMDB51 classes for the knowledge graph or appending it with Kinetics classes and training loss being calculated on UCF101/HMDB51 nodes only or both UCF101/HMDB51 and Kinetics nodes in the knowledge graphs (`A-KG`). Performance metric used is mean accuracy.

Appending Knowledge Graphs with more action classes: We augment the UCF101 and HMDB51 action class names based KG with Kinetics class labels in three different ways. In the first configuration, either the UCF101 nodes or HMDB51 nodes are used in the KG (101/51 nodes) out of which, 78 and 39 are training nodes respectively. The loss is computed by comparing the output of the GCN on these classes to the weights in the final classifier layer of the fine-tuned I3D network.

The second configuration uses the same KG as `A-KG` explained in Section 4.1. The loss is computed by comparing the output of only the UCF101 or HMDB51 training nodes (78/39 nodes) to the final classifier layer of the fine-tuned I3D net-

work.

In the third configuration, again `A-KG` is used. Although, now the loss is computed by summing the 2 MSE losses: (a) Loss 1 by comparing the output of only the UCF101 or HMDB training nodes(78/39 nodes) to the final classifier layer of the fine-tuned I3D network. (b) Loss 2 by comparing the output of the Kinetics nodes (400 nodes) to the classifier layer weight of I3D pre-trained on Kinetics. The results of these three experiments are shown in Table 4.5. For UCF101 and HMDB51, third configuration works best.

Types of connections in Knowledge Graphs: While constructing the `A-KG` with both UCF101 or HMDB51 and Kinetics dataset, we used two types of graph connections. In fully-connected graphs all nodes can be connected to all other nodes, out of which we select top 5 connections. In bipartite, for every node in UCF101 or HMDB51 dataset, we find the top 5 connections to the Kinetics dataset nodes and vice versa. The fully connected(FC) graph works better than the bipartite graph (Table 4.6).

Method	Mean-accuracy for UCF	Mean-accuracy for HMDB
FC	49.14	38.01
Bipartite	33.11	28.49

Table 4.6: Performance comparison for fully connected(FC) and bipartite graphs constructed with UCF101 or HMDB51 with Kinetics dataset nodes in A-KG. Both the models are trained on graphs consisting of class nodes from two datasets (UCF101 and Kinetics or HMDB51 and Kinetics) with losses on both. Performance metric used is mean accuracy.

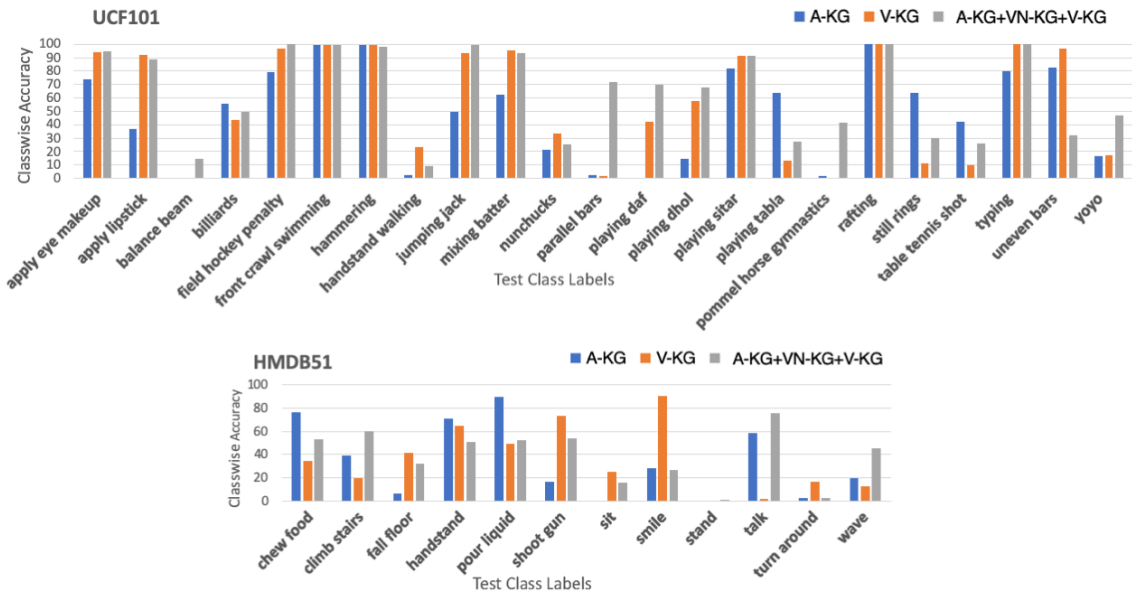

Figure 4.4: This figure shows class-wise accuracy for different KGs and combination of KGs for UCF101 and HMDB51. We added few words for better word embeddings in the labels (such as "front crawl" becomes "front crawl swimming"), which improves performance for language based KGs or their combinations, as shown here. Each color for bar represents a KG, blue is word based KG, orange is visual feature based KG and grey is combination of all three KGs (`A-KG`, `VN-KG` and `V-KG`).

Analysis of Class-wise Accuracy using different Knowledge Graphs: To understand the impact of using `A-KG`, `VN-KG` and `V-KG` for learning each test class, we plot the class-wise accuracy for UCF101 and HMDB51 in Figure 4.4. Each color of the bar represents a different KG: blue is for word based `A-KG`, orange is for visual feature based `V-KG` and grey is the combination of `A-KG`, `VN-KG` and `V-KG`.

As observed in Figure 4.4, for few classes such as "billiards", "playing tabla", `A-KG` performs the best. These classes innately have many neighbors in the word embeddings space, which help in learning them from given training classes. Few other classes, such as "front crawl swimming", "chew food" and "pour liquid" perform well with just `A-KG` as well, since we add the extra word "swimming", "food" and "liquid" respectively, to enforce good neighbors in language domain. Intuitively,

V-KG does well for "uneven bars", "fall floor", "smile" and "shoot gun", since these have distinct visual features. The combination KG works well for "parallel bars", "jumping jack", "playing daf", "playing dhol", "climb stairs", "talk" and "wave".

Ablation for Network Architecture: We experiment with different number of layers of the GCN (2,4 6, 8 and 10) to explore influence of GCN depth on performance for both UCF101 and HMDB51. The increase in the number of layers of the GCN increases smoothing and decrease in number of layers causes less information propagation. We found that 6 layers gives us the best performance.

Method	Mean accuracy
GCN	49.14
Linear Combination	42.57

Table 4.7: Performance comparison of using GCN (on UCF101 A-KG) vs a linear combination (using the adjacency matrix edge weights) of the top 4 closest training class weights to the test classes. Performance metric used is mean accuracy.

Usefulness of GCN vs a linear combination of training class weights: To show the performance improvement on UCF101 due to GCN on A-KG compared to just linear combinations, we perform an ablation study. For each test class, we find the top 4 neighbors in the training set. Then using the adjacency edge connection weights, the classifier layer weight for the test class is a weighted average of the classifier layer weights for its neighbors. The performance is in Table 4.7.

Method	Mean accuracy
without encoder-decoder	49.14
with encoder-decoder	47.72

Table 4.8: Performance comparison of using an encoder-decode layer before the GCN layers on UCF101 `A-KG` vs not using one. Performance metric used is mean accuracy.

Use encoder decoder before GCN: We run another set of experiments where a 2 layered encoder decoder network is added before GCN on UCF101 `A-KG`, for improving encoding of sentence embedding features. The results do not show any promise as seen in Table 4.8.

Method	Nodes for Loss Computation	Split 1	Split 2	Split 3	Split 4	Split 5	**Mean**
ESZSL	-	61.25	60.30	53.68	64.81	60.56	60.12
DEM	-	60.87	65.88	41.89	61.90	52.11	56.53
Ours	UCF101	59.68	48.51	42.18	49.86	43.12	48.67
Ours	UCF101+Kinetics	**83.62**	**72.60**	**71.57**	**70.85**	**49.39**	**69.61**

Table 4.9: Results on UCF101 `A-KG` with 10 randomly selected test classes leaving 91 classes to be used for training I3D and GCN. Mean accuracy is used for evaluation. The experiments are carried out 5 times and the final column provides the averaged mean accuracy scores. We compare our results to two previous work with similar settings.

Random test train splits: Some of the experiments are done on a random subsample of the test-set classes. For UCF101 `A-KG`, we choose 10 out of 23 classes 5 times; so that for each random sample of 10 test classes, the rest of the 91 classes forms the training set. The mean accuracy score is calculated after each run and the result of all 5 runs are averaged to get the final mean accuracy score. The results for each of these splits is in Table 4.9.

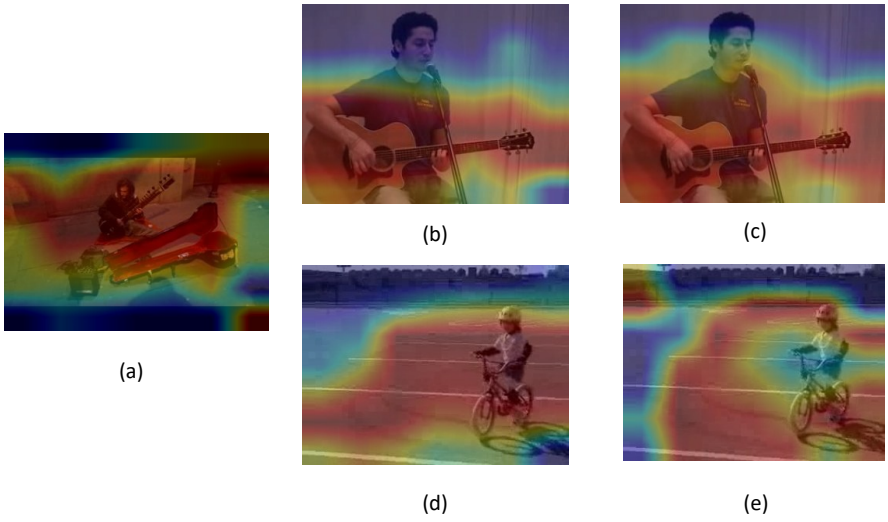

Figure 4.5: Heatmaps showing activations of various classes' classifier layers obtained from training on UCF101 `A-KG` on various class videos. (a) is the display of the activation from the "playing sitar" class on a"playing sitar" video, (b) is the display of the activation from the "playing guitar" class on a"playing guitar" video, (c) is the display of the activation from the "playing sitar" class on a"playing guitar" video, (d) is the display of the activation from the "biking" class on a"biking" video and (e) is the display of the activation from the "playing sitar" class on a"biking" video. These heatmaps show that test class "playing sitar" is correctly learning from training class "playing guitar" instead of training class "biking"

Learning classifier for unknown classes from related classes in Knowledge Graph: The heatmaps in Figure 4.5 depicts the test nodes learning from the interconnections to the train nodes in `A-KG`. They are based on CAM [182]. Considering the test class "playing sitar" in UCF101, one of the top 5 nearest train classes in UCF101 is "playing guitar" and one of the random classes that have no relation is "biking". Now among the five sub-figures in Figure 4.5, (a) is the display of the activation from the "playing sitar" class on a "playing sitar" video, (b) is the display of the activation from the "playing guitar" class on a "playing guitar" video, (c) is the display of the activation from the "playing sitar" class on a "playing guitar" video, (d) is the display of the activation from the "biking" class on a "biking" video

and (e) is the display of the activation from the "playing sitar" class on a "biking" video. What we show here is that "playing sitar" classifier is similar to the "playing guitar" classifier and hence the heat maps from both are similar. This is not the case between "playing sitar" and "biking".

Chapter 5: Learning Graphs for Knowledge Transfer with Limited Labels

We improve the GCN based semi-supervised classification and zero/few-shot action recognition performance further by learning and updating the input knowledge graph over time and also using additional constraint via triplet loss. The GCN network for semi-supervised learning is a 2 layer network based on the spectral GCN form, introduced by [34] and given in the equation 3.1. We use the same GCN framework as described in Chapter 4 for zero/few-shot action recognition. The system overview for learning the graph structure while using triplet loss is in Figure 5.1 and the algorithm specifically for zero/few-shot learning is summarized in Algorithm 1.

5.1 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 nodes lead to a better output of the GCN-based framework. 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 desirable and challenging.

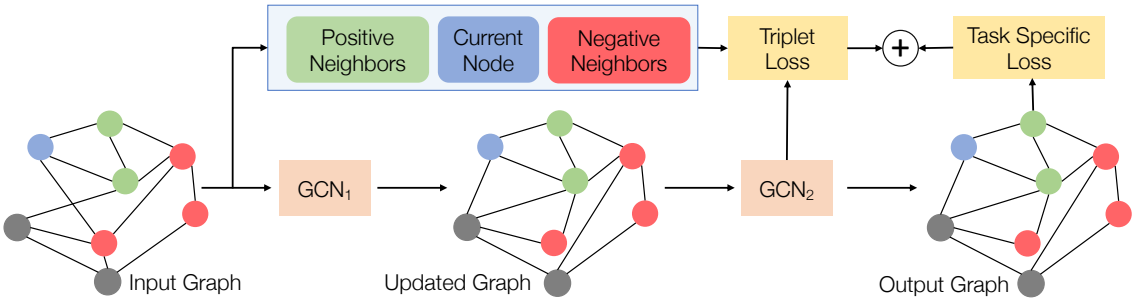

Figure 5.1: System overview for adaptive learning of graphs connections. The input graph is passed through a GCN layer and this intermediate output is used to update the graph as well as calculate a triplet loss between the current nodes and the positive and negative sets. This output is then passed 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.

Algorithm 1: System Overview for zero/few-shot learning

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

Output Classification probablity scores for all test classes (P^{test})

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

procedure GCN TRAINING AND TESTING

$A = A^{\text{in}}$

ref $\leftarrow$ example reference node

$P \leftarrow$ positive neighboring set for ref based on A^{in}

$N \leftarrow$ negative neighboring set for ref based on A^{in}

while not converge **do**

$H^{\text{inter}} \leftarrow \texttt{GCN}_1(H^{\text{feat}}, A^{\text{in}}), H^{\text{out}} \leftarrow \texttt{GCN}_2(H^{\text{inter}}, A)$

$H^{\text{train}} \leftarrow H^{\text{out}}$ for training classes,
$H^{\text{ref}} \leftarrow H^{\text{inter}}$ for ref node,
$H^{\text{P}} \leftarrow \text{mean}(H^{\text{inter}}$ for positive neighbors in $P)$,
$H^{\text{N}} \leftarrow \text{mean}(H^{\text{inter}}$ for negative neighbors in $N)$

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

Loss $\leftarrow L_{\text{MSE}} + L_{\text{triplet}} = \|W^{\text{cls}} - H^{\text{train}}\|_2 + \max(d^{\text{P}} - d^{\text{N}} + \alpha, 0), \alpha = \text{margin}$

if epoch mod $n = 0$ **then**

$A_{ij}^{\text{updated}} = \frac{H_i^{\text{inter}} \cdot H_j^{\text{inter}}}{\|H_i^{\text{inter}}\| \|H_j^{\text{inter}}\|}, A = \text{Normalize}(A^{\text{updated}})$

$H^{\text{out}*} \leftarrow$ output for optimized network,
$H^{\text{test}} \leftarrow H^{\text{out}*}$ for testing classes, $P^{\text{test}} = f^{\text{test}}(H^{\text{test}})^T$

5.1.1 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 5.1, where σ is the non-linearity following the aggregation. 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 5.2.

$$h_i^l = \sigma\left(\sum_{k \in N_i} c_{ik} h_k^{l-1} W^l\right) \tag{5.1}$$

$$c_{ij} = \text{Normalize}\left(\frac{h_i^l \cdot h_j^l}{\| h_i^l \| \| h_j^l \|}\right) \tag{5.2}$$

We denote the original adjacency matrix with A and the updated one with A_{new}. In equation 5.2 we normalize the adjacency matrix, with the node degree matrix D, given by $D^{-1/2}AD^{-1/2}$ operation from equation 3.1. $\texttt{GCN}_1$ always operates in A, whereas $\texttt{GCN}_2$ operates on A_{new}. To aid with the optimization, we update A_{new}

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 5.3).

$$A_{\text{new}} = \lambda * A_{\text{new}} + (1 - \lambda) * A \tag{5.3}$$

When we have good quality input graphs (eg., 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 (eg., those computed allegorically in Chapter4 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 5.2.

5.1.2 Training using Triplet Loss

The original network without graph learning 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. For the triplet loss, we need positive and negative sets for each node. For semi-supervised learning, each training node in the graph is a data sample and has a class label associated with it. So we can use the soft-triple loss [183], 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 5.2.

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, with triplet losses, defining negative set is more challenging. If we only use the farthest neighbors, the downstream task MSE network 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 that 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 range of the negative set classes (details in Section 5.3).

Finally, we take the average of the $\texttt{GCN}_1$ node outputs for the positive and negative sets to get positive and negative vectors. Then, the triplet loss is zero only 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, H^{ref} be the output of the current reference node, H^P and H^N are 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 5.4); and the triplet loss, L_{triplet} is calculated using equation 5.5.

$$d^P = \|H^{\text{ref}} - H^P\|_2, \qquad d^N = \|H^{\text{ref}} - H^N\|_2 \tag{5.4}$$

$$L_{\text{triplet}} = \max(d^P - d^N + \alpha, 0) \tag{5.5}$$

5.2 Experiments

Datasets: We use Citeseer, Cora and Pubmed datasets [35, 184] 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 [34, 185]. For zero/few-shot action recognition, we use Kinetics [10], UCF101 [179] and HMDB51 [180] as our datasets. Kinetics has 400 classes, UCF101 has 101 classes out of which 23 are are for test and 78 for training and HMDB51 has 51 classes out of which 12 are for test and 39 for training. We have described these datasets in Chapter 4. We make 10 random selections of c classes among test classes and we average the performance on all 10 selections 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.

Pipeline: For semi-supervised learning we use a 2 layer network where the intermediate output is used to update the graph connections. The learning rate is 0.005 for all 3 datasets. We experimentally determined the number of cluster per class for soft-triple loss and they are 2 for Pubmed and Cora and 10 for Citeseer. The rest of the hyperparameters for Soft-triple loss are the same as their paper [183]. The λ parameter in equation 5.3 is 0.8 for all datasets.

For zero/few-shot action recognition, we use an I3D [11] pre-trained on Kinetics and only finetune the last classifier layer on UCF101 and HMDB51 train-

ing classes respectively until convergence. We use 1 layer for GCN_1 and 5 layers in GCN_2 with 1 of these 5 layers belonging to fusion GCN for systems based on multiple KGs. We use the same hyperparameters as Chapter4 for our baseline network. λ from equation 5.3 is 1.0 for all zero-shot/few-shot KGs except for HMDB51 `A-KG+V-KG+VN-KG` where it is 0.5. For HMDB51 `A-KG`, we use the final output of GCN_2 to calculate A_{new} and we do not use triplet loss.

5.3 Quantitative Results

5.3.1 Semi-supervised Learning

Method	Cora	Citeseer	Pubmed
SemiEmb [186]	59.0%	59.6%	71.7%
DeepWalk [187]	67.2%	43.2%	65.3%
ICA [188]	75.1%	69.1%	73.9%
Planetoid [185]	75.7%	64.7%	77.2%
Chebyshev [98]	81.2%	69.8%	74.4%
GCN [34]	81.5%	70.3%	79.0%
MoNet [189]	81.7%	-	78.8%
GAT [104]	83.0%	72.5%	79.0%
GLNN [33]	83.4%	72.4%	76.7%†
GCN+GDC [101]	83.6%	73.4%	78.7%
H-GCN [100]	84.5%	72.8%	**79.8%**
GLCN [32]	**85.5%**	72.0%	78.3%
GCN*	80.0%	72.0%	77.8%
Ours	83.6%	**74.3%**	**79.8%**

Table 5.1: We compare accuracy of our technique to various state-of-the-art techniques for semi-supervised learning for Cora, Citeseer, and Pubmed datasets; including two graph learning techniques, GLNN and GLCN. We also provide the GCN* baseline which is our implemetation in PyTorch environment. The † in Pubmed for GLNN stands for downsampled input data. We get the best performance for both Citeseer and Pubmed datasets. For Cora, our GCN baseline (80.0%) is worse than the GCN baseline for GLCN (82.9%) by 3.0%, so the improvement using our graph learning technique is higher.

We show results on semi-supervised learning for Cora, Citeseer, and Pubmed datasets in Table 5.1. We compare against multiple state-of-the-art methods, including graph learning methods like GLCN [32] and GLNN [33]. The GCN* is our implementation of GCN [34] in PyTorch environment with 256 intermediate channels and we get slightly differing results. Since our approach builds on this baseline, we report these results 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 GCN baseline is 82.9% ($\sim$ 3.0% higher than our baseline at 80.0%).

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

Table 5.2: Ablation comparing accuracy for Pubmed validation data for different values of weighted averaging between input and updated adjacency matrix, i.e., λ from equation 5.3.

Ablation analysis. We experiment with different values of λ from equation 5.3 on Pubmed validation set and report the results in Table 5.2. We observe that $\lambda = 0.8$ achieves best performance and use this in all semi-supervised experiments.

5.3.2 Zero-shot/Few-shot Action Recognition

We compare with the results without graph learning from Chapter 4 for zero- and few-shot action recognition in Table 5.3. 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. As can be seen, our approach

of updating the graph structure during training significantly outperform our results from Chapter 4.

Input KG	UCF101		HMDB51	
	Ours	**Ours+Learning KG**	Ours	**Ours+Learning KG**
`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**

Table 5.3: Comparison of our results using graph learning with the pipeline from Chapter 4 without graph learning 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).

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 5.4: Improvements using triplet loss or updating adjacency matrix only on `V-KG` and then both together. Metric is mean accuracy (Higher is better).

Ablation analysis. We first analyze the contribution of our approach to update A and the triplet loss formulation in Table 5.4 (UCF101 using `V-KG`). We show that both contributions are better individually and are complementary to each other. Next, we study the two hyperparameters associated with these two proposals – (a) varying the number of epochs before updating adjacency matrix (n), and (b) different ordinal ranges for negative classes for the triplet loss. The results are presented in Table 5.5 and Table 5.6 respectively. We get the best performance at 30 epochs per update and negative set range of $[9, 14]$. For this ablation, we use the mean of 10 runs of randomly chosen subsets of 20 test classes.

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

Table 5.5: 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).

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

Table 5.6: 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).

Method	UCF101 23-78 split	HMDB51 12-39 split	Method	UCF101 20-81 split
ESZSL [127]	35.27	34.16	Action2vec [118]	36.5
DEM [128]	34.26	35.26	TARN [126]	42.7
TS-GCN [81]	44.5	-	SAOE[121]	51.2
Ours	50.13	40.77	UR[124]	53.8
Ours+learn KG	**53.28**	**41.05**	**Ours+learn KG**	**54.4**

Table 5.7: Comparison with State-of-the-art zero-shot action recognition results for both UCF101 and HMDB51 datasets. The results are in mean accuracy. Higher is better. We compare on the entire test set for both datasets. We also randomly choose 20 classes from UCF101 test set over 10 times and average the output to replicate the 80/20 split reported by previous work.

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 [127], DEM [128], TS-GCN [81], our own baseline without graph learning from Chapter 4, SAOE [121], UR [124], Action2vec [118], and TARN [126]. 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 and on the 80/20 split on UCF101 used by previous papers in Table 5.7. 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.

5.4 Discussion

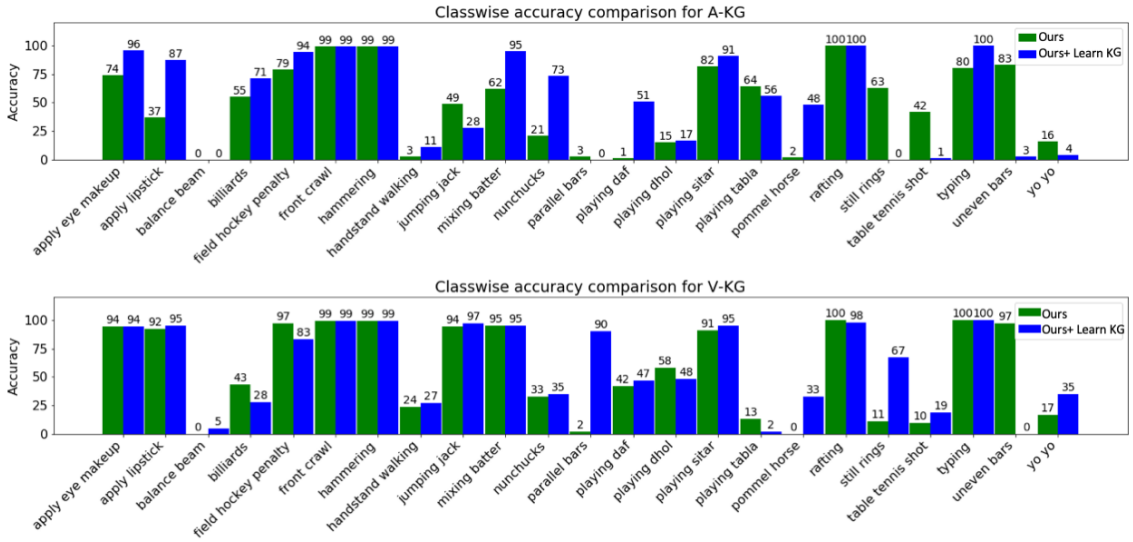

Figure 5.2: Class-wise comparison of accuracy for 23 UCF101 test classes using `A-KG` and `V-KG` as input for zero- and few-shot learning, respectively, between current results after applying graph learning (blue) and the results without graph learning i.e the baseline (green). In both cases (`A-KG` and `V-KG`), for majority of classes, we either beat or maintain the baseline performance. Best viewed in digital.

Class-wise performance. In Figure 5.2, we do a class-wise performance comparison between our output from Chapter4 without graph learning and output after learning input KG for UCF101 test classes with `A-KG` and `V-KG` as input. For zero-

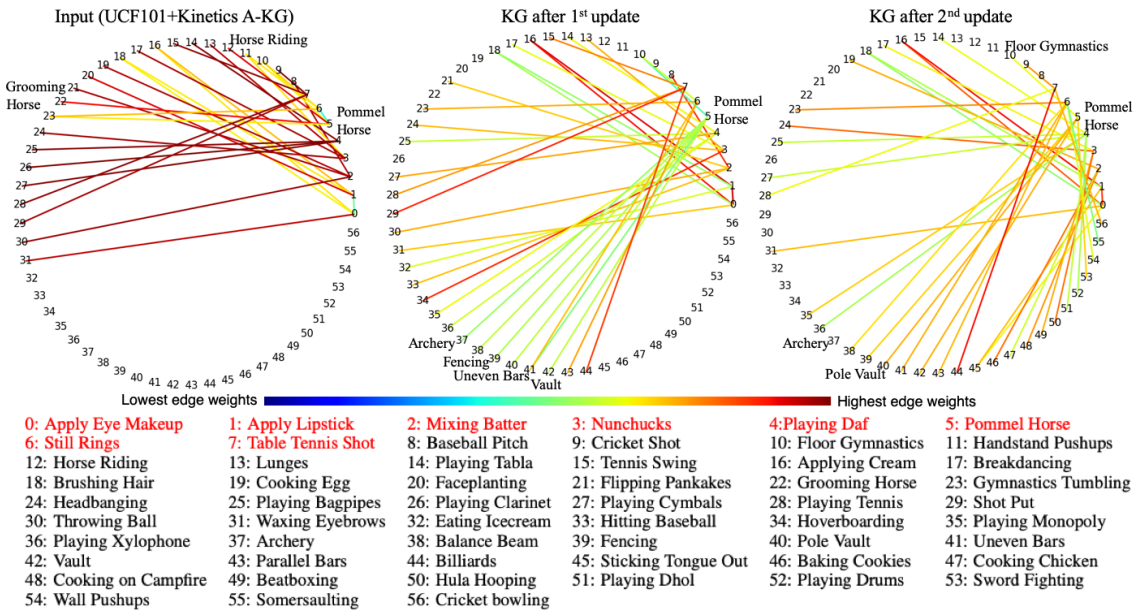

Figure 5.3: We plot the adjacency matrix connections for UCF101+Kinetics `A-KG` input and show the following two updates. We plot only a sub-graph due to space complexity. We chose 8 test classes (class names shown in red) and display all their connections in the KG. The edge colors show the weight of the connection. There are multiple regions where we can see improvements after first and second update. Best viewed in digital.

shot learning with `V-KG` as input, our technique beats the baseline for most classes (12 our of 23) like "Apply eye makeup", "Apply lipstick", "Billiards", "Nunchucks", "Playing Daf". In some cases (7 out of 23), like "Still rings", "table tennis shot", "uneven bars", we do worse. We provide an explanation for the "Still rings" class in Figure 5.4 discussed later. For few-shot learning using `V-KG`, we do better on 12 and worse on 6, and similar to fixed input graphs on 5 classes.

Qualitative results for graph updates. In Figure 5.3, we show 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 colors shown in color bar, with blue representing lower edge weights and red representing higher ones. 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 2^{nd} update at 60^{th} epoch. There are many examples where the update is improving the input KG, but due to space constraints, we only discuss one specific node here. Looking at "Pommel horse" (a gymnast action) and the input KG, we see multiple mistakes 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 from too many connections. After the 2^{nd} update, a lot of these connections (like "Archery") are removed and weight increases on connections like "Floor Gymnastics" and "Pole Vault". So overall the KG improves after each update.

Visualizing important connections. Next, we display the important graph connections with respect to the GCN network in Figure 5.4. 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. 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 5.4 labeled as linear connectivity. We also develop a way to display the closest neighbors after the GCN operation which are different from the input adjacency matrix. To do this, we follow the technique used to understand the traditional ConvNets by blocking out portions of input images [190]. If the GCN operation is

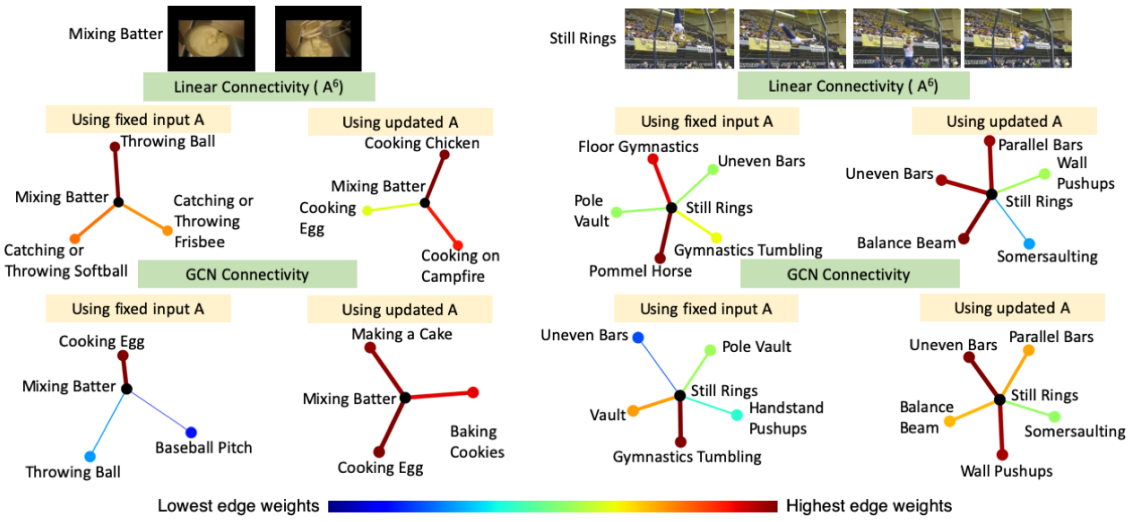

Figure 5.4: We show the connections of A^L where L is the number of layers in the GCN (linear connectivity), as well as connections after passing through the non-linear GCN network (GCN-based connectivity) for "Mixing Batter" and "Still Rings" classes. For both, we show the top-K connections using fixed input A (adjacency matrix) 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.

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 impact of connectivity between nodes n_i and the correct output class node C is given by equation 5.6, where the higher the change the more important a connection is.

$$|O - O_{\text{new}}| = |\,(G(K) - G(K - n_i)) \times f_{\text{vid}}| \tag{5.6}$$

We show GCN based connectivity, extracted using this approach, in Figure 5.4 for the two classes, "Mixing batter" and "Still Rings", using input and updated adja-

cency matrix. The edge color (depending on the given color bar) and widths represent the importance of the connectivity (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 makes a mistake by associating "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.

Chapter 6: Depth Completion Using a View-constrained Deep Prior

In this chapter we will describe in detail the technique of using the deep image prior for depth completion in stereo pipeline.

6.1 Method

Given a RGBD image with I^{in} as RGB component and D^{in} as noisy depth component, our goal is to generate denoised and inpainted depth image D^*. We leverage recently proposed Deep Image Prior (DIP) [54] to solve this problem. We first briefly describe the DIP approach.

6.1.1 Deep Image Prior

The DIP method proposed a deep network based technique for solving low level vision problems such as image denoising, restoration, inpainting, etc. At the core of their method lies the idea that deep networks can serve as a prior for such inverse problems. If x is the input image, n is the input noise and x_o is the denoised output of the network f_θ, then the optimization problem of the DIP method takes

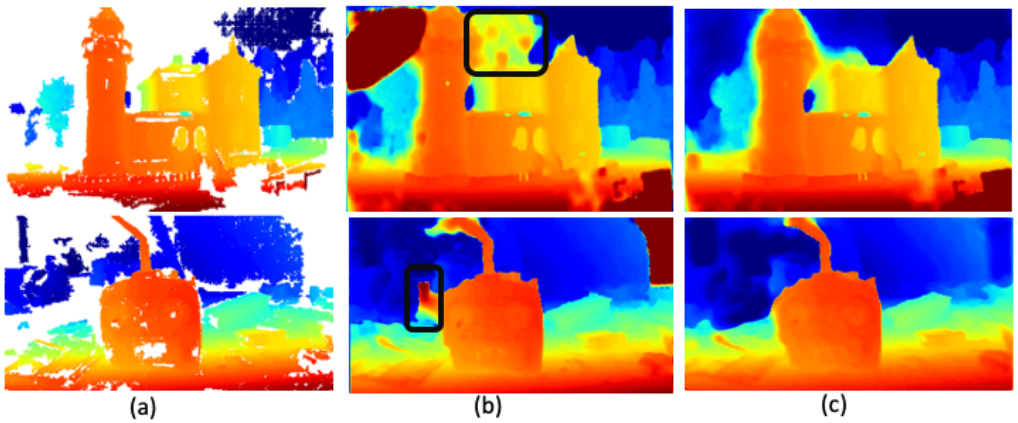

Figure 6.1: (a) Input depth map with holes (b) DDP on just depth maps and (c) DDP on RGBD images. In the black box regions in (b), DDP is filling up the holes in the sky or background based on the depth from the house or radio because it has no edge information. RGBD input provides this edge information in (c).

the following form:

$$\theta^* = \arg\min_{\theta} L(f_\theta(n); x), \quad x_o^* = f_{\theta^*}(n). \tag{6.1}$$

The task of finding the optimal neural network parameters θ^* and the optimal denoised image x_o^* is solved using the standard backpropagation approach.

A simple approach to address depth denoising and inpainting would be to use a DIP like encoder-decoder architecture to improve the depth images. Here depth images would replace RGB as inputs in the original DIP framework. However this fails to fill the holes with correct depth values. Some of the results are shown in Figure 6.1. More quantitative results are provided in Table 6.3. We hypothesize three reasons for this failure. First, holes near object boundary can cause incorrect depth filling. Second, depth images have more diverse values than RGB images that leads to large quantization errors. Finally absolute error for far objects may

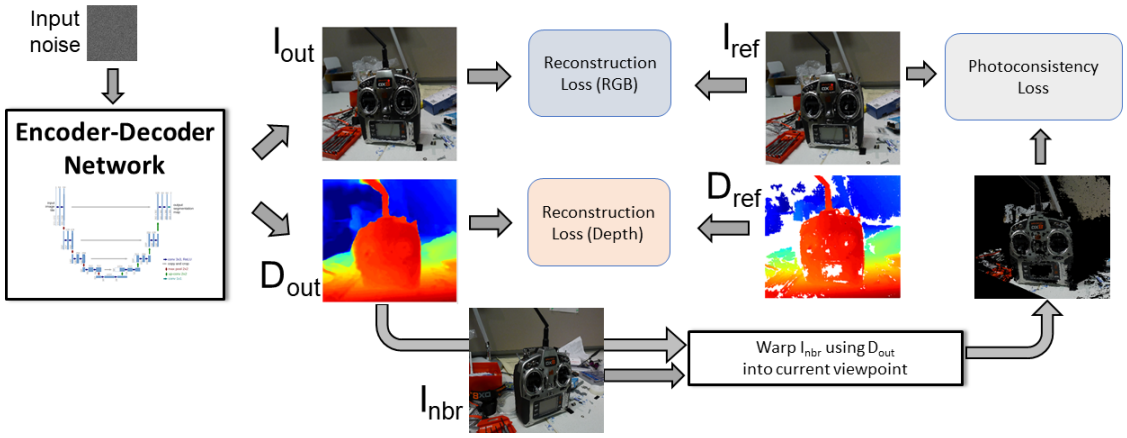

Figure 6.2: Overview of the Deep Depth Prior (DDP): The DDP network is trained using a combination of L1 and SSIM reconstruction loss with respect to a target RGB-D image and a photoconsistency loss with respect to neighboring calibrated images. This network is used to refine a set of noisy depth maps and the refined depth maps are subsequently fused to obtain the final 3D point cloud model. I^{out} and D^{out} are the RGB and depth output of our network. I^{nbr} is the RGB at a neighboring viewpoint. I^{ref} and D^{ref} are the input RGB and depth at the current (reference) viewpoint.

dominate the DIP optimization over important nearby objects.

6.1.2 Deep Depth Prior

In this work, we propose the Deep Depth Prior (DDP) and introduce three losses to solve the issues discussed before. Our approach is built on top of the inpainting task in Ulyanov et al. [54] where we create a mask for the holes in the depth map and calculate the loss over the non-masked regions. Figure 6.2 gives an overview of our system.

To solve the issue of absolute error for far objects dominating the DIP optimizer, inverted depth or disparity images are used. We also add a constant value to depth image that reduces the ratio between the maximum and minimum depth values. Further, masking of all far away objects beyond a certain depth is performed

	min depth	max depth	constant
Ignatius	2.0	7.5	0.0
Barn	2.0	16.5	2.0
Caterpillar	2.0	7.5	0.0
Meetingroom	0.2	25.0	4.0
Truck	0.5	10.0	2.0
Courtroom	0.2	46.0	4.0
Church	0.2	16.0	4.0

Table 6.1: Minimum and maximum depth clipping values and the constant depth value added per scene before doing DDP for 7 scenes in TnT dataset

by clipping to a predefined maximum depth value. We also clip depth to a minimum value so that the maximum disparity value does not go to infinity. We provide these values in Table 6.1.

Let D^{out} be the desired depth output from our network and let f_θ denote the generator network. Input to the network is noise n^{in} and the input depth map is inverted to get Z^{in} as the noisy disparity map. Let us represent the output from the network as Z^{out} where $Z^{\text{out}} = f_\theta(n^{\text{in}}; Z^{\text{in}})$. On convergence, optimal D^* is obtained by inverting Z^*.

We use three different losses to optimize our network. The total loss is defined as follows.

$$L^{\text{total}} = \gamma_1 L^{\text{disp}} + \gamma_2 L^{\text{RGB}} + (1 - \gamma_1 - \gamma_{2)} L^{\text{warp}}. \tag{6.2}$$

Disparity-based loss (L^{disp}). The simplest technique to obtain Z^* is to optimize only on disparity. The disparity based loss L^{disp} is a weighted combination of Mean Absolute Error (MAE) or L^1 loss and Structural Similarity metric (SSIM) or L^{SSIM}

loss [191], and takes the following form:

$$L^{\text{disp}} = \lambda_z L^1(Z^{\text{in}}, Z^{\text{out}}) + (1 - \lambda_z) L^{\text{SSIM}}(Z^{\text{in}}, Z^{\text{out}}). \tag{6.3}$$

We use L^1 loss instead of Mean Squared Error(L^2) loss to remove the effect of very high valued noise having a major effect on the optimization. It takes the form as $L^1(Z^{\text{in}}, Z^{\text{out}}) = |Z^{\text{in}} - Z^{\text{out}}|$. The structural similarity L^{SSIM} loss measures similarity between the input Z^{in} and reconstructed disparity map Z^{out}. Here similarity is defined at the block level where each block size is 11x11. It provides consistency at the region level. The loss (L^{SSIM}) takes the following form $L^{\text{SSIM}} = 1 - SSIM(Z^{\text{in}}, Z^{\text{out}})$. where structural similarity index (SSIM) [191] is defined by the equation 6.4,

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.4}$$

where, μ_x and μ_y are the averages of x and y, σ_x^2 and σ_y^2 are the variances of x and y, σ_{xy} is the covariance of x and y, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ where L = dynamic range of pixel values and k_1 and k_2 are constants.

RGB-based loss (L^{RGB}). We observed that under certain situations the disparity based loss leads to blurred edges in the final generated depth map. This happens when there is a hole in the depth map near an object boundary. The generator network produces a depth map that fuses the depths of different objects appearing around the hole.

For example, consider regions belonging to sky in the top row of Figure 6.1.

Due to the homogeneous nature of the sky pixels, standard disparity/depth estimation methods fail to produce any valid values for such regions. However, pixels corresponding to house region have depth values. When the image is passed to a DIP generator, the edge between the house and the sky gets blurred because the network is trying to fill up the space without any additional knowledge, e.g., boundary information. It just bases its decision on depth of neighboring space to fill up the incomplete regions as seen in the top row of Figure 6.1(b) in the black box region.

To solve this problem, we also pass the color RGB image along with the disparity image. The encoder-decoder based DDP architecture is now trained on the 4 channel RGBD image. The network weights are updated not only on the masked disparity map but also on the full RGB image. This helps the network to leverage edge and texture information for the object boundary in the RGB image to fill the holes in the disparity (and so in depth) image. This important edge information provided to the network helps to generate crisp depth images as seen in the Figure 6.1(c).

Let I^{out} be the output corresponding to input data I^{in} using the noise n^{in} and generator model f_θ. It takes the form as $I^{\text{out}} = f_\theta(n^{\text{in}}; I^{\text{in}})$. The Loss L^{RGB} is also a weighted combination of L^1 and $SSIM$ losses, and is defined as:

$$L^{\text{RGB}} = \lambda_I L^1(I^{\text{in}}, I^{\text{out}}) + (1 - \lambda_I) L^{\text{SSIM}}(I^{\text{in}}, I^{\text{out}}). \tag{6.5}$$

The RGB based loss helps to resolve the issue of blurring observed around edges near object boundaries. However, putting equal weights to disparity L^{disp} and

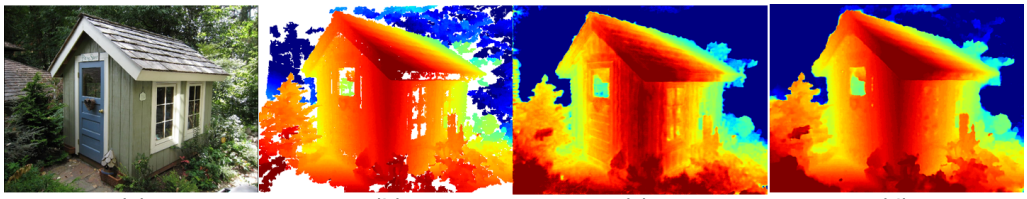

Figure 6.3: (a) RGB image (b) Input disparity map (c) Disparity output from DDP trained with equal weight for RGB and depth loss. The RGB artifacts are evident in (c) through the vertical and horizontal lines representing the wooden planks in the wall (d) Disparity output from the DDP trained with lower weight for the RGB loss compared to depth loss. The artifacts disappear in (d).

RGB L^{RGB} components of the total loss leads to artifacts appearing in the disparity image and so in depth output images as well. In particular, these artifacts are due to textures and edges from an object's appearance that are unrelated to the depth boundaries. For example in Figure 6.3 the wall of the house is one surface and should have smooth depth maps. However, the DDP network trained on RGBD data generates vertical textures in the depth images that appear due to the vertical wooden planks in the RGB image.

Warping-based loss (L^{warp}). Lastly, we include a warping loss. Before defining the warping loss, let us first define the warping function $T^{\text{ref}}_{\text{nbr}}$. Given the camera poses C_{ref} and C_{nbr} of the reference and neighboring view, the function $T^{\text{ref}}_{\text{nbr}}$ warps neighboring view to reference view.

We are trying to generate denoised output Z^{out} for the reference view. We first find top N neighboring views of the reference view using the method used in MVSNet [49]. Let *nbr* denote one of these N views and let $W^{\text{ref}}_{\text{nbr}}$ be the warped image from neighboring view to reference view. Let $D^{\text{out}}_{\text{ref}}$ be the predicted depth (inverted $Z^{\text{out}}_{\text{ref}}$) for the reference view and $I^{\text{in}}_{\text{nbr}}$ is the input RGB for the neighboring

view, then the warped image is $W^{\text{ref}}_{\text{nbr}} = T^{\text{ref}}_{\text{nbr}}(D^{\text{out}}_{\text{ref}}, I^{\text{in}}_{\text{nbr}}; C_{\text{nbr}}, C_{\text{ref}})$. Further, we use bilinear interpolation while warping.

Now given $I^{\text{in}}_{\text{ref}}$, the input RGB for the reference view, we can compute warping loss as:

$$L^{\text{warp}}_{\text{nbr-ref}} = \lambda_w L^1(I^{\text{in}}_{\text{ref}}, W^{\text{ref}}_{\text{nbr}}) + (1 - \lambda_w) L^{\text{SSIM}}(I^{\text{in}}_{\text{ref}}, W^{\text{ref}}_{\text{nbr}}). \tag{6.6}$$

When there are multiple neighboring views, the loss is averaged over them as $L^{\text{warp}}_{\text{ref}} = \frac{1}{N} \sum_{\text{nbr}=1}^{N} L^{\text{warp}}_{\text{nbr-ref}}$.

The warping loss not only resolves the issue of artifacts appearing around edges within objects, it helps to improve the accuracy of the disparity (and depth) values in other regions as well. The importance of each loss term is explored in section 6.2.

Optimization. All three losses that we use are differentiable with respect to the network parameters and so the network is optimized using standard backpropagation.

6.2 Experiments

We demonstrate the effectiveness of our proposed approach on two different tasks - 1) depth completion and 2) depth refinement. We also show the generalization ability of our technique on new datasets with unseen statistical distributions. We evaluate results on five different datasets: 1) Tanks and Temples(TnT) [146], 2) KITTI stereo benchmark [144], 3) Our own collected videos, 4) NYU depth V2 [192] and 5)Middlebury Dataset [193, 194]

6.2.1 Tanks and Temples

We first evaluate the effectiveness of the presented approach on the multi-view reconstruction task. We demonstrate the qualitative and quantitative improvement on seven sequences from the Tanks and Temples dataset (TnT dataset) [195]. These sequences include Ignatius, Caterpillar, Truck, Meetingroom, Barn, Courthouse, and Church.

Implementation. In this work, we applied the deep depth prior on a sequence of depth images of a static scene. In all our experiments, the base network is primarily an encoder-decoder based UNet architecture [196]. The UNet encoder consists of 5 convolution blocks each consisting of 32, 64, 128, 256, and 512 channels. Each convolution operation uses 3x3 kernels. We have also conducted experiments using skip networks [54]. The input noise to the network is of size $m \times n \times 16$, where m and n are the dimensions of the input depth images. Training is performed using the Adam optimizer. Initial learning rate is set to 0.00005, that is reduced by a factor of 0.01 after 12000 and 15000 epochs. The model is trained for 16000 epochs. Hyper-parameters of the loss function are set through searching on a small subset (10 images randomly chosen) from each of Ignatius and Barn sequence in TnT.

The depth images are fused using the approach proposed by Galliani et al. [197] (Fusibile) to reconstruct the final 3D point cloud. Fusibile has hyperparameters that determine the precision and recall values for the resultant point cloud. These parameters include the disparity threshold and the number of consistent views. The

disparity threshold determines the threshold of difference in disparity that is allowed for two points from two different depth maps to be merged. The number of consistant images shows the number of images in which a point has to have consistant depth values for it to appear in the merged point cloud.

Quantitative results. Our primary comparison is with the popular semi-global matching (SGM) method [4] for depth image estimation. SGM is an optimization based method that does not need any training data. We also compare with a state-of-the-art learning based method: MVSNet [49]. Further, it should be noted that our approach is agnostic to the depth estimation method, i.e., it can be used to improve depth maps from any source.

We compare our reconstructed point clouds to the ground truth point clouds for all 7 sequences in the TnT dataset [195], using the benchmarking code included with the dataset, which returns the precision ($\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$), recall ($\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$) and f-score ($\mathrm{F} = 2.\frac{\mathrm{P.R}}{\mathrm{P+R}}$) values for each scene given the reconstructed point cloud model and a file containing estimated camera poses used for that reconstruction. Here, TP, FP, and FN are true positives, false positives, and false negatives respectively. For each of the sequences we report values at the same points of the precision-recall curve as specified by the TnT dataset.

Ablation studies. We conduct a series of experiments on the Ignatius dataset to study impact of each parameter and component of the proposed method. We first study the effect of number of depth images on the final reconstruction. For this

Dataset	SGM	SGM+ Zhang *et al.*	SGM+ DDP (Ours)
Ignatius(87 images)	45.3	45.2	45.6
Ignatius(44 images)	44.0	44.0	44.2
Ignatius(22 images)	30.7	30.7	31.1

Table 6.2: We compare the f-score of DDP on datasets of different sizes. As the number of images become smaller, the holes increase and the most relative performance gain is at 22 images. We also compare to [2] applied to an data that is out of distribution and show we do better. (Higher is better)

purpose, we skip a constant number of images in the dataset. For example, a skip size of 2 means that only every second image is used for reconstruction giving us 87 images in total for Ignatius. Such reduction in the data size increases the number of holes on the SGM-based reconstruction method. The same reduced dataset is used for the baseline and our approach. One of the goals of this work is for our accurate hole-filling to allow for fewer images to be captured and used for reconstruction. We also conducted experiments with skip values of 4 and 8 giving 44 and 22 images. Table 6.2 provides details about the impact of this skip size on relative improvement. For SGM+DDP, we keep SGM values where there are no holes, and replace the holes with DDP output values in those regions. It can be observed that at skip sizes of 2, 4 and 8, we see an improvement of 0.3, 0.2 and 0.4 percentage point of relative improvement in f-scores over the baseline respectively. It suggests that at higher skip sizes, our approach provides necessary prior information to fill holes. Please note that we have not included experiments with skip size of one. Running Fusibile on all of the Ignatius images produces dense reconstruction without holes. Running DDP on this dense reconstruction has no effect, and so we have not included results with skip size of one in the table. Table 6.2 also contains results in comparison to

Zhang and Funkhouser [2] which we will discuss later.

Network + Loss	P	R	F
UNet + D	34.3	37.2	35.7
UNet + RGBD	39.2	49.0	43.6
UNet + RGBD + warp	40.0	50.6	**44.7**
SkipNet + RGBD + warp	40.2	50.3	44.7

Table 6.3: The results of comparing the DDP output using disparity, RGBD, and warping loss and using UNet or SkipNet as the network are shown here, P:precision, R:recall, F:f-Score. (Higher is better)

Next we show the advantage of using the RGBD and warping losses over disparity loss only. Running the optimization for too many epochs on the depth only DDP can return the holes in the result, thus we run the depth only based networks for 6000 epochs instead of 16000. The results are in Table 6.3, showing a 7.9% improvement in f-measure by using RGBD. Table 6.3 also shows the benefit of photo-consistency based warping loss. We observe a relative improvement of 1.1% in f-measure after incorporation of the warping loss. Finally, we also conducted experiments with Skip-Net [54] to show the impact of using a network different from UNet. Table 6.3 shows they give comparable results, and we chose to use UNet for our other experiments since it is a more commonly used network.

Comparison to prior work. Quantitative comparison with the SGM baseline on the 7 TnT dataset is shown in the Table 6.4. The SGM+DDP (Ours) column shows the results of using DDP to improve depth maps from SGM. We combine DDP depth maps with SGM depth maps, keeping the SGM values everywhere where there are no holes and replace the holes with DDP depth values. The table includes the precision,

				SGM			SGM+DDP(Ours)		
Data	N	C	D	P	R	F	P	R	F
I	87	5	1.0	41.7	49.5	45.3	41.2	51.1	**45.6**
I	22	2	2.0	32.7	29.0	30.7	32.2	30.1	**31.1**
B	180	2	0.5	23.3	27.8	25.4	22.8	29.3	**25.6**
B	45	1	4.0	19.1	21.4	20.2	18.1	22.8	**20.2**
T	99	4	1.0	35.8	38.8	37.2	34.5	41.4	**37.6**
T	25	1	2.0	29.4	33.8	31.5	27.7	36.7	**31.6**
C1	156	4	1.0	24.9	41.5	**31.1**	24.0	42.9	30.8
C1	39	1	2.0	17.3	36.5	**23.5**	16.2	37.9	22.7
MR	152	4	1.0	27.5	13.4	18.1	25.2	15.2	**19.0**
MR	38	1	4.0	17.8	9.0	12.0	15.7	10.7	**12.7**
CH	110	2	1.0	1.8	0.8	1.1	3.2	1.2	**1.7**
C2	86	4	1	8.9	8.5	8.7	8.9	8.6	**8.8**

Table 6.4: Quantitative results comparing 7 sequences for SGM based depths and applying the DDP on SGM depths. We combine DDP with SGM by replacing the depth values in the holes of SGM depth with DDP depth. Here the datasets are I:Ignatius, B: barn, T: truck, C1: caterpillar, MR: Meetingroom, CH: courthouse and C2: church. Also N: number of images in sequence, C: number of consistent views while constructing point cloud, D: disparity threshold, P:precision, R:recall, F:f-score. (Higher is better)

recall and f-scores on each dataset with sub-sampled number of images using the technique described in the ablation studies sub-section. We observe improvement in both recall and f-score values of the presented approach over the SGM-method. It suggests the method helps in hole filing. The f-score improves in 6 of the 7 sequences. Overall SGM+DDP(Ours) helps to improve recall by 1.5 percentage points and f-score by 0.2 percentage points. In particular, we see a significant improvement of 1.8 in recall and 0.8 in f-score on the MeetingRoom sequence.

We also tested using the learning-based MVSNet method as an input to our method. While the results from MVSNet do not contain any holes, as they predict a value at every pixel, just as we do, they do have areas where the predictions have low confidence. We use their output probability map which shows confidence of depth

Dataset	MVSNet			MVSNet + DDP		
	P	R	F	P	R	F
I (126 images)	45.9	52.2	48.8	45.2	54.9	**49.6**
I (32 images)	40.3	46.8	43.3	38.9	50.0	**43.8**

Table 6.5: We compare the reconstruction performance using depth maps generated by MVSNet and by applying the DDP on MVSNet. I: Ignatius, P:precision, R:recall, F:f-score. (Higher is better)

prediction and remove depths at places with confidence below a certain threshold (0.1) to see if we can fill those areas more accurately than MVSNet did. We then fill up these holes using DDP and compare the results with the original. These results for Ignatius are in Table 6.5. We see 0.7% improvement in F-score and 3.0% in recall.

Finally we also compare to Zhang and Funkhouser [2] in Table 6.2. We used the trained networks provided by Zhang and Funkhouser using SUNCG-RGBD [198] and ScanNet [199] datasets for inpainting on Ignatius dataset. As we can see here this method does not work well because it is a learning based system, Ignatius is an out-of-distribution test data and the model would require finetuning. This emphasizes the usefulness of our network being optimized on each image separately and it being independent of training data statistics.

Qualitative results. Next we provide visual results on the TnT dataset to highlight the impact of our approach in achieving high quality reconstruction. In Figure 6.4, we show output disparity images at 16000 (column c) epochs of our proposed approach on 5 TnT sequences. Note that the holes in the input disparity images (marked as white in column b) are filled in the output images (c).

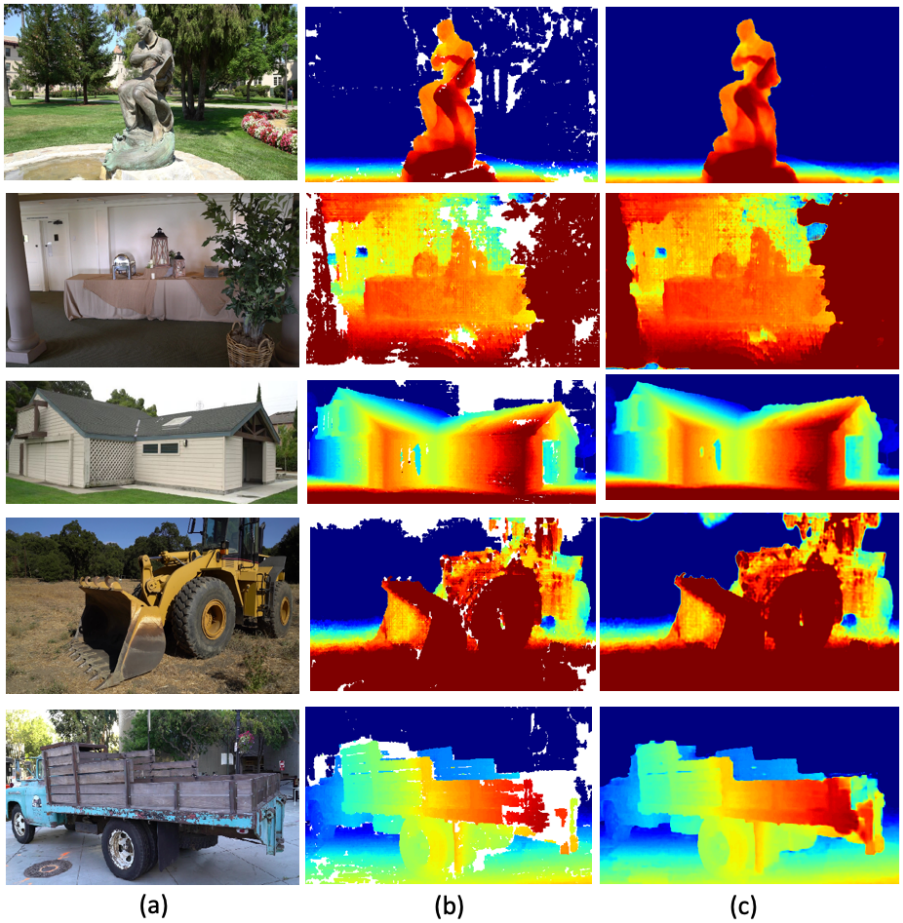

Figure 6.4: (a) Input RGB and (b) Input depth images and (c) Predicted depth at 16000 epochs for Ignatius, Meeting Room, Barn, Caterpillar and Truck.(Best viewed in digital. Please zoom in.)

6.2.2 KITTI

Next we show the performance of the proposed approach on the KITTI stereo benchmark (2015) [200] for ELAS [201] based disparity maps in Table 6.6. Traditional stereo methods like ELAS works without dependence on training data distributions unlike state of the art deep learning based systems. The evaluation metric used in this benchmark is D1 that measures the error percentage when the predicted value differs from the groundtruth value by 3px or 5% or more. This error is

separately measured for background, foreground, and all regions together. The two approaches we used for generating input depths for DDP are as follows:

1. We use first set of parameters for ELAS to generate disparity maps without any holes and DDP for refinement and completion. We observe a small performance improvement (∼0.1% reduction in D1-all error).

2. We use a second set of parameters for ELAS to generate depth images with holes. In this case, DDP applied over ELAS depth helps to inpaint and complete the ELAS depth with a 8.4% reduction in D1-all error.

	Method	D1-bg	D1-fg	D1-all
params1	ELAS	7.5	21.1	9.8
	ELAS + DDP	7.5	21.1	9.7
params2	ELAS	20.6	28.7	22.0
	ELAS + DDP	12.3	20.2	13.6

Table 6.6: Results on KITTI Dataset using D1 error. Lower is better.

6.2.3 Our Dataset

We used an off-the-shelf consumer camera to capture 5 scenes, both indoor and outdoor. The datasets are monocular image sequences and are named Guitar, Shed, Stones, Red Couch, and Van.

To better understand the quality of the depth images generated by the proposed method, we warp RGB images using the original and the proposed DDP based depth images for Shed and Stones from our video sequences and Truck from TnT dataset. The warped RGB images are shown in the Figure 6.5. Holes can be

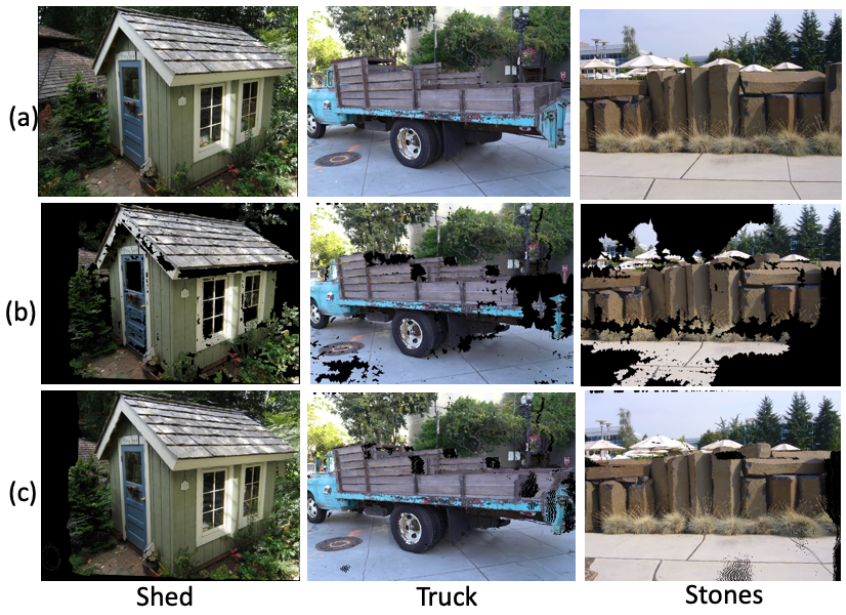

Figure 6.5: (a) Original image from the reference view point (b) Novel view synthesized from neighboring to reference viewpoint using the original SGM depth (c) Novel view synthesis from a neighboring to reference viewpoint using DDP depth. The holes that appear in (b) gets filled in (c) (Best viewed in digital. Please zoom in.)

observed in the RGB images warped using original depth maps for example along the roof of the shed, some parts of the truck and along the base of the Stone Wall. However, RGB images warped using our depth images removes large portions of these holes and are far smoother.

We show reconstructed point clouds on RedCouch, Guitar and Van in Figure 6.6. The first row is one of the input RGB images used for the reconstruction, second row is the reconstruction from the original SGM output, the third one is from SGM + DDP, and the fourth row is the reconstructed result from MVSNet.

The number of views used for these reconstructions are small (∼10). We chose these datasets to show specific ways in which our reconstructions improve on the original SGM and even MVSNet outputs. As we can see from our results, there are

Figure 6.6: (a) Input RGB image. Reconstructed point-cloud from (b) SGM (c) DDP (Ours) and (d) MVSNet depth images for RedCouch, Guitar and Van. Our reconstructions are better and more complete. (Best viewed in digital. Please zoom in.)

a lot less holes in the reconstructions computed from our depth maps compared to original SGM and MVSNet. For example from the top view of the RedCouch in the first column, we can see the relatively obscure region behind the pillow. The DDP successfully fills up a big portion of this hole. Next the reflective surfaces of the guitar in the second column and the van in the third column also get completely or at least partially filled depending on how big the original hole was. For example note how the text "FREE" on the van is more readable in the DDP result.

6.2.4 NYU v2

We demonstrate performance of our approach on hole-filling Kinect depth data. For this, we use data from the popular NYU v2 dataset [192]. Qualitative experiments on the NYU v2 depth images are shown in Figure 6.7. We cannot directly use the test data from NYU, since we need at least two views as input. So we downloaded 3 video scenes, and used a structure-from-motion (SfM) pipeline [202] to get the extrinsic camera pose information. Using this, we apply DDP to remove holes in the Kinect depth maps given by the dataset. We directly refined the Kinect depth images, and as the originals are incomplete, we cannot use them for performance analysis. Instead we use the depth maps to project one RGB view to another in the video sequences to compute and RGB re-projection error, which we quantify with Peak Signal to noise ratio (PSNR). It is defined as $PSNR = 20\log_{10}(MAX_I/\sqrt{MSE})$, where MAX_I is the maximum value of the image and MSE is the mean squared error of the image. As we can see in Fig-

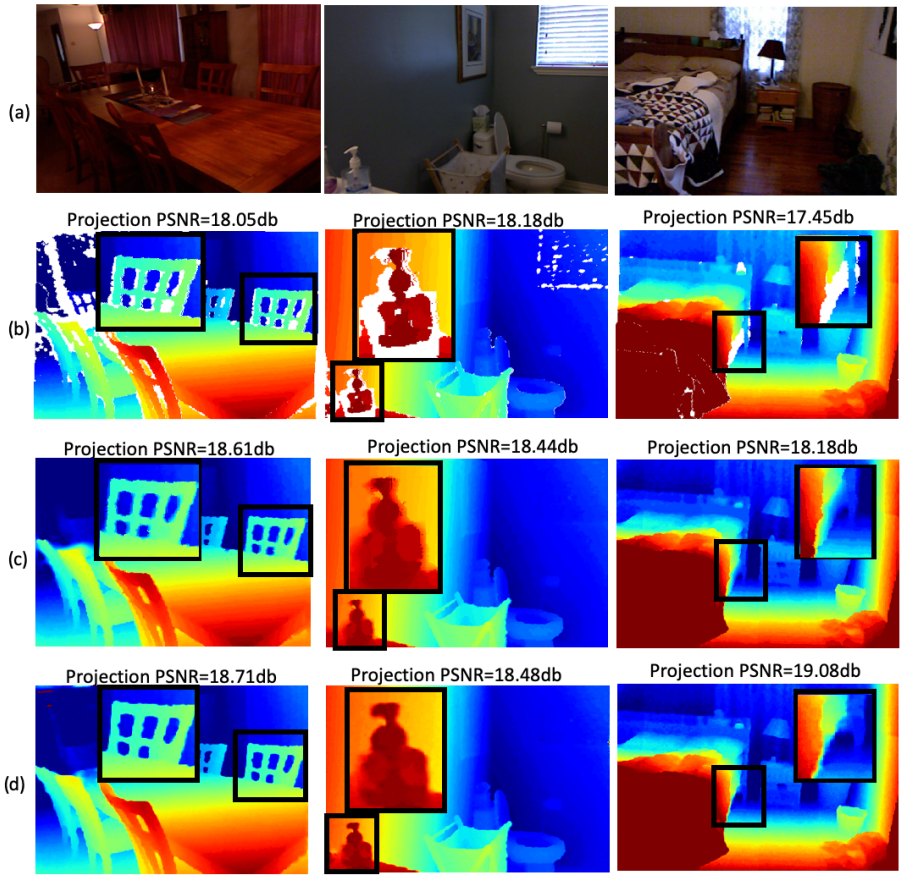

Figure 6.7: (a) Input RGB (b) Input depth (c) Depth completed using a cross-bilateral filter and (d) Depth completed using DDP (Best viewed in digital. Please zoom in.)

ure 6.7, DDP fills up holes and improves the PSNR. We also use the cross-bilateral hole filled depth maps included in the NYU v2 dataset as a baseline. We observe consistent qualitative and quantitative improvements using DDP compared to the cross-bilateral method.

6.2.5 Middlebury

Finally we compare to the baseline techniques described in Depthcomp [3] for Middlebury Dataset scenes [193, 194] in Table 6.7. We took the input images and disparity maps from Depthcomp [3] with the holes they created artificially in the

Method	Plastic	Baby	Bowling	Average
SSI [203]	1.7573	2.9638	6.4936	*3.74*
Linear Inter	1.3432	1.3473	1.4503	*1.38*
Cubic Inter	1.2661	1.3384	1.4460	*1.35*
FMM [204]	0.9580	0.8349	1.2422	*1.01*
GIF [205]	0.7947	0.6008	0.9436	*0.78*
FBF [206]	0.8643	0.6238	0.5918	*0.69*
EBI [207]	0.6952	0.6755	0.4857	*0.62*
DepthComp [3]	0.6618	0.3697	0.4292	*0.49*
DDP (Ours)	**0.4951**	**0.3232**	**0.5743**	**0.46**

Table 6.7: Comparison of our method with techniques mentioned in [3] on Middlebury dataset using RMSE. (Lower is better)

disparity maps and compared to the baseline techniques they mention. The metric is Root Mean Square Error (RMSE) which is $||D_{\text{out}} - D_{\text{GT}}||_2$ where D_{out} is the output disparity and D_{GT} is the ground-truth disparity. Our average result is better than all of the other techniques in Table 6.7 for depth completion.

Chapter 7: Conclusion

The proposed Stacked-STGCN in Chapter 3 introduces a stacked hourglass architecture to STGCN for improved generalization performance and localization accuracy. Its building block STGCN is generic enough to take in a variety of nodes/edges and to support flexible graph configuration. We applied our Stacked-STGCN to action segmentation and demonstrated improved performances on the CAD120 and Charades datasets. We also note that adding spatial edge connections between nodes from same model lead to performance improvement on Charades rather than across different feature nodes. This is mainly due to the oversimplified edge model (i.e., with fixed weights). Instead of using a binary function to decide on the correlation between these nodes, more sophisticated weights could be explored. We leave this as future work. Furthermore, graph representation based on actor, action, object and scene provides inherent explanations of corresponding detection of action categories. Such explanation requires visualizing the traces of most activated nodes/edges, which we also leave as future work. Finally, we anticipate that due to its generic design Stacked-STGCN can be applied to a wider range of applications that require inference over a sequence of graphs with heterogeneous data types and varied temporal extent.

In Chapter 4 we investigate different combinations of knowledge graphs (KG) for actions that give better performance for zero and few shot action recognition. We show significant improvement on zero shot learning by using a network that models a sequence of words instead of traditional single word based models. Moreover, extending KG using other action classes leads to better results. We observe that combining word based knowledge graphs with visual knowledge graphs help in few shot learning. Also combining verbs and noun based KG, improves both zero and few shot learning.

In Chapter 5 we experiment with adaptive learning of adjacency matrix and constraining neighbors in a KG through triplet loss based training in addition to task specific loss like MSE loss. We show visually how the graph connectivity changes with each update. We use previous research on convolutional networks to develop an understanding of how the GCN itself modifies the input connectivity vs just displaying the input adjacency matrix connections. Our results beat the state of the art on many standard datasets for semi-supervised learning and zero/few-shot learning by a significant margin.

In Chapter 6 we have presented an approach to reconstruct depth maps from incomplete ones. We leverage the recently proposed idea of utilizing a neural network as a prior for natural color images, and introduced three new loss terms for depth map completion. Extensive qualitative and quantitative experiments on sequences from the Tanks and Temples, KITTI, NYU v2, Middlebury, and our own dataset demonstrated that the depth maps generated by our method were more accurate. An important future extension is improving the speed of the method, where an efficient

version of the presented approach could be used for a real-time depth enhancement and 3D reconstruction pipeline. Further, the presented method could benefit from deeper understanding of the convergence properties of training deep image priors.

Overall we have investigated different aspects of holistic scene understanding including action recognition and depth perception. Future work can consider combination of these field like using improved depth perception for a simultaneous localization and mapping (SLAM) system that can help tracking action over longer periods of time.

Bibliography

[1] AJ Piergiovanni and Michael S Ryoo. Learning latent super-events to detect multiple activities in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 4, 2018.

[2] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018.

[3] Amir Atapour-Abarghouei and Toby P Breckon. Depthcomp: real-time depth image completion based on prior semantic scene segmentation. 2017.

[4] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[5] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

[6] 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.

[7] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[8] Jing Yang, Qingshan Liu, and Kaihua Zhang. Stacked hourglass network for robust facial landmark localisation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 2025–2033. IEEE, 2017.

[9] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[10] 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.

[11] 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.

[12] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. Temporal 3d convnets: New architecture and transfer learning for video classification. *arXiv preprint arXiv:1711.08200*, 2017.

[13] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[14] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[15] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[16] Xiang Xiang, Ye Tian, Austin Reiter, Gregory D Hager, and Trac D Tran. S3d: Stacking segmental p3d for action quality assessment. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 928–932. IEEE, 2018.

[17] 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.

[18] Dedre Gentner. Some interesting differences between verbs and nouns. In *Cognition and brain theory*, volume 4, pages 161–178, 1981.

[19] Pallabi Ghosh, Nirat Saini, Larry S. Davis, and Abhinav Shrivastava. All about knowledge graphs for actions, 2020.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[21] 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.

[22] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[23] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[24] 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*, pages 4444–4451, 2017.

[25] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

[26] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009.

[27] Genevieve Patterson and James Hays. Coco attributes: Attributes for people, animals, and objects. In *European Conference on Computer Vision*, pages 85–100. Springer, 2016.

[28] Phillip Isola, Joseph J Lim, and Edward H Adelson. Discovering states and transformations in image collections. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1383–1391, 2015.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[35] 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.

[36] 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.

[37] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[38] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1434–1441. IEEE, 2010.

[39] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[40] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[41] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2115–2128, 2009.

[42] Michael Bleyer, Carsten Rother, Pushmeet Kohli, Daniel Scharstein, and Sudipta Sinha. Object stereo-joint stereo matching and object segmentation. In *CVPR 2011*, pages 3081–3088. IEEE, 2011.

[43] Tatsunori Taniai, Yasuyuki Matsushita, and Takeshi Naemura. Graph cut based continuous stereo matching using locally shared labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1613–1620, 2014.

[44] Tatsunori Taniai, Yasuyuki Matsushita, Yoichi Sato, and Takeshi Naemura. Continuous 3d label stereo matching using local expansion moves. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2725–2739, 2017.

[45] Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016.

[46] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[47] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017.

[48] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2315, 2017.

[49] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

[50] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[51] C. Zhou, H. Zhang, X. Shen, and J. Jia. Unsupervised learning of stereo matching. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1576–1584, Oct 2017.

[52] Alessio Tonioni, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Unsupervised adaptation for deep stereo. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1605–1613, 2017.

[53] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[54] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, pages 9446–9454, 2018.

[55] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *CVPR*, pages 10130–10139, 2019.

[56] Yossi Gandelsman, Assaf Shocher, and Michal Irani. " double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *CVPR*, 2019.

[57] Pallabi Ghosh, Vibhav Vineet, Larry S. Davis, Abhinav Shrivastava, Sudipta Sinha, and Neel Joshi. Depth completion using a view constrained deep prior, 2020.

[58] Patrick Knobelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-end training of hybrid cnn-crf models for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2339–2348, 2017.

[59] Youze Xue, Jiansheng Chen, Weitao Wan, Yiqing Huang, Cheng Yu, Tianpeng Li, and Jiayu Bao. Mvscrf: Learning multi-view stereo with conditional random fields. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4312–4321, 2019.

[60] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[61] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[62] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[63] Xikun Zhang, Chang Xu, and Dacheng Tao. Graph edge convolutional neural networks for skeleton based action recognition. *arXiv preprint arXiv:1805.06184*, 2018.

[64] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. *arXiv preprint arXiv:1806.01810*, 2018.

[65] Effrosyni Mavroudi, Divya Bhaskara, Shahin Sefati, Haider Ali, and René Vidal. End-to-end fine-grained action segmentation and recognition using conditional random field models and discriminative sparse coding. *arXiv preprint arXiv:1801.09571*, 2018.

[66] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1970, 2016.

[67] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018.

[68] Colin Lea Michael D Flynn René and Vidal Austin Reiter Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[69] Li Ding and Chenliang Xu. Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation. *arXiv preprint arXiv:1705.07818*, 2017.

[70] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6508–6516, 2018.

[71] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6742–6751, 2018.

[72] Earl Rennison. Constructing a search query to execute a contextual personalized search of a knowledge base, January 11 2011. US Patent 7,870,117.

[73] David L Gilmour and Eric Wang. Method and apparatus for constructing and maintaining a user knowledge profile, July 16 2002. US Patent 6,421,669.

[74] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015.

[75] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 956–966, 2014.

[76] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee, 2017.

[77] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 851–861, 2015.

[78] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[79] Antoine Bordes and Evgeniy Gabrilovich. Constructing and mining web-scale knowledge graphs: Kdd 2014 tutorial. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1967–1967. ACM, 2014.

[80] Sutanay Choudhury, Khushbu Agarwal, Sumit Purohit, Baichuan Zhang, Meg Pirrung, Will Smith, and Mathew Thomas. Nous: Construction and querying of dynamic knowledge graphs. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1563–1565. IEEE, 2017.

[81] 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.

[82] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, 1998.

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

[84] Fereshteh Sadeghi, Santosh K Kumar Divvala, and Ali Farhadi. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1456–1464, 2015.

[85] Yuan Fang, Kingsley Kuan, Jie Lin, Cheston Tan, and Vijay Chandrasekhar. Object detection meets knowledge graphs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1661–1667, 2017.

[86] Zhanglin Peng, Lingyun Wu, Jiamin Ren, Ruimao Zhang, and Ping Luo. Cuimage: A neverending learning platform on a convolutional knowledge graph of billion web images. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1787–1796. IEEE, 2018.

[87] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2027–2034, 2014.

[88] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *European conference on computer vision*, pages 408–424. Springer, 2014.

[89] Rob Fergus, Hector Bernal, Yair Weiss, and Antonio Torralba. Semantic label sharing for learning with many categories. In *European Conference on Computer Vision*, pages 762–775. Springer, 2010.

[90] Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR 2011*, pages 1481–1488. IEEE, 2011.

[91] Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 666–672. IEEE Computer Society, 2005.

[92] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, (1):61–80, 2009.

[93] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[94] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation recognition with graph neural networks. *arXiv preprint arXiv:1708.04320*, 2017.

[95] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: an efficient subgraph-based framework for scene graph generation. In *European Conference on Computer Vision*, pages 346–363. Springer, 2018.

[96] 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.

[97] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[98] 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.

[99] 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.

[100] 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.

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

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

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

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

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

[106] Michelle Guo, Edward Chou, De-An Huang, Shuran Song, Serena Yeung, and Li Fei-Fei. Neural graph matching networks for fewshot 3d action recognition. In *European Conference on Computer Vision*, pages 673–689. Springer, 2018.

[107] 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.

[108] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[109] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[110] 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.

[111] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

[112] Marcus Rohrbach, Michael Stark, and Bernt Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR 2011*, pages 1641–1648. IEEE, 2011.

[113] 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.

[114] 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.

[115] Xun Xu, Timothy Hospedales, and Shaogang Gong. Semantic embedding space for zero-shot action recognition. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 63–67. IEEE, 2015.

[116] Xun Xu, Timothy M Hospedales, and Shaogang Gong. Multi-task zero-shot action recognition with prioritised data augmentation. In *European Conference on Computer Vision*, pages 343–359. Springer, 2016.

[117] Xun Xu, Timothy Hospedales, and Shaogang Gong. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, (3):309–333, 2017.

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

[119] 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.

[120] Mihir Jain, Jan C. van Gemert, and Cees G. M. Snoek. What do 15, 000 object categories tell us about classifying and localizing actions? In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 46–55, 2015.

[121] 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.

[122] Chuang Gan, Yi Yang, Linchao Zhu, Deli Zhao, and Yueting Zhuang. Recognizing an action using its name: A knowledge-based approach. *International Journal of Computer Vision*, (1):61–77, 2016.

[123] Ioannis Alexiou, Tao Xiang, and Shaogang Gong. Exploring synonyms as context in zero-shot action recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4190–4194. IEEE, 2016.

[124] 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.

[125] Devraj Mandal, Sanath Narayan, Sai Kumar Dwivedi, Vikram Gupta, Shuaib Ahmed, Fahad Shahbaz Khan, and Ling Shao. Out-of-distribution detection for generalized zero-shot action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9985–9993, 2019.

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

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

[128] 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.

[129] 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.

[130] 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.

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

[132] 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.

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

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

[135] 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.

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

[137] 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.

[138] 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.

[139] Sai Kumar Dwivedi, Vikram Gupta, Rahul Mitra, Shuaib Ahmed, and Arjun Jain. Protogan: Towards few shot learning for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[140] Keizo Kato, Yin Li, and Abhinav Gupta. Compositional learning for human object interaction. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[141] Norbert Haala and Mathias Rothermel. Dense multi-stereo matching for high quality digital elevation models. *Photogrammetrie-Fernerkundung-Geoinformation*, 2012(4):331–343, 2012.

[142] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 873–881, Dec 2015.

[143] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014.

[144] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[145] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3260–3269, 2017.

[146] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):78:1–78:13, July 2017.

[147] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[148] Stefan K Gehrig, Felix Eberli, and Thomas Meyer. A real-time low-power stereo vision engine using semi-global matching. In *International Conference on Computer Vision Systems*, pages 134–143. Springer, 2009.

[149] Christian Banz, Holger Blume, and Peter Pirsch. Real-time semi-global matching disparity estimation on the gpu. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 514–521. IEEE, 2011.

[150] Sudipta N Sinha, Daniel Scharstein, and Richard Szeliski. Efficient high-resolution stereo matching using local plane sweeps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1582–1589, 2014.

[151] Daniel Scharstein, Tatsunori Taniai, and Sudipta N. Sinha. Semi-global stereo matching with surface orientation priors. *2017 International Conference on 3D Vision (3DV)*, pages 215–224, 2017.

[152] Akihito Seki and Marc Pollefeys. Sgm-nets: Semi-global matching with neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 231–240, 2017.

[153] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011.

[154] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1854–1861, 2013.

[155] Philipp Heise, Sebastian Klose, Brian Jensen, and Alois Knoll. Pm-huber: Patchmatch with huber regularization for stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2360–2367, 2013.

[156] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[157] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu, and Chang Huang. A deep visual correspondence embedding model for stereo matching costs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 972–980, 2015.

[158] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[159] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

[160] Patrick Knöbelreiter and Thomas Pock. Learned collaborative stereo refinement. In *German Conference on Pattern Recognition*, pages 3–17. Springer, 2019.

[161] Oleg Voynov, Alexey Artemov, Vage Egiazarian, Alexander Notchenko, Gleb Bobrovskikh, Evgeny Burnaev, and Denis Zorin. Perceptual deep depth super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5653–5663, 2019.

[162] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 887–895, 2017.

[163] Tatsunori Taniai and Takanori Maehara. Neural inverse rendering for general reflectance photometric stereo. In *ICML*, 2018.

[164] Zezhou Cheng, Matheus Gadelha, Subhransu Maji, and Daniel Sheldon. A bayesian perspective on the deep image prior. In *CVPR*, pages 5443–5451, 2019.

[165] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[166] Léonard Blier and Yann Ollivier. The description length of deep learning models. In *NIPS*, 2018.

[167] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.

[168] http://openni.org.

[169] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):14–29, 2016.

[170] Gunnar A. Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Actor and observer: Joint modeling of first and third-person videos. In *CVPR*, 2018.

[171] 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*, 2016.

[172] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[173] Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5534–5542, 2016.

[174] Gunnar A Sigurdsson, Santosh Kumar Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, volume 5, page 7, 2017.

[175] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictivecorrective networks for action detection. In *Proceedings of the Computer Vision and Pattern Recognition*, 2017.

[176] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: region convolutional 3d network for temporal activity detection. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 5794–5803, 2017.

[177] Edward Loper Bird, Steven and Ewan Klein. Natural language processing with python, 2009.

[178] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[179] 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.

[180] 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.

[181] 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.

[182] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[183] 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.

[184] 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.

[185] 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.

[186] 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.

[187] 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.

[188] Q Lu and L Getoor. Link-based classification.(2003). In *Proceeding 2003 international conference on machine learning, Washington DC*, pages 496–503.

[189] 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.

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

[191] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[192] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. Springer, 2012.

[193] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[194] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[195] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):78, 2017.

[196] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[197] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015.

[198] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[199] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[200] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[201] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38. Springer, 2010.

[202] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[203] Daniel Herrera, Juho Kannala, Janne Heikkilä, et al. Depth map inpainting under a second-order smoothness prior. In *Scandinavian Conference on Image Analysis*, pages 555–566. Springer, 2013.

[204] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[205] Junyi Liu, Xiaojin Gong, and Jilin Liu. Guided inpainting and filtering for kinect depth maps. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2055–2058. IEEE, 2012.

[206] Amir Atapour-Abarghouei, Gregoire Payen de La Garanderie, and Toby P Breckon. Back to butterworth-a fourier basis for 3d surface relief hole filling within rgb-d imagery. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2813–2818. IEEE, 2016.

[207] Pablo Arias, Gabriele Facciolo, Vicent Caselles, and Guillermo Sapiro. A variational framework for exemplar-based image inpainting. *International journal of computer vision*, 93(3):319–347, 2011.