저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 제이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer
Ph.D. DISSERTATION

Compositional Graph Neural Networks for Spatiotemporal Structure Learning in Video Data

비디오 데이터의 시공간 구조 학습을 위한 구성적 그래프 신경망

BY

Kyoung-Woon On

August 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY
Ph.D. DISSERTATION

Compositional Graph Neural Networks for Spatiotemporal Structure Learning in Video Data

비디오 데이터의 시공간 구조 학습을 위한 구성적 그래프 신경망

BY

Kyoung-Woon On

August 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY
Abstract

The cognitive abilities of human depend on the structured representations that express the world as entities and their relations, and build compositional hierarchies among the entities. With the advances in machine learning, especially deep neural networks, artificial intelligence has demonstrated human-level performance in key areas such as computer vision and natural language processing. However, simulating the ability to learn structured representations still remains as a challenging problem. Recently, Graph Neural Networks (GNNs) which learn representations of graph structured data have been proposed to learn structured representations. Nevertheless, it is difficult to learn an overall compositional hierarchy with conventional GNNs as the GNNs focus on learning representations of entities (nodes) with flat operators such as message passing.

In this dissertation, we postulate learning compositional and hierarchical structure from data as a fundamental problem, and study on methods to learn the compositional hierarchies. To this end, we divide the problem into two levels, spatial-level and temporal-level, and propose two compositional graph neural networks to learn the structured representations of spatial and temporal data, respectively. Also, we propose multilevel Video QA dataset to unify spatial and temporal-level compositional structured representation learning.

Firstly, to learn spatial-level compositional structured representations, we propose a novel graph pooling algorithm, Spectrally Similar Graph Pooling (SSGPool). The main idea of the SSGPool algorithm is to learn a coarsening matrix which maps nodes from an original graph to a smaller number of nodes in a coarsened graph. The coarsening matrix is trained based on feature vectors
of nodes while keeping the spectral characteristics of the original graph in the coarsened one. To validate the effectiveness of the SSGPool, experiments on various graph benchmarks are conducted compared to strong baselines. Also, we evaluate our approach on a real-world problem, image retrieval with visual scene graphs.

Next, to learn temporal-level compositional structured representations, we propose Cut-Based Graph Learning Networks (CB-GLNs). The CB-GLNs learn representations of video data by discovering complex dependency structures that imply variable-length semantic flows and their composition. To this end, the video data is expressed as a graph, with nodes and edges corresponding to frames of the video and their dependencies respectively. The CB-GLNs find compositional hierarchies of the video in multilevel graph forms via a parameterized kernel with graph-cut and a message passing framework. For evaluations, two different tasks for video understanding are conducted: Video theme classification and Video question answering.

Finally, we propose a multilevel Video Question Answering (Video QA) dataset to unify compositional structured representation learning. For multilevel QA, hierarchical difficulty levels are proposed with two criteria: memory capacity and logical complexity. Then the hierarchical difficulties are aligned with proposed spatiotemporal-level to construct dataset. The dataset is built upon the TV drama “Another Miss Oh” and contains QA pairs with multilevel difficulties and various length video clips. To evaluate unified learning method, we combine two compositional graph neural networks proposed above and conduct an experiment for multilevel Video QA task.

**Keywords:** Compositional structure learning, Deep neural networks, Graph neural networks, Video representation learning, Video Question Answering

**Student Number:** 2014-21783
# Contents

Abstract i

Chapter 1 Introduction 1

1.1 Motivation & Research Objectives . . . . . . . . . . . . . . . . . 1
1.2 Approaches & Contributions . . . . . . . . . . . . . . . . . . . 4
1.3 Organization of the Dissertation . . . . . . . . . . . . . . . . . 6

Chapter 2 Background 7

2.1 Structured Representations with Graphs . . . . . . . . . . . . . 7
2.1.1 Graph notations . . . . . . . . . . . . . . . . . . . . . . . . 7
2.1.2 Graph Neural Networks (GNNs) . . . . . . . . . . . . . . . 8
2.2 Spatial Structured Representations for Images . . . . . . . . . . 11
2.3 Temporal Structured Representations for Video . . . . . . . . . . 13

Chapter 3 Compositional Learning for Spatial Structure: Spectrally Similar Graph Pooling 16

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.3 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
Chapter 4  Compositional Learning for Temporal Structure: Cut-Based Graph Learning Networks

4.1 Introduction ................................................. 41
4.2 Preliminaries ................................................... 43
  4.2.1 Normalized Graph-cut ................................. 43
  4.2.2 Message Passing Neural Networks ................. 44
4.3 Problem Statement ............................................ 45
4.4 Cut-Based Graph Learning Networks ......................... 46
  4.4.1 Structure Learning Module ........................... 46
  4.4.2 Representation Learning Module ...................... 48
4.5 Experiments .................................................. 49
  4.5.1 Baselines ............................................... 49
  4.5.2 Video Theme Classification Task on YouTube-8M .... 51
  4.5.3 Video Question & Answering Task on TVQA ........... 57
Chapter 5  Dataset for Learning Spatiotemporal Structure: Multilevel Video QA  

5.1 Introduction  

5.2 Related Work  
5.2.1 Datasets for Video Understanding  
5.2.2 Cognitive Developmental Stages  

5.3 Two Criteria for Multilevel Video QA  
5.3.1 Criterion 1: Memory Capacity  
5.3.2 Criterion 2: Logical Complexity  
5.3.3 Interpretation as Cognitive Development Stage  

5.4 Multilevel Video QA Dataset  
5.4.1 Overview  
5.4.2 Question-Answer Hierarchy for Levels of Difficulty  
5.4.3 Rich Video Annotations  
5.4.4 Data Collection  

5.5 Experiments  
5.5.1 Unified Learning Framework  
5.5.2 Experimental Result  

5.6 Conclusions  

Chapter 6  Concluding Remarks  

6.1 Summary of Methods & Contributions  
6.2 Discussions for Future Directions  

초록  

Acknowledgements
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An illustration of compositional hierarchies for both spatial-level and temporal-level.</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>An illustration of Graph Convolution Networks. Figure is taken from the original paper of GCNs (Kipf and Welling, 2016).</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Examples of visual scene graph.</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>An illustrative example of compositional hierarchy in a visual scene graph. (a) is an original image and (b) is a hierarchical structure of visual scene graph for the image.</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>The architecture of SSGPool layer combined with graph neural networks. The SSGPool learns coarsening matrices $P$ to minimize task-specific loss while retaining spectral similarity. To represent the spectral similarity, we use second smallest eigenvector (Fiedler vector) of graph Laplacian.</td>
<td>23</td>
</tr>
</tbody>
</table>
Figure 3.3  Left: An original image corresponding to the scene graphs on the right. Right: Pooling results on nodes of each graph in each layer. Same color of nodes are meant to be mapped to the same coarsened node in the pooled layer. Since DiffPool coarsens the graph with soft assignment matrix, we selected a top-1 coarsened node for each original node for visualization. The grey colored nodes in layer-2 are left-over coarsened nodes that were not chosen as top-1 by any original nodes. Some significant node labels are specified to demonstrate different properties between the pooling methods.

Figure 3.4  Other qualitative examples of pooling results on SSGPool, DiffPool and SAGPool. As Figure 3.3 LaTeX Error: Can be used only in preamble See the LaTeX manual or LaTeX Companion for explanation. Your command was ignored. Type I \texttt{\textbackslash command} \texttt{\textbackslash return} to replace it with another command, or \texttt{\textbackslash return} to continue without it. 3.3, same color of nodes are meant to be mapped to the same coarsened node in the pooled layer.

Figure 4.1  (a): The overall architecture of the Cut-Based Graph Learning Networks (CB-GLNs) for a video representation learning task. (b), (c): sophisticated illustrations of Structure Learning Module and Representation Learning Module.
Figure 4.2 Examples of YouTube-8M dataset. A set of videos which has “airplane” as a theme label are shown. (https://research.google.com/youtube8m) 51

Figure 4.3 Visualizations of adjacency matrices of CB-GLNs (a and b) and attention maps of the self-attention mechanism (c and d). 54

Figure 4.4 An example of the constructed temporal dependency structures for a real input video in YouTube-8M, titled “Rice Pudding” (https://youtu.be/cD3enxnS-JY) is visualized. The topical themes (true labels) of this video are {Food, Recipe, Cooking, Dish, Dessert, Cake, baking, Cream, Milk, Pudding and Risotto}. (a): Learned adjacency matrices in the layer 1 and 2 are visualized. The strength of connections are encoded in a gray-scale where 1 is white and 0 is black. (a)-1: 12 bright blocks in layer 1 are detected (blue rectangles), each block (highly connected frames) represents a semantic unit. (a)-2: Sub-graphs of the input are denoted by orange rectangles. It shows that semantically meaningful scenes are found by temporally constrained graph-cut. (a)-3 and (a)-4: learned high-level dependency structures in layer 2 are revealed with red and green rectangles. (b): The whole composite temporal dependencies are presented. 58

Figure 4.5 Examples of TVQA dataset. Figure is taken from original paper (Lei et al., 2018) 59
Figure 4.6  An example of the learned attention hierarchy for a real video clip of “Friends” in TVQA is visualized. The question and answer for this clip are “What did Chandler say he was going to get, when he got up?” and “Chandler said he was going to get cigarettes.” each. We dropped the Scene 1 and Scene 2 parts in (a) and (b), because the attention values in these scenes are nearly identical, making them non-informative. (a): Attention maps for each layer given query. the orange rectangles in layer 1 denote the cut scenes discovered by CB-GLNs and the red rectangle in layer 2 highlights scenes with high attention values given a question. (b): Cumulative attentions to the frame-level are visualized by multiplying attention values in layer 1 & 2. (c): Visualization of the learned adjacency matrices in layer 1 & 2. The orange and red rectangles are consistent with (a).

Figure 5.1  An example of multilevel Video QA dataset which contains video clips, scripts, and QA pairs with levels of difficulty. A pair of QA corresponds to either a shot or a scene, and each QA is assigned one out of a possible four stages of difficulty. A video clip consists of a sequence of images with visual annotations centering the main characters.
Figure 5.2 Four examples of different QA level: Difficulty 1, 2, 3, 4, respectively. **Difficulty 1** and **Difficulty 2** target the length of shot video. **Difficulty 1** requires single supporting fact to answer and **Difficulty 2** requires multiple supporting facts to answer. Questions for these two levels need spatial-level reasoning process. **Difficulty 3** or **Difficulty 4** requires a time factor to answer and target the length of scene video. Especially, **Difficulty 4** requires causality between supporting facts from a different time. Questions for these two levels need temporal-level reasoning process.

Figure 5.3 (a) The number of QA pairs per episode and difficulty level. Given that the scene length is tens of times than the size of the shot, the variation between levels is small compared to the number of videos. (b) The number of 5W1H question types per difficulty level.

Figure 5.4 Examples of visual metadata and coreference resolved scripts. Our dataset provides visual metadata containing the main characters’ bounding box, name, behavior, and emotion for each frame.
Figure 5.5  (a) The percentage of each character’s frequency in visual metadata. *Haeyoung1* and *Dokyung* is two main characters of drama *AnotherMissOh*. *Haeyoung2* is the person who has same name with *Haeyoung1*, but we divided their name with numbers to get rid of confusion.  
(b) The percentage of each behavior frequency in the visual metadata. *none* behavior occupies a lot because there are many frames with only character’s face.  
(c) The percentage of each emotion frequency in the visual metadata.

Figure 5.6  (a) **Top:** Top-3 most frequently the person who the speaker talk to, for each top 6 most spoken person. **Bottom:** Top-3 most frequently the person who the speaker talks about, for each top 6 most spoken person.  
(b) The percentage of each person’s utterance in the script.

Figure 5.7  The overall architecture of unified learning framework for Video QA. The spatial-level structure learning module and temporal-level structure learning module are integrated to answer the questions for video story.
## List of Tables

| Table 3.1 | Statistics of graph benchmark datasets | 31 |
| Table 3.2 | Average accuracy and standard deviation for graph benchmarks are presented. The DiffPool* denotes DiffPool with additional losses originally proposed in (Ying et al., 2018). We highlight the best results (bold) and second best results (blue). | 32 |
| Table 3.3 | Effect of proposed regularizer in SSGPool for various benchmark datasets along with pooling ratio and the number of pooling layer. | 33 |
| Table 3.4 | Comparison of regularization in condition of 2 pooling layers along with pooling ratio. Original Reg. is the SSGPool with original regularizer and Simpler Reg. is the one with simpler regularizer. | 34 |
| Table 3.5 | Comparison of computation time (millisecond) for single graph of regularization in condition of 2 pooling layers along with pooling ratio. Original Reg. is the SSGPool with original regularizer and Simpler Reg. is the one with simpler regularizer. | 35 |
Table 3.6 The results of image retrieval in terms of NDCG. Higher the NDCG score is, better the performance. DiffPool* denotes DiffPool with additional losses originally proposed in (Ying et al., 2018).

Table 4.1 Comparison on classification accuracy with the GAP measure on validation dataset. Logistic regression is as a classifier for all of the presented methods.

Table 4.2 An ablation study of Cut-Based Graph Learning Networks.

Table 4.3 Training time of CB-GLNs in a single batch by number of cut operations.

Table 4.4 Comparison on classification accuracy with the GAP measure on validation dataset. Logistic regression is as a classifier for all of the presented methods.

Table 5.1 Comparison between video story QA datasets (MovieQA (Tapaswi et al., 2016), PororoQA (Kim et al., 2017b), TVQA (Lei et al., 2018) and TVQA+ (Lei et al., 2019)). Only DramaQA dataset provides hierarchical QAs and character based visual metadata. Average target video length for single QA is divided into shot level and scene level, since we provide multi level QA.
Table 5.2 Quantitative result on the multilevel Video QA dataset.
We divide test set by difficulty level and get performance of each set. The Overall column shows the average performance of each level. The QA-only indicates a simple MLP model that inputs questions only and Our−{SSLM, TSLM} indicates the proposed model without Spatial Structure Learning Module and Temporal Structure Learning Module respectively.
Chapter 1

Introduction

1.1 Motivation & Research Objectives

Artificial intelligence (AI) has made major advancement in key domains such as vision, language and decision-making with the success of deep learning. Many characteristics of human intelligence, however, still remain out of reach for current approaches. One of key characteristics for human intelligence that we have missed is the ability to make “infinite use of finite means” (Von Humboldt et al., 1999; Chomsky, 2014), which reflects the principle of combinatorial generalization. Human’s ability for combinatorial generalization depends on the structured representations of our cognitive mechanisms that represent the world as entities and their relations and build the compositional hierarchies to abstract entities to high-level concepts (Battaglia et al., 2018; Hudson and Manning, 2019a; Fodor, 1975; Spelke and Kinzler, 2007). In terms of machine learning, the compositional hierarchies allow feature sharing between entities at multiple levels of representations, can code exponential variability in a very compact way.
and enable fast inference. This makes them potentially suitable for learning and recognizing a higher level of concepts, which drives powerful generalizations.

To be more specific, the structured representations can be divided into two levels: Spatial-level and temporal-level. The spatial-level structure can be understood easily: Human identifies objects and their relations from an image and understands visual scenes by building their compositional hierarchies. Also, we can naturally extend this concept to the temporal-level by considering a video: Successive visual scenes in a video are closely related each other and compose to a semantic event. Likewise, multiple semantic events compose to whole video and human understand the video based on this compositional hierarchy. (See Figure 1.1).
In this dissertation, we aim to study on learning these compositional and hierarchical structured representations. For the study, we discuss the problem from two perspectives: How to learn (methodology) and What to learn (data).

In terms of methodology, a natural form to represent structured information is a graph. A graph describes a collection of entities, represented as nodes, and their pairwise relationships, represented as edges. Recently, a class of neural networks models, Graph Neural Networks (GNNs), has arisen to learn structured representations, which focuses on approaches for reasoning about graph data (Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2013; Kipf and Welling, 2016; Gilmer et al., 2017; Monti et al., 2017; Veličković et al., 2018). The general approach of GNNs is to learn node embeddings with neural networks by passing, transforming, and aggregating node feature information across the graph. In that sense, the structure of GNNs naturally supports combinatorial generalization since it applies computations across the entities and relations in the graph (Battaglia et al., 2018, 2016; Sanchez-Gonzalez et al., 2018). However, it is inherently flat as they propagate information only through edges of the graph so that they are unable to learn structured representations with compositional hierarchies.

In terms of data, video could be a good source as a real-world data from two points of view. First, video is a spatiotemporal data, which human naturally processes in their everyday life. Accordingly, developing methods to learn video can be seen as simulating humans’ intelligence. Second, video data shows a cross-section of everyday life. Therefore, understanding video involves analyzing human vision, language, behavior, and thinking, which is a significant challenge to current machine learning technology.

In this dissertation, we study how to give compositionality to the graph-based neural networks approaches for the structured representations learning.
We consider spatial-level and temporal-level structured representations, and propose two types of compositional graph neural networks to cover both levels. Furthermore, we suggest multilevel video question and answering dataset to integrate the spatial-level and temporal-level learning methods.

1.2 Approaches & Contributions

Firstly, to learn spatial-level compositional structured representations, we consider the problem of learning hierarchical representations of graphs. Even though structural characteristics of graphs can be learned by Graph Neural Networks (GNNs), it is difficult to find an overall compositional hierarchy as the GNNs inherently have a flat operators. We propose a new graph pooling algorithm, Spectrally Similar Graph Pooling (SSGPool), which can be adapted to graph neural network architectures and build compositional hierarchy in an end-to-end fashion. The main idea of the proposed SSGPool algorithm is to learn a coarsening matrix which maps nodes from an original graph to a smaller number of nodes in a coarsened graph. The coarsening matrix is trained to coarsen the nodes based on their feature vectors while keeping the spectral characteristics of the original graph in the coarsened one. Although existing graph pooling methods take either feature-based pooling or structure-preserving pooling, SSGPool considers two properties simultaneously in an end-to-end manner. Experiments on various graph benchmarks show the advantage of our method compared to strong baselines. To further investigate the effectiveness of our proposed method, we evaluate our approach on a real-world problem, image retrieval with visual scene graphs. Quantitative and qualitative analyses on the retrieval problem confirm that the proposed method efficiently captures the hierarchical semantic structure of scene graphs.
Next, to learn temporal-level compositional structured representations, we propose the method to integrate graph-cut mechanism with graph neural networks, Cut-Based Graph Learning Networks (CB-GLNs). Conventional temporal learning methods such as Recurrent Neural Networks (RNNs) focus on interactions between consecutive inputs, i.e. first-order Markovian dependency. However, most of temporal data, as seen with videos, have complex dependency structures that imply variable-length semantic flows and their compositions, and those are hard to be captured by conventional methods. The CB-GLNs learn representations of video by discovering the compositional structure of video in multilevel graph forms. A single video data input is represented as a graph, where nodes and edges represent frames of the video and relationships between all node pairs. From the input, the Cut-Based Graph Learning Networks (CB-GLNs) find temporal structures in the graphs with two key operations: temporally constrained normalized graph-cut and message-passing on the graph. A set of semantic units is found by parameterized kernel and cutting operations, then representations of the inputs are updated by message passing operations. For evaluations, two different tasks for video understanding are conducted: Video theme classification and Video Question and Answering.

Finally, we propose a multilevel video question answering (Video QA) dataset to unify compositional structured representation learning. Several previous studies have suggested datasets for video QA tasks, because the question answering is used as a general benchmark to measure the level of intelligence for video. However, they did not really incorporate levels of understanding, resulting in highly-biased and lack of variance in degree of question difficulty. Therefore, we propose a hierarchical method for building a QA dataset, i.e. hierarchical difficulty levels. The hierarchical difficulty levels are based on the cognitive developmental stages of human intelligence, and defined with two criteria: mem-

5
서울대학교
SEOUl NATIONAL UNIVERSITY
ory capacity and logical complexity. Also, the difficulty levels are aligned with spatial-level and temporal-level based on the two criteria. To evaluate unified learning method, we combine two compositional graph neural networks proposed above and conduct an experiment for multilevel Video QA task.

1.3 Organization of the Dissertation

The remained part of this dissertation is organized as follows.

In Chapter 2, backgrounds for the dissertation are described. First, as a basic module to learn structured representations, Graph Neural Networks are reviewed. Next, backgrounds of structured representation learning for spatial & temporal data are introduced.

In Chapter 3, we propose differentiable graph pooling method, Spectrally Similar Graph Pooling (SSGPool) for spatial-level compositional learning. With SSGPool, structured representations of an image can be effectively learned by capturing compositional hierarchies in the image.

In Chapter 4, we propose Cut-based Graph Learning Networks (CB-GLNs) for temporal-level compositional learning. The CB-GLNs learn compositional hierarchy in the video so that show the improvement of the performance for various video understanding tasks.

In Chapter 5, we propose multilevel Video Question Answering (Video QA) dataset to unify spatial & temporal-level structure learning. To construct multilevel Video QA, Difficulties of questions are proposed based on two criteria, and aligned with spatial & temporal-level. Also, we combine two compositional models proposed in Chapter 3 and 4 to evaluate unified learning framework.

Finally, we summarize the dissertation and discuss contributions and future research directions in Chapter 6.
Chapter 2

Background

In this chapter, we will provide a brief introduction to several background topics that will be extensively used throughout this dissertation. Additional background will be introduced where necessary in later chapters. In what follows, we will give an introduction to Graph Neural Networks (GNNs) in Section 2.1, to structured representations for images in Section 2.2 and to structured representations for video in Section 2.3.

2.1 Structured Representations with Graphs

2.1.1 Graph notations

A graph $G$ is denoted as a pair $(\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, ..., v_N\}$ the set of nodes (vertices), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges. Each node $v_i$ is associated with a feature vector $x_i \in \mathbb{R}^m$. To make notation more compact, the set of node feature vectors of graph $G$ is denoted as a matrix $X = [x_1, x_2, ..., x_N]^\top \in \mathbb{R}^{N \times m}$. Also, a graph has a $N$-by-$N$ weighted adjacency matrix $A$ where $A_{i,j}$ represents the
weight of the edge between \( v_i \) and \( v_j \) and a degree matrix \( D \), a diagonal matrix which contains information about the degree of each node — that is, the sum of edge weights attached to each node. As usual, we denote the combinatorial Laplacian \( L \) of graph \( G \) with \( L = D - A \) and let \( \lambda_k \) and \( \mu_k \) be the \( k \)-th (smallest) eigenvalue and corresponding eigenvector of \( L \) respectively.

### 2.1.2 Graph Neural Networks (GNNs)

Graph neural networks (GNNs) are a class of neural network models suitable for processing graph-structured data, and are of central importance to the topics covered in this dissertation.

Since the first proposed by (Gori et al., 2005), interest in combining deep learning and structured approaches has steadily increased. This has led to various graph-based neural networks being proposed over the years. (Scarselli et al., 2008) extended existing recurrent neural networks for processing the data represented in graph domains. In a graph, each node is naturally defined by its features and the adjacent nodes. The target of GNNs is to learn a state embedding \( h_v \in \mathbb{R}^m \) which contains the information of neighborhood for each node. The state embedding \( h_v \) is an \( m \)-dimension vector of node \( v \) and can be used to produce an output \( o_v \) such as the node label. Let \( f \) be a parametric function, called local transition function, that is shared among all nodes and updates the node state according to the input neighborhood. Also, let \( g \) be the local output function that describes how the output is produced. Then, \( h_v \) and \( o_v \) are defined as follows:

\[
h_v = f(x_v, h_{ne[v]}) \quad (2.1)
\]

\[
o_v = g(h_v) \quad (2.2)
\]
Figure 2.1: An illustration of Graph Convolution Networks. Figure is taken from the original paper of GCNs (Kipf and Welling, 2016).

where $x_v$ is feature of node $v$, $ne[v]$ denotes neighbor nodes of $v$. Based on spectral graph theory (Chung and Graham, 1997), spectral approaches which convert the graph to the spectral domain and apply the convolution kernel of the graph were proposed (Bruna et al., 2013; Henaff et al., 2015; Kipf and Welling, 2016). The spectral convolution operation on graphs is defined as the multiplication of a feature matrix $X \in \mathbb{R}^{N \times m}$ with a filter $g_\theta = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the frequency domain:

$$g_\theta \ast X = U g_\theta U^\top X$$

where $U$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$, and $g_\theta$ can be naturally analyzed as a function of the eigenvalues of $L$, i.e. $g_\theta(\Lambda)$.

To achieve localized filtering with $g_\theta(\Lambda)$, (Hammond et al., 2009) and (Defferrard et al., 2016) suggested applying Chebyshev polynomials to filter $g_\theta(\Lambda)$.

Recently, (Kipf and Welling, 2016) proposed Graph Convolution Networks (GCNs).
(Figure 2.1) by showing a first-order approximation to the Chebyshev polynomials as the graph filter spectrum:

\[ g_\theta * X = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X W \]  
(2.4)

where \( \tilde{A} = A + I_N \), \( \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \) and \( W \) is a matrix of filter parameters. A rich class of convolutional filter functions can be recovered by stacking multiple convolutional layers of the form of Equation 2, with each layer being followed by non-linearity functions.

(Gilmer et al., 2017) suggested the message passing neural networks (MPNNs), which encompass a number of previous neural models for graphs under a differentiable message passing interpretation.

In detail, the MPNNs have two phases, a message passing phase and a update phase. In the message passing phase, the message for vertex \( v \) is defined in terms of message function \( M \):

\[ m_v = \sum_{w \in N(v)} M(h_v, h_w) \]  
(2.5)

where in the sum, \( N(v) \) denotes the neighbors of \( v \) in a graph \( G \). With the message \( m_v \), the representations of vertex \( v \) are updated via the update function \( U \):

\[ h_v = U(h_v, m_v) \]  
(2.6)

The message functions \( M \) and the update functions \( U \) are differentiable so that the MPNNs can be trained in an end-to-end fashion. As extensions, there are some attempts to have been made to improve message passing by putting a gating or attention mechanism into a message function, which can have computational benefits (Monti et al., 2017; Duan et al., 2017; Hoshen, 2017; Veličković et al., 2018; Satorras and Estrach, 2018; van Steenkiste et al., 2018).
We further note other researches for learning a structure of a graph. Neural Relational Inference (NRI) (Kipf et al., 2018) used a Variational Autoencoder (VAE) (Kingma and Welling, 2013) to infer the connectivity between nodes with latent variables. Other generative models based approaches also have been well studied (Bojchevski et al., 2018; De Cao and Kipf, 2018; Simonovsky and Komodakis, 2018).

2.2 Spatial Structured Representations for Images

One of the ultimate goal of computer vision is the complete understanding of images. The recent interest in understanding natural images beyond to perceptual tasks, such as image classification. Most notably, (Krishna et al., 2017b) suggested visual scene graphs as image-specific knowledge graphs to represent symbolic structure within images. The visual scene graph represents the semantic information in image with a set of combined triplets in form of \{head entity(h)-predicate(r)-tail entity(t)\} (Figure 2.2).

One of the important things for visual scene graph is to recognize interactions between entities in images, not just recognizing each entity in isolation. From the aspect of learning visual relationships, Visual Relationship Detection (Lu et al., 2016; Liang et al., 2018; Dai et al., 2017; Zhang et al., 2017), Human-Object Interaction (Chao et al., 2015; Qi et al., 2018; Kato et al., 2018)
and Scene Graph Generation (Xu et al., 2017; Herzig et al., 2018; Zellers et al., 2018; Yang et al., 2018) are studied in previous work. (Zellers et al., 2018) found regularly appearing sub-structures in a scene named motifs, and suggested Stacked Motifs Networks to capture higher-order motifs by encoding and decoding nodes and edges as a global contextualized representation using LSTMs. Recently, (Yang et al., 2018; Woo et al., 2018) tried to infer the graph structure about interactions between entities in an image scene. (Woo et al., 2018) suggested relational embedding module to explicitly model interdependency between entities in an image. Furthermore, (Yang et al., 2018) proposed an edge pruning stage to produce sparse graph structures from a fully-connected graph and employed an attentional graph convolution to iterative update node and edge representations based on its neighbors in the produced graph.

Another important thing is to learn structured representations given visual scene graphs. Since the symbolic structure of an image is well reflected in a visual scene graphs, it is effective to use visual scene graphs in high-level visual tasks such as question answering (Hudson and Manning, 2019b) and image/caption retrieval. (Teney et al., 2017) proposed graph structure based visual question answering model, which use RNNs to model learn representations of graph by sequentially passing the message along with edges. (Hudson and Manning, 2019c) proposed a new task for image question answering, called GQA, by applying symbolic structure of images and question answering with the symbolic structure. (Johnson et al., 2015) used scene graphs as queries for image retrieval by learning scene graph representations via conditional random field. More recently, (Wang et al., 2020) applied scene graph to image-caption retrieval task, learning similarity between image and caption by using distinct GNN models. Most of works for learning representations of graphs can be ex-
plained by GNN framework introduced earlier Section 2.1. However, they do not consider compositional hierarchy to learn representations of spatial structure.

2.3 Temporal Structured Representations for Video

Traditional methods for video representation learning typically use hand-crafted local features, such as dense trajectories, HOG, HOF, etc. (Wang and Schmid, 2013), or mid-level representations on them, such as Fisher Vectors (Sadanand and Corso, 2012). With the resurgence of deep learning methods, there have been numerous attempts to adapt deep models to video representation learning.

One of mainstream deep learning methods for sequential data is Recur- rent Neural Network (RNN) as it naturally take temporal inputs frame by frame. However, as RNN-based methods take frames in (incremental) order, the parameters of methods are trained to capture patterns in transition between successive frames. This makes it hard to find long-term dependencies through overall frames. To consider the long-term dependency, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Chung et al., 2014) introduced switches to the RNN structures and Hierarchical RNN (Chung et al., 2018) stacked multiple layers to find hierarchical structures.

To deal with video data with RNNs, (Ranzato et al., 2014) proposed RNN-based model for feature learning in video explored both spatial and temporal correlations of videos. (Yue-Hei Ng et al., 2015; Donahue et al., 2015) process a continuous video sequence with RNNs, which used 2D CNNs for frame-level feature extraction. A LSTM-based Encoder-Decoder model is also proposed for feature representation learning and the prediction of video frame (Srivastava et al., 2015). Even though those methods ignore noisy (unnecessary) frames
and maintain the semantic flow through the whole sequence, it is still hard for RNN variants to retain multiple semantic flows and to learn their hierarchical and compositional relationships.

To model temporal sequences of adjacent frames and their compositions, Convolutional Neural Network (CNN) based methods, such as ByteNet (Kalchbrenner et al., 2016), ConvS2S (Gehring et al., 2017) and WaveNet (Oord et al., 2016), applied 1D convolution operator, are suggested. More recently, 3D CNNs have been proposed (Tran et al., 2015; Carreira and Zisserman, 2017; Wang et al., 2018; Feichtenhofer et al., 2019). By considering a video as a stack of frames, it is natural to develop 3D convolutions applied directly on video sequence. However, 3D CNNs often introduce a large number of model parameters, which inevitably require a large amount of training data to achieve good performance. Also, two-stream architectures which is fed the video data, including RGB frames and optical flow subsequences, has been proposed (Feichtenhofer et al., 2019, 2017a,b; Simonyan and Zisserman, 2014a). However these methods hardly captured variable-length dependencies which play a significant role as semantic units.

Recent researches revisited the traditional idea of clustering successive frames into representative clusters. Deep Bag of Frame (DBoF) (Abu-El-Haija et al., 2016) randomly selects $k$ frames from whole sequences as the representatives. NetVLAD (Arandjelovic et al., 2016) divides all features into $k$ clusters and calculates residuals, which are difference vectors between the feature vectors and their corresponding cluster center, as the new representations for each feature vector. Even though the main idea of this type of research is quite simple, it helps to understand the semantics of long sequences by focusing on a small number of representative frames. However, it is hard to consider the complex temporal relationships.
Along with enormous interest of attention mechanism, a number of significant researches (Vaswani et al., 2017; Devlin et al., 2019) have been aimed to understanding the long (temporal) inputs relying purely on self-attention mechanisms. With real-world applications on natural language understanding, such as question and answering (QA) and paragraph detection, those methods focus on meaningful words (or phrases, sentences) among the long reading passages and ignore irrelevant words to the passage by stacking multiple attention layers. To apply self-attention mechanism to video representation learning, various approaches has been proposed (Zhou et al., 2018; Girdhar et al., 2019; Sun et al., 2019; Wu et al., 2019; Ma et al., 2018). However, these methods consist of a large number of layers with a huge number of parameters and require a huge amount of training dataset.
Chapter 3

Compositional Learning for Spatial Structure: Spectrally Similar Graph Pooling

3.1 Introduction

By virtue of the recent progress on graph neural networks (GNNs) (Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2013; Kipf and Welling, 2016; Gilmer et al., 2017; Velićković et al., 2018), various types of data including structural data can be dealt with using neural network algorithms. While conventional neural network algorithms, such as convolutional neural networks and recurrent neural networks, take regular structured inputs (images with grid pixel structure and sound signals with Markovian temporal dependencies), GNNs have been recently suggested as a method for extending the scope of the inputs to graphs having irregular structures, such as molecular data, knowledge graphs, social networks and visual scene graphs. Most GNNs attempt to implicitly reflect the structural information through node (graph) representations. In
other words, GNNs assign feature vectors to each node and update the node features by transforming and aggregating information from the neighborhoods. Even though structural characteristics can be learned by applying these message passing steps repeatedly, it is difficult to find an overall compositional hierarchy using such flat operators.

Recent work has proposed using pooling methods such as CNNs in order to discover hierarchical structures between nodes in GNNs (Vinyals et al., 2015; Ying et al., 2018; Zhang et al., 2018; Lee et al., 2019; Gao and Ji, 2019; Diehl, 2019; Ma et al., 2019). These studies are divided into two categories depending on what information is mainly used for the pooling operator: structure-based approaches and feature-based approaches. Structure-based approaches learn node features with GNNs, however, the original graph is coarsened by deterministic graph clustering algorithms based on graph theory. Therefore, the resultant coarsened graph reflects the topology of the original graph, but the node features are not used during coarsening. Also the deterministic clustering methods are not end-to-end trainable. On the other hand, feature-based approaches assign similar feature vectors in the original graph to the same node in the coarsened graph. Even though these approaches can be trained in an end-to-end manner, it is hard to maintain the topology information of the original graph.

In this chapter, we propose a new graph pooling method, Spectrally Similar Graph Pooling (SSGPool), which makes use of both node features and structural information between the nodes (Figure 3.1). The main idea of SSGPool is to learn a coarsening matrix which maps nodes from an original graph to a smaller number of nodes in a coarsened graph. The coarsening matrix is trained to coarsen the nodes based on correlations between their feature vectors while maintaining the spectral characteristics of the original graph. To utilize the node feature vectors, SSGPool basically builds upon conventional GNN algorithms.
In addition, structural similarities between two different sized graphs are defined in order to be used as a regularizer during training.

Experiments on various graph benchmarks show the advantage of our method compared to strong baselines. To further investigate the effectiveness of our proposed method, we evaluate our approach on a real-world problem, image retrieval with visual scene graphs. Quantitative and qualitative analyses on the retrieval problem confirm that the proposed method efficiently captures the hierarchical semantic structures of scene graphs.

The remainder of the paper is organized as follows. In Section 2, we review related work about the graph pooling algorithms. Next, we introduce notations about the graphs, GNN algorithms and spectral similarity between graphs as preliminaries. After that, the proposed SSGPool method is explained in detail and experimental results on various datasets, comparing our proposed algorithm with other well-known graph pooling algorithms are presented.
3.2 Related Work

Pooling operations in graph neural networks (GNNs) can scale down the size of inputs and enlarge the receptive fields, thus giving rise to better generalization and performance. In this section, we review several recent methods for graph pooling coupled with GNNs. Graph pooling methods can be grouped into the following two categories: structure-based pooling and feature-based pooling.

**Structure-based Pooling.** Including earlier works of neural networks on graph, several proposed GNNs perform pooling with existing graph clustering algorithm. These methods learn the representations of graphs in 2-steps: First these pooling methods build hierarchical structures using a graph clustering algorithm. Next, they learn embeddings of nodes in each layer based on GNN modules. (Bruna et al., 2013) built a hierarchy of the graph with agglomerative clustering. (Defferrard et al., 2016) and (Fey et al., 2018) used the Graclus algorithm (Dhillon et al., 2007) which computes graph clustering without eigenvectors. (Simonovsky and Komodakis, 2017) constructed the graph hierarchies through a combined use of spectral polarity and Kron reduction. More recently, (Ma et al., 2019) proposed EigenPool, which used spectral graph clustering methods to produce a coarsened graph. These methods leverage topological information from graphs in order to produce coarsened graph. However these methods do not use node features which have useful information for learning representations of graphs. Furthermore, as the existing graph clustering algorithms are not differentiable, they are incapable of learning in an end-to-end fashion.

**Feature-based Pooling.** In contrast to structure-based pooling, several end-to-end trainable pooling methods are proposed. (Ying et al., 2018) proposed a differentiable graph pooling module (DiffPool) to softly assign nodes
to a set of clusters using neural networks, forming fully connected coarsened graphs through a dense cluster assignment matrix. (Gao and Ji, 2019) and (Lee et al., 2019) devised a top-K node selection-based pooling method (gPool and SAGPool) to form an induced subgraph for the next layer. Although it is efficient, this method loses the completeness of the graph structure information. In addition, (Vinyals et al., 2015) proposed Set2Set, the global pooling operation by aggregating information through RNNs. (Zhang et al., 2018) proposed SortPool which pools graphs according to the feature map values that are sorted in descending order. (Diehl, 2019) designed a pooling operation by contracting the edges (EdgePool). The contracting scores are calculated by features from the two incident nodes. These approaches learn hierarchical structures from node features with differentiable parameters. However, they tend not to reflect the topology information of the graph for pooling.

3.3 Preliminaries

3.3.1 Basic Module of GNNs

Due to an ever increasing interest in combining deep learning and structured approaches, various graph-based neural networks have been proposed over the years. Based on spectral graph theory (Chung and Graham, 1997), approaches which convert graphs to the spectral domain and apply convolution kernels of the graphs have been proposed (Bruna et al., 2013; Henaff et al., 2015; Kipf and Welling, 2016). (Gilmer et al., 2017) suggested the message passing framework, which encompasses a number of previous neural models for graphs under a differentiable message passing interpretation. (Xu et al., 2018) analyzed the representation power of various GNN architectures and proposed Graph Isomorphism Networks (GIN), where representational power is equal to the
power of the Weisfeiler-Lehman test.

In this chapter, we use a simple form of a message passing function similar to GIN.

\[ M(A, X) = (X + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X)W_m \]  

(3.1)

where \( W_m \in \mathbb{R}^{f \times f} \). After that, we define a single GNNs layer as follows:

\[
\text{GNN}(A, X) = \left[ \sigma (M^2 (A, \sigma (M^1 (A, X)))) ; \sigma (M^1 (A, X)) \right] W_g 
\]  

(3.2)

where \( M_1 \) and \( M_2 \) are message passing layer, \( \sigma \) is an activation function, \( [X; Y] \) denotes row-wise concatenation of two matrix and \( W_g \in \mathbb{R}^{2f \times f'} \) is a learnable parameter for \( \text{GNN}(A, X) \). In the rest of the paper, we use \( \text{GNN}(A, X) \) in Equation (3.2) as a base GNNs module.

### 3.3.2 Spectral Similarity between Graphs

Spectral graph theory has been considered as a powerful way to describe structural characteristics of graphs. Therefore, structural similarity between two graphs can be clearly defined by comparing the spectral properties of graphs.

For two graphs having the same number of nodes, (Spielman and Srivastava, 2011; Spielman and Teng, 2011) proposed spectral similarity to determine how closely a graph \( G_s \) approximates a \( G \):

\[
\forall f \in \mathbb{R}^N, \quad (1 - \epsilon)f^T L f \leq f^T L_s f \leq (1 + \epsilon)f^T L f
\]  

(3.3)

where \( L \) is a Laplacian matrix of a graph \( G \). If the equation holds, we can say that \( G_s \) is an \( \epsilon \)-spectral approximation of \( G \).

For the graph coarsening problem which has different number of nodes between the original graphs and coarsened graphs, (Loukas and Vandergheynst, 2018) generalized it by restricting to first \( K \)-eigenspace: the restricted spectral
similarity (RSS). If there is a mapping matrix $P \in \mathbb{R}^{N \times n}$ between original vertex set $V = \{v_1, ..., v_N\}$ and the coarsened vertex set $V_c = \{v'_1, ..., v'_n\}$, then RSS is defined as follows:

**Restricted Spectral Similarity (RSS).** Suppose that there exist an integer $K$ and positive constant $\epsilon_k$, such that for every $k \leq K$,

$$(1 - \epsilon_k)u_k^\top Lu_k \leq u_k^\top \tilde{L}u_k \leq (1 + \epsilon_k)u_k^\top Lu_k, \quad \tilde{L} = P^\top L_c P^+, \quad L_c = P^\top L P$$

(3.4)

where $u_k$ is $k$-th eigenvector of $L$, $P^+$ and $P^\top$ are pseudo-inverse of $P$ and its transpose, and $\tilde{L}$ is Laplacian matrix of lifted (reverse of coarsening) graph of $G_c$ from $\mathbb{R}^n$ back to $\mathbb{R}^N$. Then, the $G_C$ is said to satisfy the restricted spectral similarity property with the RSS constants $\{\epsilon_k\}_{k=1}^K$.

### 3.4 Spectrally Similar Graph Pooling

We suggest a new graph pooling algorithm which learns coarsening matrix to construct adjacency matrix and node feature matrix of upper layers while keeping spectral characteristics of original graphs. The main idea is to keep the spectral information by maximizing the similarity between the Fiedler vector of original graphs and its coarsened ones. As two vectors are on different dimensional spaces, the vector of the coarsened graph is lifted back to the original space using the inverse of the coarsening matrix. In order to make the whole process end-to-end trainable, we define the coarsening matrix and derive the easy inversion of the coarsening matrix. Then, we explain how SSGPool learns hierarchical representations of graphs. Lastly, we introduce a novel regularization objective which maintains the global structure of the graph during the pooling procedure. Figure 3.2 shows the overall architecture of proposed method.
Figure 3.2: The architecture of SSGPool layer combined with graph neural networks. The SSGPool learns coarsening matrices $P$ to minimize task-specific loss while retaining spectral similarity. To represent the spectral similarity, we use second smallest eigenvector (Fiedler vector) of graph Laplacian.
3.4.1 Graph Coarsening

The coarsening can be expressed with a surjective map (i.e., many-to-one) \( \varphi : \mathcal{V}_N \to \mathcal{V}_n \) between the original vertex set \( \mathcal{V}_N \) and the smaller vertex set \( \mathcal{V}_n \).

Then, graph coarsening can be defined via a coarsening matrix:

**Definition 1** (Coarsening matrix). Matrix \( P \in \{0, 1\}^{N \times n} \) is a coarsening matrix with regard to graph \( G \) if and only if it satisfies the condition that it is a surjective mapping of the vertex set, meaning that if \( P(i, r) = 1 \) then \( P(i, r') = 0 \) for every \( r' \neq r \).

Similar to (Loukas, 2019), the expensive pseudo-inverse computation for \( P \) can be substituted by simple transposition and re-scaling:

**Proposition 1** (Easy inversion). The pseudo-inverse of a coarsening matrix \( P \) is given by \( P^+ = Q^{-2}P^\top \), where \( Q \in \mathbb{R}^{n \times n} \) is a diagonal matrix with \( Q(r, r) = \|P(:, r)\|_2^2 \).

**Proof.** Suppose we have a coarsening matrix \( P \in \{0, 1\}^{N \times n} \). Similar to (Loukas, 2019), to show that matrix \( \Pi = PQ^{-2}P^\top \) is a projection matrix, i.e., \( \Pi^2 = \Pi \), of rank \( n \) is a sufficient condition to prove \( P^+ = Q^{-2}P^\top \), where \( P \) has rank \( n \). Suppose we particularly sort rows of \( P \), such that for any two columns \( r < r' \), if \( P(i, r) = 1 \) then \( P(i', r') = 0 \) for \( i' < i \). Furthermore, denoted by \( p_r \) the \( 1 \) vector containing all 1 entries of \( P(:, r) \) such that \( \|p_r\|_2 = \|P(:, r)\|_2 = Q(r, r) \). Then the \( B_r = p_rD(r, r)^{-2}p_r^\top \) is a rank 1 projection matrix as \( B_r^2 = B_rB_r = (p_rD(r, r)^{-2}p_r^\top)(p_rD(r, r)^{-2}p_r^\top) = p_rD(r, r)^{-2}p_r^\top p_r \|p_r\|_2^2 = B_r \).

Finally, the matrix \( \Pi \) is a block-diagonal matrix (in our particular sorting), \( \Pi = diag(B_1, ..., B_n) \), so that the \( \Pi^2 = \Pi \) is a projection matrix of rank \( n \).
### 3.4.2 Pooling with Coarsening Matrix

Suppose we have the learned coarsening matrix at $l$-th layer, $P_l \in \mathbb{R}^{N_l \times N_{l+1}}$. With $P_l$, SSGPool layer coarsens the graph, generating a new coarsened adjacency matrix $A_{l+1}$ and a new node feature matrix $X_{l+1}$.

Most previous coarsening based pooling approaches such as (Ying et al., 2018; Ma et al., 2019) used a quadratic form of the adjacency matrix to obtain new coarsened adjacency matrix, $A_{l+1} = P_l^T A_l P_l$. Instead, we use the Laplacian matrix $L_l$ to obtain a new coarsened adjacency matrix $A_{l+1}$:

$$L_{l+1} = P_l^T L_l P_l$$
$$A_{l+1} = D_{l+1} - L_{l+1} \quad (3.5)$$

where $D_{l+1}$ is a degree matrix obtained by leaving only diagonal terms of $L_{l+1}$.

Utilizing $L_l$ instead of $A_l$ has two noteworthy benefits. First, the obtained coarsened adjacency matrix is not diagonal-dominant: the coarsened graph obtained from the quadratic form of $A$ has significantly stronger self-loops than any other connections, and these self-loops might hamper the message passing of GNNs. Second, our coarsening is consistent with regard to the Laplacian form: the Laplacian matrix of the coarsened graph retains spectral properties as is desired, e.g., the nullspace of $L$ is preserved both by coarsening and lifting because $P_l 1_{N_{l+1}} = 1_{N_{l+1}}$ and $P_l^+ 1_{N_l} = 1_{N_l}$.

Further, the new node feature matrix of the next layer $X_{l+1}$ is obtained as follows:

$$Z_l = \text{GNN}_{l, \text{embed}}(A_l, X_l)$$
$$X_{l+1} = P_{l, \text{soft}}^+ Z_l \quad (3.6)$$

where $P_{\text{soft}}$ is softmax outputs of $P$, which will be covered in the next section.
3.4.3 Learning the Coarsening Matrix

We describe how SSGPool generates the coarsening matrix at the $l$-th layer, $P_l \in \mathbb{R}^{N_l \times N_{l+1}}$. For convenience, we drop the notation of layer $l$ and denote $p_i = P(i,:)$. According to Definition 1, $p_i$ can be defined as a categorical random variable with probabilities $\pi_{i1}, \pi_{i2}, ..., \pi_{in}$, where $n$ is the number of nodes in the coarsened graph.

It is straightforward to sample from $p_i$, but we cannot backpropagate gradients though the sampling since the variables are discrete. A recently popular approach to handle this difficulty is to sample from a continuous approximation of the discrete distribution (Maddison et al., 2016; Jang et al., 2017a), and use the reparameterization trick to get (biased) gradients from this approximation. We employ a Straight-Through Gumbel-Softmax estimator (i.e., ST-Gumbel) (Jang et al., 2017a) to ensure end-to-end training. The probability $\pi$ is estimated via the GNN module followed by softmax function:

$$\Pi = P_{\text{soft}} = \text{softmax}(\text{GNN}_{\text{pool}}(A, X))$$

(3.7)

Finally, the $p_i$ can be drawn as follows:

$$p_i = \text{one_hot} \left( \arg \max_j [\pi_{ij}] \right)$$

(3.8)

Although the original ST-Gumbel trick utilizes samples drawn from $g \sim \text{Gumbel}(0, 1)$ to give stochasticity, we drop this sampling procedure and choose $j$ only with the probability $\pi$.

3.4.4 Spectral Similarity of Graphs as Regularization

In this section, we propose the spectral regularizer for a graph pooling, which enforces coarsening matrices to keep coarsened graph spectrally similar to the original graph.
In spectral graph theory, the second smallest eigenvector of graph Laplacian, also known as Fiedler vector, entails the global structure of graphs, as it is the function that maps adjacent nodes with similar values: The larger difference between values of nodes has the farther topological distance between nodes is.

The Fiedler vector $u_f^{(l-1)}$ of the graph in $(l-1)$-th layer can be coarsened and lifted given a coarsening matrix $P_{l-1}$:

$$
\tilde{u}_f^{(l)} = P_{l-1} u_f^{(l-1)}, \quad \tilde{u}_f^{(l-1)} = P_{l-1} \tilde{u}_f^{(l)}
$$

(3.9)

where $\tilde{u}_f$ is a vector that has been sequentially coarsened and lifted from eigenvector $u$ given the matrix $P$. Then, the $\tilde{u}_f$ is the best approximation of $u$ given $P$, because the $PP^+$ is the projection matrix with a smaller rank (See the proof of Proposition 1). Therefore, as the distance between $\tilde{u}_f$ and $u_f$ gets closer, the original graph and coarsened graph become more similar to each other in terms of global structure.

$$
\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Task}} + \lambda \cdot \frac{\sum_{k=1}^{L} \left(1 - \frac{u_k^\top \tilde{u}_f}{|u_f| |\tilde{u}_f|} \right)}{L-1}
$$

(3.10)

where $\mathcal{L}_{\text{Total}}$ is task-specific loss term, $L$ is the number of layers and $\lambda$ is a hyperparameter for the regularization term.

**Connection to Restricted Spectral Similarity (RSS).** Followed by (Loukas, 2019), The RSS can be re-formulated through the following induced semi-norm:

$$
||u||_L = \sqrt{u^\top Lu}, \quad ||u_c||_{L_c} = \sqrt{u_c^\top L_c u_c}, \quad \text{where} \quad u_c = P^+ u
$$

$$
(1 - \epsilon)||u_c||_L \leq ||u_c||_{L_c} \leq (1 + \epsilon)||u||_L
$$

(3.11)

Then, we can obtain an upper bound of difference between semi-norms of the original graph and the coarsened graph with a triangular inequality.

$$
\frac{||u - \tilde{u}||_L}{||u||_L} \geq \frac{||u||_L - ||u_c||_{L_c}}{||u||_L}, \quad \text{where} \quad \tilde{u} = Pu_c
$$

(3.12)
**proof.** refered to (Loukas, 2019), by the triangle inequality:

\[
||u||_L - ||u_c||_{L_c} = \left| \sqrt{u^\top Lu} - \sqrt{u_c^\top L_c u_c} \right| = \left| \sqrt{u^\top Lu} - \sqrt{u^\top P^+ P^\top L P P^+ u} \right|
\]

\[
= \left| \sqrt{u^\top Lu} - \sqrt{\tilde{u}^\top L \tilde{u}} \right| \leq \sqrt{(u - \tilde{u})^\top L (u - \tilde{u})} = ||(u - \tilde{u})||_L
\]

(3.13)

Therefore, reducing the distance between \(u\) and \(\tilde{u}\) with our regularization term makes the original graph and coarsened graph spectrally similar.

**Simpler Spectral Loss for Efficient Computation.** As the size of the input graph and the number of layers increase, the burden to compute eigen-decompositions for graphs in all layers is also increase. Therefore, we also propose simpler version of spectral loss to avoid this heavy computations. To start with, the relationship between the original graph and the final coarsened graph is expressed in compact form:

\[
L_f = P_* L_0 P_*^\top, \quad \tilde{L}_0 = P_*^+ L_f P_*^\top
\]

(3.14)

where \(L_f\) and \(L_0\) are Laplacian matrices of the final coarsened graph and the original graph, \(P_* = P_f \cdots P_0\) and \(P_*^+ = P_0^+ \cdots P_f^+\).

Then, the Fiedler vector \(u_f^{(0)}\) of the original graph can be coarsened and lifted given a coarsening matrix \(P_*\):

\[
u_c = P_*^+ u_f^{(0)}, \quad \tilde{u}_f^{(0)} = P_* u_c
\]

(3.15)

Finally, the simpler spectral loss can be obtained as follows:

\[
\mathcal{L}_{\text{smpl}} \text{specloss} = \lambda \cdot \left( 1 - \frac{u_f^{(0)} \tilde{u}_f^{(0)}}{|u_f^{(0)}| \cdot |	ilde{u}_f^{(0)}|} \right)
\]

(3.16)

By doing so, we only need to compute eigen-decomposition for original graphs, and it can be done in pre-processing steps.
3.5 Experiments

In this section, we show the advantages of SSGPool compared to other competitive graph pooling algorithms with various graph benchmark datasets. Also, we apply our method to image retrieval with visual scene graphs and highlight that our method effectively works in real-world problem.

3.5.1 Experimental Details

Baselines

For the experiments, we use five competitive baselines recently proposed for differentiable graph pooling: SortPool (Zhang et al., 2018), gPool (Gao and Ji, 2019), SAGPool (Lee et al., 2019), EdgePool (Diehl, 2019) and DiffPool (Ying et al., 2018). The details of each baselines are shown below:

SortPool (Zhang et al., 2018): SortPool is a spatial pooling method that sorts nodes in spatial order according to their structural roles. After sorting, they rearrange truncated k nodes from the graph and feed it to a following 1D convolutional layer.

gPool (Gao and Ji, 2019): gPool is a top-k pooling operation that samples a set of important nodes from a graph. gPool samples k nodes according to the score based on a trainable projection vector. In this experiment, the parameter k was selected from the pooling ratios of other models.

SAGPool (Lee et al., 2019): A variant of gPool, SAGPool considers both node feature and graph topology for sampling nodes. SAGPool computes self attention scores between nodes after passing through graph convolution layers to select top-k important nodes.

EdgePool (Diehl et al., 2019): EdgePool is a pooling method that utilizes edge contraction. Edge contraction is to choose an edge based on a certain edge
scoring function and combine two nodes that are linked with it. Unlike other models, the pooling ratio is fixed.

**DiffPool (Ying et al., 2018):** DiffPool is an end-to-end differentiable pooling method that learns hierarchical representations of graphs. DiffPool makes use of a trainable soft assignment matrix that regards both node feature and graph topology.

### Implementation Details

For the graph classification task, we evaluated all the comparative models over 10 random seeds using 10-fold cross validation. The total of 100 testing results was used to obtain the final accuracy of each method on each dataset. 10 percent of the training data was used for validation in the training session. In order to ensure a fair comparison, both the proposed and comparative pooling methods are implemented through the same neural network architecture with same GNN module. We use Adam optimizer with $1e^{-3}$ learning rate, and the models are trained for 100 epochs with batch size 16. To get optimal hyper-parameters for each model, the number of pooling layer $K \in \{1, 2\}$, pooling ratio $r \in \{0.1, 0.2, 0.4, 0.8\}$ and 32 dimensional feature vectors nodes are used. A 1-pooling layer architecture is composed of \{Graph Convolution - Graph Pooling - Graph Convolution\} and 2-pooling layer architecture is composed of \{Graph Convolution - Graph Pooling - Graph Convolution - Graph Pooling - Graph Convolution\} with GNN modules.

For the image retrieval task, we compared our model with two representative pooling algorithms, DiffPool and SAGPool. To get optimal hyper-parameters for each model, A single pooling layer is used with pooling ratio $r \in \{0.1, 0.2, 0.4, 0.8\}$ and 300 dimensional feature vectors nodes are used. We use Adam optimizer with $1e^{-4}$ learning rate, and the models are trained for 30 epochs with batch
### Table 3.1: Statistics of graph benchmark datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th># of graphs</th>
<th># of classes</th>
<th># of avg. nodes</th>
<th># of avg. edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUTAG</td>
<td>188</td>
<td>2</td>
<td>17.93</td>
<td>19.79</td>
</tr>
<tr>
<td>ENZYMES</td>
<td>600</td>
<td>6</td>
<td>32.63</td>
<td>62.14</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>1,113</td>
<td>2</td>
<td>39.06</td>
<td>72.82</td>
</tr>
<tr>
<td>NCI1</td>
<td>4,110</td>
<td>2</td>
<td>29.87</td>
<td>32.30</td>
</tr>
</tbody>
</table>

size 32.

#### 3.5.2 Graph Classification Task with Graph Benchmarks

We evaluate SSGPool on a variety of graph datasets from benchmarks commonly used for graph classification tasks. To examine the general ability of our model, four datasets are selected according to their statistics such as amount of data and graph size: MUTAG (Debnath et al., 1991), ENZYME (Borgwardt et al., 2005), PROTEINS (Feragen et al., 2013) and NCI1 (Shervashidze et al., 2011). The details of each dataset are explained in Table 3.1.

Table 3.2 shows overall results for graph benchmarks compared to other state-of-the-art graph pooling methods. The average and standard deviation are obtained from 10 times of 10-fold cross validations test. The best scores of the proposed method are selected from the versions of SSGPool without regularization, SSGPool with regularization and SSGPool with simpler regularization.

We observed that, for ENZYME and NCI datasets, the SSGPool showed best performance. Even though the SSGPool achieves second best performance in MUTAG and PROTEINS datasets, it shows very competitive results compared to other methods. Also, it is worthwhile to note that the selected four datasets have distinct statistics in terms of the number of data and graph size.
Table 3.2: Average accuracy and standard deviation for graph benchmarks are presented. The DiffPool* denotes DiffPool with additional losses originally proposed in (Ying et al., 2018). We highlight the best results (bold) and second best results (blue).

<table>
<thead>
<tr>
<th>Model</th>
<th>MUTAG</th>
<th>ENZYME</th>
<th>PROTEINS</th>
<th>NCI1</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNNs</td>
<td>0.746±0.007</td>
<td>0.301±0.023</td>
<td>0.726±0.007</td>
<td>0.733±0.005</td>
</tr>
<tr>
<td>SortPool</td>
<td>0.832±0.016</td>
<td>0.277±0.020</td>
<td>0.730±0.012</td>
<td>0.734±0.011</td>
</tr>
<tr>
<td>gPool</td>
<td>0.732±0.018</td>
<td>0.303±0.019</td>
<td>0.734±0.006</td>
<td>0.721±0.004</td>
</tr>
<tr>
<td>SAGPool</td>
<td>0.803±0.015</td>
<td>0.326±0.028</td>
<td>0.730±0.006</td>
<td>0.738±0.009</td>
</tr>
<tr>
<td>EdgePool</td>
<td>0.770±0.033</td>
<td>0.329±0.025</td>
<td>0.731±0.004</td>
<td>0.751±0.006</td>
</tr>
<tr>
<td>DiffPool*</td>
<td><strong>0.853±0.019</strong></td>
<td>0.283±0.043</td>
<td><strong>0.756±0.009</strong></td>
<td>0.743±0.009</td>
</tr>
<tr>
<td>SSGPool</td>
<td><strong>0.852±0.009</strong></td>
<td><strong>0.382±0.012</strong></td>
<td><strong>0.750±0.005</strong></td>
<td><strong>0.753±0.010</strong></td>
</tr>
</tbody>
</table>

As a result, all comparative models show considerably different performance depending on the datasets. For example, The DiffPool shows best performance at MUTAG and PROTEINS but for the ENZYME and NCI1, it achieves degraded scores. However, the proposed method consistently showed good performance across all datasets.

To see the effect of proposed regularizer, we reported the results of SSGPool with and without regularization loss. Table 3.3 shows the comparison results along with pooling ratio and the number of pooling layers for each dataset. We observed that the proposed regularization term usually improves performance most of configurations across the all datasets. This implies that preserving global structures while simultaneously pooling graphs has a substantial impact on graph representation learning.
Table 3.3: Effect of proposed regularizer in SSGPool for various benchmark datasets along with pooling ratio and the number of pooling layer.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pooling 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.844±0.019</td>
<td>0.834±0.015</td>
<td>0.839±0.019</td>
<td>0.846±0.015</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.847±0.014</strong></td>
<td><strong>0.845±0.013</strong></td>
<td><strong>0.844±0.014</strong></td>
<td>0.844±0.012</td>
</tr>
<tr>
<td><strong>Pooling 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.841±0.017</td>
<td>0.845±0.025</td>
<td>0.841±0.027</td>
<td>0.825±0.032</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.844±0.011</strong></td>
<td><strong>0.852±0.009</strong></td>
<td><strong>0.846±0.008</strong></td>
<td><strong>0.850±0.010</strong></td>
</tr>
</tbody>
</table>

(a) Classification results for MUTAG dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pooling 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.362±0.037</td>
<td>0.364±0.002</td>
<td>0.348±0.040</td>
<td>0.369±0.021</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.382±0.012</strong></td>
<td><strong>0.378±0.022</strong></td>
<td><strong>0.377±0.034</strong></td>
<td><strong>0.375±0.028</strong></td>
</tr>
<tr>
<td><strong>Pooling 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.348±0.023</td>
<td>0.352±0.020</td>
<td>0.302±0.052</td>
<td>0.317±0.033</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.363±0.025</strong></td>
<td><strong>0.374±0.029</strong></td>
<td><strong>0.343±0.016</strong></td>
<td>0.303±0.025</td>
</tr>
</tbody>
</table>

(b) Classification results for ENZYMES dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pooling 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.735±0.010</td>
<td><strong>0.741±0.008</strong></td>
<td>0.728±0.014</td>
<td>0.723±0.012</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.750±0.010</strong></td>
<td>0.736±0.013</td>
<td><strong>0.741±0.009</strong></td>
<td><strong>0.735±0.014</strong></td>
</tr>
<tr>
<td><strong>Pooling 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.733±0.011</td>
<td><strong>0.745±0.006</strong></td>
<td>0.742±0.007</td>
<td>0.742±0.006</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.743±0.009</strong></td>
<td>0.744±0.003</td>
<td><strong>0.750±0.007</strong></td>
<td><strong>0.750±0.005</strong></td>
</tr>
</tbody>
</table>

(c) Classification results for PROTEINS dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pooling 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td><strong>0.738±0.019</strong></td>
<td><strong>0.738±0.018</strong></td>
<td><strong>0.734±0.015</strong></td>
<td><strong>0.729±0.018</strong></td>
</tr>
<tr>
<td>w/ Reg.</td>
<td>0.735±0.022</td>
<td>0.735±0.012</td>
<td>0.731±0.014</td>
<td>0.728±0.017</td>
</tr>
<tr>
<td><strong>Pooling 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o Reg.</td>
<td>0.742±0.014</td>
<td>0.703±0.041</td>
<td>0.737±0.015</td>
<td>0.752±0.005</td>
</tr>
<tr>
<td>w/ Reg.</td>
<td><strong>0.746±0.010</strong></td>
<td><strong>0.743±0.007</strong></td>
<td><strong>0.753±0.010</strong></td>
<td><strong>0.753±0.012</strong></td>
</tr>
</tbody>
</table>

(d) Classification results for NCI1 dataset.
Table 3.4: Comparison of regularization in condition of 2 pooling layers along with pooling ratio. **Original Reg.** is the SSGPool with original regularizer and **Simpler Reg.** is the one with simpler regularizer.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Reg.</strong></td>
<td>0.840±0.014</td>
<td>0.837±0.015</td>
<td>0.837±0.027</td>
<td>0.831±0.025</td>
</tr>
<tr>
<td><strong>Simpler Reg.</strong></td>
<td><strong>0.842±0.019</strong></td>
<td><strong>0.852±0.009</strong></td>
<td><strong>0.846±0.008</strong></td>
<td><strong>0.850±0.010</strong></td>
</tr>
</tbody>
</table>

(a) Classification results for MUTAG dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Reg.</strong></td>
<td>0.363±0.025</td>
<td>0.374±0.029</td>
<td>0.343±0.016</td>
<td>0.303±0.025</td>
</tr>
<tr>
<td><strong>Simpler Reg.</strong></td>
<td>0.290±0.022</td>
<td>0.264±0.028</td>
<td>0.307±0.014</td>
<td><strong>0.338±0.020</strong></td>
</tr>
</tbody>
</table>

(b) Classification results for ENZYMES dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Reg.</strong></td>
<td>0.743±0.009</td>
<td><strong>0.744±0.003</strong></td>
<td><strong>0.750±0.007</strong></td>
<td><strong>0.750±0.005</strong></td>
</tr>
<tr>
<td><strong>Simpler Reg.</strong></td>
<td><strong>0.747±0.008</strong></td>
<td><strong>0.744±0.007</strong></td>
<td>0.745±0.005</td>
<td><strong>0.750±0.003</strong></td>
</tr>
</tbody>
</table>

(c) Classification results for PROTEINS dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Reg.</strong></td>
<td>0.729±0.003</td>
<td>0.742±0.014</td>
<td>0.752±0.007</td>
<td>0.747±0.017</td>
</tr>
<tr>
<td><strong>Simpler Reg.</strong></td>
<td><strong>0.746±0.010</strong></td>
<td><strong>0.743±0.007</strong></td>
<td><strong>0.753±0.010</strong></td>
<td><strong>0.753±0.012</strong></td>
</tr>
</tbody>
</table>

(d) Classification results for NCI1 dataset.

Furthermore, as we suggest to compute the spectral regularizer for every layers, we compared the performance to a simpler version which computes spectral loss with original graph and final coarsened graph\(^1\). Table 3.4 shows the comparison results of original version of regularizer and its simpler version along with pooling ratio when 2 pooling layers are used. For the ENZYME dataset, the original regularizer performs considerably better than simpler one. However, for the MUTAG, PROTEINS and NCI1, it was found that the simpler version achieves comparative scores, or even higher ones. As the all coarsening matrices are engaged in reducing the regularization loss in simpler version, we can conclude that the simpler version of regularization is powerful enough to learn

\(^1\)With this condition, we need to compute eigen-decomposition only for Laplacian matrix of original graph and it can be done in preprocessing steps.
Table 3.5: Comparison of computation time (millisecond) for single graph of regularization in condition of 2 pooling layers along with pooling ratio. Original Reg. is the SSGPool with original regularizer and Simpler Reg. is the one with simpler regularizer.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Reg.</td>
<td>21.9</td>
<td>22.1</td>
<td>22.4</td>
<td>22.5</td>
</tr>
<tr>
<td>Simpler Reg.</td>
<td>20.7</td>
<td>21.1</td>
<td>21.1</td>
<td>21.2</td>
</tr>
</tbody>
</table>

(a) computation time (millisecond) for MUTAG dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Reg.</td>
<td>22.0</td>
<td>22.4</td>
<td>24.6</td>
<td>28.1</td>
</tr>
<tr>
<td>Simpler Reg.</td>
<td>21.3</td>
<td>21.5</td>
<td>21.8</td>
<td>22.2</td>
</tr>
</tbody>
</table>

(b) computation time (millisecond) for ENZYMES dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Reg.</td>
<td>27.1</td>
<td>28.0</td>
<td>30.2</td>
<td>37.5</td>
</tr>
<tr>
<td>Simpler Reg.</td>
<td>24.5</td>
<td>24.9</td>
<td>26.0</td>
<td>29.8</td>
</tr>
</tbody>
</table>

(c) computation time (millisecond) for PROTEINS dataset.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (0.1)</th>
<th>Ratio (0.2)</th>
<th>Ratio (0.4)</th>
<th>Ratio (0.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Reg.</td>
<td>22.3</td>
<td>23.1</td>
<td>25.2</td>
<td>26.5</td>
</tr>
<tr>
<td>Simpler Reg.</td>
<td>21.4</td>
<td>21.6</td>
<td>21.7</td>
<td>22.2</td>
</tr>
</tbody>
</table>

(d) computation time (millisecond) for NCI1 dataset.

representations of graphs. We also report the training time of a single graph by changing the pooling ratio along with datasets to compare time complexities between original and simpler version of regularization. (Table 3.5).

3.5.3 Image Retrieval Task with Visual Scene Graph

In this subsection, we apply SSGPool to perform image retrieval via visual scene graph matching. A visual scene graph, initially proposed in (Johnson et al., 2015), represents contents of an image in the form of a graph consisting of three kinds of components: objects, their attributes, and relationships between two objects. Each node in a visual scene graph denotes one of the three components and each edge denotes the association of two components. Since the symbolic
Table 3.6: The results of image retrieval in terms of NDCG. Higher the NDCG score is, better the performance. DiffPool* denotes DiffPool with additional losses originally proposed in (Ying et al., 2018).

<table>
<thead>
<tr>
<th>Model</th>
<th>NDCG 5</th>
<th>NDCG 10</th>
<th>NDCG 20</th>
<th>NDCG 30</th>
<th>NDCG 40</th>
<th>NDCG 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet152</td>
<td>0.7198</td>
<td>0.7284</td>
<td>0.7420</td>
<td>0.7561</td>
<td>0.7708</td>
<td>0.7864</td>
</tr>
<tr>
<td>GNNs</td>
<td>0.7853</td>
<td>0.7993</td>
<td>0.8202</td>
<td>0.8361</td>
<td>0.8495</td>
<td>0.8619</td>
</tr>
<tr>
<td>SAGPool</td>
<td>0.7887</td>
<td>0.8032</td>
<td>0.8242</td>
<td>0.8393</td>
<td>0.8523</td>
<td>0.8654</td>
</tr>
<tr>
<td>DiffPool*</td>
<td>0.7904</td>
<td>0.8045</td>
<td>0.8249</td>
<td>0.8400</td>
<td>0.8531</td>
<td>0.8654</td>
</tr>
<tr>
<td>SSGPool</td>
<td><strong>0.7957</strong></td>
<td><strong>0.8103</strong></td>
<td><strong>0.8296</strong></td>
<td><strong>0.8441</strong></td>
<td><strong>0.8570</strong></td>
<td><strong>0.8692</strong></td>
</tr>
</tbody>
</table>

structure of an image is well reflected in a visual scene graphs, it is effective to use visual scene graphs in tasks such as question answering (Hudson and Manning, 2019b) and image captioning (Yang et al., 2019). Visual scene graphs also can be used to build an image-to-image retrieval system (Gordo and Larlus, 2017), which returns a list of images sorted by relevance with respect to an image query. In an image retrieval system based on a visual scene graph, the relevance measure is defined as a degree of matching between visual scene graphs. The matching between two visual scene graphs can be evaluated by computing their cosine similarity between embedded visual scene graphs, either annotated by a human or algorithmically generated, into a fixed-length vector.

To train and evaluate the image retrieval system, we need a ground truth measure of image relevance, possibly obtained from human annotations. However, it is impossible to gather annotations for all image pairs in the dataset. Following prior work (Gordo and Larlus, 2017) which demonstrates that the similarity between image captions is highly correlated to the human rating of image relevance, we utilize caption similarity as a proxy metric during our experiment. We use S-BERT, a transformer pretrained to generate sentence embeddings, to compute the similarity between captions. A proxy relevance
measure between two images is obtained by first computing S-BERT representations of the captions and then obtaining the cosine similarity between them. The resulting cosine similarity is clipped at zero to ensure the measure to be positive. With the proxy relevance score defined, Normalized Discounted Cumulative Gain (NDCG) is used to measure the performance of retrieval.

The proxy relevance score also provides supervision for learning graph representation. In every iteration, a batch of training image pairs (and corresponding visual scene graph pairs) are sampled, and the squared error between the cosine similarity of embeddings in each pair and their proxy relevance score is minimized. To obtain both captions and scene-graphs for images, we use 48,220 images which belong to both MS COCO dataset (Lin et al., 2014) and Visual Genome (VG) dataset (Krishna et al., 2017b). Following the Stanford split (Xu et al., 2017), we manually split the VG-COCO dataset with 36,601 train, 1,000 validation and 5,000 test images.

We use ResNet152 (Simonyan and Zisserman, 2014b), GNNs without pooling, DiffPool and SAGPool are chosen as comparative baselines. Table 3.6 shows the performance on the image retrieval task. Among the overall models, the SS-GPool achieves the best results over all NDCG scores.

As noted in Section 3.1, it is important to learn the compositional hierarchy for visual scene graphs. To compare the learned hierarchical structure among the graph pooling methods, we visualize the coarsening results of each model (Figure 3.3). As shown in the first column, SSGPool coarsens the graph by reflecting the structural information well. Due to this characteristic, the trees and their attributes (leaf-green) are coarsened to a single node, and deer eating grass and zebra are coarsened to another node. Furthermore, it can be seen that our method successfully maintains the global structure of the original graph in the upper layer. In the case of DiffPool taking the coarsening form like our
Figure 3.3: Left: An original image corresponding to the scene graphs on the right. Right: Pooling results on nodes of each graph in each layer. Same color of nodes are meant to be mapped to the same coarsened node in the pooled layer. Since DiffPool coarsens the graph with soft assignment matrix, we selected a top-1 coarsened node for each original node for visualization. The grey colored nodes in layer-2 are left-over coarsened nodes that were not chosen as top-1 by any original nodes. Some significant node labels are specified to demonstrate different properties between the pooling methods.
Figure 3.4: Other qualitative examples of pooling results on SSGPool, DiffPool and SAGPool. As Figure 3.3, same color of nodes are meant to be mapped to the same coarsened node in the pooled layer.
method, however, nodes with similar features tend to be coarsened together. Also, as DiffPool has a dense coarsening matrix, the upper layer graph cannot reflect the original graph structure and has the form of a fully connected graph. Lastly, the SAGPool constitutes hierarchies by selecting important nodes. We can see that it selects important nodes (e.g., eating, deer, zebra) but loses considerable amounts of other peripheral information. Additionally, SAGPool’s upper layer graph loses structural information from the original graph due to it is masking out all other nodes not selected. We attach more examples of qualitative results in Figure 3.4.

3.6 Conclusions

In this Chapter, we proposed the end-to-end graph pooling method, Spectrally Similar Graph Pooling. In contrast to previous work, our method learns compositional hierarchies while preserving the global structure of the graph. The proposed method shows competitive results not only in graph benchmarks datasets, but in real-world problem such as image retrieval with visual scene graphs. We also show that our proposed method learns meaningful hierarchical structures.
Chapter 4

Compositional Learning for Temporal Structure: Cut-Based Graph Learning Networks

4.1 Introduction

One of the fundamental problems in learning temporal video data is to find semantic structures underlying sequences for better representation learning. As most semantic flows cannot be modeled with simple temporal inductive biases, i.e., Markov dependencies, it is crucial to find the complex temporal semantic structures from long sequences to understand the video data. We believe there are two ingredients to solving this problem: segmenting the whole long-length sequence into (multiple) semantic units and finding their compositional structures. In this work, we propose Cut-based Graph Learning Networks (CB-GLNs), a new graph-based method which can discover composite semantic flows in video inputs and utilize them for representation learning of videos. The compositional semantic structures are defined as multilevel graph forms, which make
it possible to find long-length dependencies and their hierarchical relationships effectively.

The proposed method learns representations of video by discovering the compositional structure in multilevel graph forms. A single video data input is represented as a graph, where nodes and edges represent frames of the video and relationships between all node pairs. From the input representations, the CB-GLNs find temporal structures in the graphs with two key operations: temporally constrained normalized graph-cut and message-passing on the graph. A set of semantic units is found by parameterized kernel and cutting operations, then representations of the inputs are updated by message passing operations.

We thoroughly evaluate our method on the large-scale video theme classification task, YouTube-8M dataset (Abu-El-Haija et al., 2016). As a qualitative analysis of the proposed model, we visualize compositional semantic dependencies of sequential input frames, which are automatically constructed. Quantitatively, the proposed method shows a significant improvement on classification performance over the baseline models. Furthermore, as an extension of our work, we apply the CB-GLNs to another video understanding task, Video Question and Answering on TVQA dataset (Lei et al., 2018). With this experiment, we show how the CB-GLNs can fit into other essential components such as attention mechanism, and demonstrate the effectiveness of the structural representations of our model.

The remainder of the paper is organized as follows. Firstly, related work of previous approaches for temporal data are introduced. As preliminaries, basic concepts of graph-cut mechanisms and graph representation learning method are shown. Next, the problem statement of this paper is described to make further discussion clear. After that, the proposed Cut-Based Graph Learning Networks (CB-GLNs) are suggested in detail and the experimental results with
the real datasets, YouTube-8M and TVQA are presented.

4.2 Preliminaries

In this section, basic concepts related to graphs are summarized. First, mathematical definitions and notations of graphs are clarified. Second, the normalized graph-cut method is described. Lastly, Message Passing Neural Networks (MPNNs) for graph representation learning are introduced.

4.2.1 Normalized Graph-cut

A graph $G = (V, E)$ can be partitioned into two disjoint sets $V_1, V_2$ by removing edges connecting the two parts\(^1\). The partitioning cost is defined as the total weight of the edges that have been removed. In addition to the cut cost, the normalized graph-cut method (Shi and Malik, 2000) considers the total edge weight connecting a partition with the entire graph (the degree of the partition) to avoid the trivial solutions which can make extremely imbalanced clusters. The objective of the normalized graph-cut can be formally described as follows.

\[
Ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_1, V_2)}{assoc(V_2, V)}
\]

(4.1)

with

\[
cut(V_1, V_2) = \sum_{v_1 \in V_1, v_2 \in V_2} w(v_1, v_2)
\]

(4.2)

\[
assoc(V_1, V) = \sum_{v_1 \in V_1, v \in V} w(v_1, v)
\]

(4.3)

where $w(v_1, v_2)$ is an edge weight value between node $v_1$ and $v_2$. It is formulated as a discrete optimization problem and usually relaxed to continuous, which can be solved by eigenvalue problem with the $O(n^2)$ time complexity. By applying

\(^1V_1 \cup V_2 = V\) and $V_1 \cap V_2 = \emptyset$
the cut method recursively, an input graph is divided into fine-grained subgraphs.

4.2.2 Message Passing Neural Networks

To learn representations of graph, (Gilmer et al., 2017) suggested the message passing neural networks (MPNNs), which encompass a number of previous neural models for graphs under a differentiable message passing interpretation.

In detail, the MPNNs have two phases, a message passing phase and a update phase. In the message passing phase, the message for vertex $v$ in $l$-th layer is defined in terms of message function $M^l$:

$$m^l_v = \sum_{w \in N(v)} M^l(h_v, h_w)$$ (4.4)

where in the sum, $N(v)$ denotes the neighbors of $v$ in a graph $G$. With the message $m^l_v$, the representations of vertex $v$ in $(l+1)$-th layer are obtained via the update function $U^l$.

$$h^{l+1}_v = U^l(h^l_v, m^l_v)$$ (4.5)

The message functions $M^l$ and the update functions $U^l$ are differentiable so that the MPNNs can be trained in an end-to-end fashion. As extensions, there are some attempts to have been made to improve message passing by putting a gating or attention mechanism into a message function, which can have computational benefits (Monti et al., 2017; Duan et al., 2017; Hoshen, 2017; Veličković et al., 2018; Satorras and Estrach, 2018; van Steenkiste et al., 2018).

We further note other previous research for learning a structure of a graph. Neural Relational Inference (NRI) (Kipf et al., 2018) used a Variational Autoencoder (VAE) (Kingma and Welling, 2013) to infer the connectivity between nodes with latent variables. Other generative models based approaches also have been well studied (Bojchevski et al., 2018; De Cao and Kipf, 2018; Simonovsky et al., 2018).
and Komodakis, 2018). However, those suffer from availability of structural information in training data or have complex training procedures.

4.3 Problem Statement

The problem to be tackled in this work can be clearly stated with the notations in the previous section as below.

We consider videos as inputs, and a video is represented as a graph $G$. The graph $G$ has nodes corresponding to each frame respectively in the video with feature vectors and the dependencies between two nodes are represented with weight values of corresponding edges.

Suppose that video data $X$ has $N$ successive frames and each frame has an $m$-dimensional feature vector $x \in \mathbb{R}^m$. Each frame corresponds to a node $v \in V$ of graph $G$, and the dependency between two frames $v_i, v_j$ is represented by a weighted edge $e_{ij} \in E$. From $G = (V, E)$, the dependency structures among video frames are defined as the weighted adjacency matrix $A$, where $A_{ij} = e_{ij}$.

With aforementioned notations and definitions, we can now formally define the problem of video representations learning as follows:

Given the video frames representations $X \in \mathbb{R}^{N \times m}$, we seek to discover a weighted adjacency matrix $A \in \mathbb{R}^{N \times N}$ which represents dependency among frames.

$$f : X \rightarrow A \quad (4.6)$$

With $X$ and $A$, final representations for video $h \in \mathbb{R}^l$ are acquired by $g$.

$$g : \{X, A\} \rightarrow h \quad (4.7)$$

The obtained video representations $h$ can be used for various tasks of video understanding. In this chapter, the video theme classification and the video question and answering tasks are mainly considered.
4.4 Cut-Based Graph Learning Networks

The Cut-Based Graph Learning Networks (CB-GLNs) consist of two sub-modules: a structure learning module with the graph-cuts and a representation learning module with message-passing operations. The key idea of the method is to find inherent semantic structures using the graph-cuts and to learn feature vectors of the video with the message-passing algorithm on the semantic structures. Stacking these modules leads to the subsequent discovery of compositional structures in the form of a multilevel graph. Figure 4.1(a) illustrates the whole structure of the CB-GLNs. In the next sections, operations of each of these modules are described in detail.

4.4.1 Structure Learning Module

In the structure learning module, the dependencies between frames $\hat{A}$ are estimated via parameterized kernels and the temporally constrained graph-cut.

As the first step, the initial temporal dependencies over all frames are con-
structured via the parameterized kernel $\mathcal{K}$:

$$\hat{A}_{ij} = \mathcal{K}(x_i, x_j) = ReLU\left(f(x_i)^T f(x_j)\right)$$

where $f(x)$ is a single-layer feed-forward network without non-linear activation:

$$f(x) = W_f x + b_f$$

with $W_f \in \mathbb{R}^{m \times m}$ and $b_f \in \mathbb{R}^m$.

Then, as the second step, the meaningful dependency structure among all pairwise relationships is refined by applying normalized graph-cut to the $\hat{A}$. The objective of the normalized graph-cut for CB-GLNs is:

$$Ncut(V_1, V_2) = \frac{\sum_{v_i \in V_1, v_j \in V_2} \hat{A}_{ij}}{\sum_{v_i \in V_1} \hat{A}_{ii}} + \frac{\sum_{v_i \in V_1, v_j \in V_2} \hat{A}_{ij}}{\sum_{v_j \in V_2} \hat{A}_{jj}}$$

To reduce the complexity of the equation (4.10) and to keep the inherent characteristics of the video data, an additional constraint is added to the normalized graph-cut. As the video data is composed of time continuous subsequences, no two partitioned sub-graphs have an overlap in physical time. This characteristic is implemented by applying the temporal constraint (Rasheed and Shah, 2005; Sakarya and Telatar, 2008) as follows.

$$(i < j \text{ or } i > j) \text{ for all } v_i \in V_1, v_j \in V_2$$

(4.11)

By virtue of the temporal constraint, the cut operation can only divide the video into two successive parts and each part should be composed of the consecutive frames. Therefore, we only need to compare $N - 1$ cost of cut, a linear-time computational complexity. It is important to note that, without the temporal constraint, the eigen-decomposition should be executed for every cut operation, which not only has $O(N^2)$ time complexity, but also is unstable to train.
The graph-cut can be recursively applied to the $\hat{A}$, so the $\hat{A}_{\text{cut}}$ with removed edges and $K$ partitioned sub-graphs can be obtained. The number of cut operations is determined by the length of the video $N$, $\lfloor \log_2 \sqrt{N} \rfloor$. Figure 4.1(b) depicts the detailed operations of the structure learning module.

**End-to-end Trainability with Graph-cut**

After determining which edges to cut (via recursive graph-cut), the weight of cut edge in $\hat{A}$ is masked to 0 so that the gradients only flow through the surviving edges (similar to ReLU or Max pooling). These surviving gradients are flowed to the kernel in Equation (4.8) so that the function $f$ in the kernel can be updated through gradient descent in an end-to-end fashion. Also, as the kernel is updated, the cut boundary is changed (to get optimal sub-graph) in the training phase. It is worthwhile to note that the graph-cut mechanism which chooses the edge to be cut is not optimized through the gradient descent, instead it is deterministically obtained based on the estimated edge weight in $\hat{A}$.

**4.4.2 Representation Learning Module**

After estimating the weighted adjacency matrix $\hat{A}_{\text{cut}}$, the representation learning module updates the representations of each frame via a differentiable message-passing framework (Gilmer et al., 2017). For the message function $F_M$, we simply use the weighted sum of adjacent nodes’ representations after linear transformation similar to (Kipf and Welling, 2016):

\[
M = F_M(X, \hat{A}_{\text{cut}}) = D^{-1} \hat{A}_{\text{cut}} X W_M
\]  

(4.12)

where $D$ is a degree matrix of the graph and $\hat{A}_{\text{cut}}$ is an adjacency matrix after cut operations.
For the update function $F_U$, we integrate the message with node representations by using low-rank bilinear pooling (Kim et al., 2017a) followed by a position-wise fully connected network.

\[ Z = F_U(X, M) \]
\[ = LN(\sigma(Z'W^Z + b^Z) + Z') \]  
\[ Z' = LN(f(XW_{U_1} \circ MW_{U_2}) + X) \]  

(4.13)

where $f$ is a single-layer position-wise fully connected network. We also employ a residual connection (He et al., 2016) around each layer followed by layer normalization (Ba et al., 2016).

Once the representations of all frames are updated, a pooling operation for each partitioned sub-graph is applied. Then we can obtain higher level representations $H \in \mathbb{R}^{K \times m}$, where $K$ is the number of partitioned sub-graphs (Figure 4.1(c)). If we have additional information such as query (e.g. a question feature vector in video QA setting), we can pool the sub-graph with attentive pooling similar to (Santos et al., 2016). In the same way, $H$ is fed into the next layer and we can get the final video-level representation $h_f \in \mathbb{R}^m$.

### 4.5 Experiments

In this section, we thoroughly evaluate the proposed model to video theme classification task with YouTube-8M dataset (Abu-El-Haija et al., 2016). Also, to show the applicability of our model, we adapt our model with attention mechanism and apply to video question and answering task with TVQA dataset (Lei et al., 2018).

#### 4.5.1 Baselines

**Average pooling method:** The average pooling method is the simplest way to make a representative video-level feature vector from the frame-level ones. It
takes average values over all frame-level features, and then the average valued video-level vector is used as the input of a classifier module.

**Recurrent Neural Networks:** Four RNNs’ variants are compared with the suggested model: Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Gated Recurrent Units (GRU) (Chung et al., 2014), bi-directional LSTM (BiLSTM) and bi-directional GRU (BiGRU). They naturally take temporal input frame by frame and learn to capture the transition between successive frames.

**1D Convolutional Neural Networks:** To model temporal sequences of adjacent frames and their compositions, 1-dimensional CNNs with different kernel size are examined. For the comparison, 3, 5, and 7 size of kernel are used.

**Clustering methods:** Two variants of clustering methods have recently been suggested to get a set of visual descriptors from an image sequence. Deep Bag of Frame (DBoF) (Abu-El-Haija et al., 2016) randomly selects $k$ frames from all feature vectors, and the selected vectors are fed into a fully connected non-linear layer followed by pooling operations. NetVLAD (Arandjelovic et al., 2016) clusters all features into $k$ clusters and calculates residuals, which are difference vectors between the feature vectors and its corresponding cluster center, as the new representations for each feature vector. And then vectors in the same cluster is aggregated into a vector and fed into a fully connected non-linear layer followed by pooling operations.

**Self-attention methods:** When the size of the input data is large and complex, the attention mechanisms are widely used to give weights to informative regions and to reduce the model complexity. Recently, (Vaswani et al., 2017; Yu et al., 2018) proposed a self-attention mechanism to expand the attention mechanism to temporal data. It sets the query and the key as same, so it simply
finds a weight of each values of frames based on the similarities between the all 
pair of keys of frames.

4.5.2 Video Theme Classification Task on YouTube-8M

Data Specification

YouTube-8M (Abu-El-Haija et al., 2016) is a benchmark dataset for video 
understanding, where the main task is to determine the key topical themes of 
a video. It consists of 6.1M video clips collected from YouTube and the video 
inputs consist of two multimodal sequences, which are the image and audio. 
Each video is labeled with one or multiple tags referring to the main topic of 
the video. The dataset split into three partitions, which are 70% for training, 
20% for validation and 10% for test. As we have no access to the test labels, all 
results in this chapter are reported for validation set.

Each video is encoded at 1 frame-per-second up to the first 300 seconds. As 
the raw video data is too huge to be treated, each modality is pre-processed with 
pretrained models by the author of the dataset. More specifically, the frame-level
visual features were extracted by inception-v3 network (Szegedy et al., 2016) trained on imagenet and the audio features were extracted by VGG-inspired architecture (Hershey et al., 2017) trained for audio classification. PCA and whitening method are then applied to reduce the dimensions to 1024 for the visual and 128 for audio features.

Global Average Precision (GAP) is used for the evaluation metric for the multi-label classification task as used in the YouTube-8M competition. For each video, 20 labels are predicted with confidence scores. Then the GAP score computes the average precision and recall across all of the predictions and all the videos.

**Implementation Details**

All models performed in this chapter are trained using the stochastic gradient descent method with Adam optimizer (Kingma and Ba, 2014) and 128-size mini-batches. The models are trained for 5 epochs to converge and the learning rate is initially set to 0.001 and is then decreased exponentially with the factor of 0.5 every 4M samples. The dimension size of the all hidden layers of the models is $m = 1024$. After getting the final representation $h$, a simple logistic regression is used as the classifier. To avoid over-fitting, we use $l_2$ regularizer to the classifier module. Also, dropout with ratio of 0.5 and 0.1 are applied to the classifier module and all other layers, respectively.

In the clustering methods, i.e. DBoF and NetVLAD, we randomly sample $K = 4096$, $K = 256$ frame-level features respectively, with replacement from each video. $K$ is fixed for all videos in the training and testing phases. For the others, i.e. RNN-based, CNN-based, self-attention and the proposed model, 2-stacked layers are used. The average pooling operation is then applied to the last layer output vectors to form a video-level feature. All models described
Experimental Results

Firstly, we evaluate the classification performance of the proposed model against five types of representative sequential model baselines and two state-of-the-art models (Lin et al., 2018; Mao et al., 2018) previously reported. The results with GAP score are summarized in Table 4.1. The proposed model considerably outperforms all the comparative models. The second best performing model is the self-attention based approach, followed by RNNs, CNNs and Clustering based approaches.

As (Gulcehre et al., 2018) mentioned the similarity between attention mechanism and graph neural networks from a message-passing point of view, it is...
worth comparing CB-GLNs and the self-attention method more thoroughly. Because the self-attention method learns representations with the obtained attention map, we visualize the attention map of the self-attention method and adjacency matrix of CB-GLNs to compare the characteristics of the models (Figure 4.3). Figure 4.3(a) and (b) show that the CB-GLNs can capture structural information of the video, effectively. However, it is hard to see that the self-attention is good to learn the inherent structures of video (in Figure 4.3(c) and (d)). By virtue of hierarchical structures, the CB-GLNs combine correlated frames effectively and propagate information to get better representation.

Figure 4.3: Visualizations of adjacency matrices of CB-GLNs (a and b) and attention maps of the self-attention mechanism (c and d).
Table 4.2: An ablation study of Cut-Based Graph Learning Networks.

<table>
<thead>
<tr>
<th>Ablation model</th>
<th>GAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) layer normalization</td>
<td>0.8486</td>
</tr>
<tr>
<td>(b) residual connection</td>
<td>0.8447</td>
</tr>
<tr>
<td>(c) residual connection with layer normalization</td>
<td>0.8370</td>
</tr>
<tr>
<td>(d) graph-cut after learning representations</td>
<td>0.8576</td>
</tr>
<tr>
<td>CB-GLNs</td>
<td>0.8597</td>
</tr>
</tbody>
</table>

For ablation studies, we firstly selected three critical characteristics of CB-GLNs for in-depth analysis: layer normalization, residual connection and graph-cut after learning representations. The GAP scores on the validation set for each ablation experiments are shown in Table 4.2. As can be seen from (a) to (c) in the Table 4.2, the residual connections followed by layer normalization are crucial for the representation learning module. Also, to see the effect of sparsifying an adjacency matrix via the graph-cut, reversed order of representation learning and graph-cut is also conducted ((d) in Table 4.2). By doing so, the representations of each node are updated with $\hat{A}$ obtained only using kernel $K$ (in Equation 4.8) and graph-cut algorithm is used just for sub-graph pooling. Thus, the model has to learn representations with dense and noisy connections, degrading the performance of the model. From this result, we can argue that the temporally constrained graph-cut effectively reduce the noisy connections in the graph.

Secondly, to see the effect cut operations on the actual computing time, we report the training time of a single batch by changing the number of cut operation. As shown in Table 4.3, the difference of the time for forward-backward propagation in a mini batch (size: 128) between no-cut model and 7-cut model is less than 100 milliseconds. Therefore, the time consumed for the cut operation
Table 4.3: Training time of CB-GLNs in a single batch by number of cut operations.

<table>
<thead>
<tr>
<th>Ablation model</th>
<th>time (s/batch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB-GLN w/o cut</td>
<td>0.481</td>
</tr>
<tr>
<td>CB-GLN w/ 1 cut</td>
<td>0.497</td>
</tr>
<tr>
<td>CB-GLN w/ 3 cuts</td>
<td>0.508</td>
</tr>
<tr>
<td>CB-GLN w/ 7 cuts</td>
<td>0.556</td>
</tr>
</tbody>
</table>

is almost neglectable. (Consuming time for a single cut is less than 4% over the whole process).

**Qualitative Results on Learned Compositional structure**

In this section, we demonstrate compositional learning capability of CB-GLNs by analyzing constructed multilevel graphs. To make further discussion clear, four terms are used to describe the compositional semantic flows: semantic units, scenes, sequences and a video for each level. In Figure 4.4, a real example with the usage of video titled “Rice Pudding” is described to show the results.

In Figure 4.4(a), the learned adjacency matrices in each layer are visualized in gray-scale images: the two leftmost images are from the 1st layer and the two rightmost images are from the 2nd layer. To denote multilevel semantic flows, four color-coded rectangles (blue, orange, red and green) are marked and those colors are consistent with Figure 4.4(b).

Along with diagonal elements of the adjacency matrix in the 1st layer (Figure 4.4(a)-1), a set of semantic units are detected corresponding to bright blocks (blue). Interestingly, we found that each semantic unit contains highly correlated frames. For example, the #1 and #2 are each shots introducing the YouTube cooking channel and how to make rice pudding, respectively. The #4

and #5 are shots showing a recipe of rice pudding and explaining about the various kinds of rice pudding. The #6 and #7 are shots putting ingredients into boiling water in the pot and bringing milk to boil along with other ingredients. At the end of the video clip, #11 is a shot decorating cooked rice pudding and #12 is an outro shot that invites the viewers to subscribe.

These semantic units compose variable-length scenes of the video, and each scene corresponds to a sub-graph obtained via graph-cut (Figure 4.4(a)-2.). For example, #13 is a scene introducing this cooking channel and rice pudding. Also, #15 is a scene of making rice pudding with detailed step by step instructions, and #16 is an outro scene wrapping up with cooked rice pudding. The 1st-layer of the model updates representations of frame-level nodes with these dependency structures, then aggregates frame-level nodes to form scene-level nodes (Layer 1 in the Figure 4.4(b)).

In Figure 4.4(a)-3 and (a)-4, the sequence-level semantic dependencies (red) are shown. #17 denotes a sequence of making rice pudding from beginning to end, which contains much of the information for identifying the topical theme of this video. Finally, the representations of scenes are updated and aggregated to get representations of the whole video (Layer 2 in the Figure 4.4(b)).

4.5.3 Video Question & Answering Task on TVQA

Data Specification

TVQA (Lei et al., 2018) is a video question and answering dataset on TV show domain. It consists of total 152.5k question-answer pairs on six TV shows: The Big Bang Theory, How I Met Your Mother, Friends, Grey’s Anatomy, House and Castle. Also, it contains 21.8k short clips of 60-90 seconds segmented from the original TV show for question-answering. The provided inputs are 3 fps image frames, subtitles and multiple choice questions with 5 candidate answers
Figure 4.4: An example of the constructed temporal dependency structures for a real input video in YouTube-8M, titled “Rice Pudding” (https://youtu.be/cD3enxnS-JY) is visualized. The topical themes (true labels) of this video are {Food, Recipe, Cooking, Dish, Dessert, Cake, baking, Cream, Milk, Pudding and Risotto}. (a): Learned adjacency matrices in the layer 1 and 2 are visualized. The strength of connections are encoded in a gray-scale where 1 is white and 0 is black. (a)-1: 12 bright blocks in layer 1 are detected (blue rectangles), each block (highly connected frames) represents a semantic unit. (a)-2: Sub-graphs of the input are denoted by orange rectangles. It shows that semantically meaningful scenes are found by temporally constrained graph-cut. (a)-3 and (a)-4: learned high-level dependency structures in layer 2 are revealed with red and green rectangles. (b): The whole composite temporal dependencies are presented.
for each question, for which only one is correct.

The questions in the dataset are localized to a specific sub-part in the video clips by restricting questions to a composition of two parts, e.g., "Where was Sheldon sitting / before he spilled the milk?". Models should answer questions using both visual information and associated subtitles from the video.

**Implementation Details**

The input visual features were extracted by the pooled 2048D feature of the last block of ResNet101 (He et al., 2016) trained on imagenet and the text-based features for subtitles, questions and answers were extracted by GloVe (Pennington et al., 2014). The visual features and subtitle features are manually aligned with time-stamp and answer features also aligned with visual and subtitle features by attention mechanism to construct input $X$. Then the $X$ is fed into the CB-GLNs to extract final representations $h$. Different from the YouTube-8M dataset case, we use a question feature vector as a query of attentive pooling, so that the representations of the sequence are pooled via weighted sum with the attention values.
Table 4.4: Comparison on classification accuracy with the GAP measure on validation dataset. Logistic regression is as a classifier for all of the presented methods.

<table>
<thead>
<tr>
<th>Frame-level model</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiGRU (2 layers)</td>
<td>63.14</td>
</tr>
<tr>
<td>1D CNN (2 layers, kernel size 3)</td>
<td>61.46</td>
</tr>
<tr>
<td>Self-Attention(4 heads, 2 layers)</td>
<td>61.08</td>
</tr>
<tr>
<td>CB-GLNs (2 layers)</td>
<td>61.76</td>
</tr>
</tbody>
</table>

**Experimental Results**

With the TVQA dataset, our main intention is to show how the CB-GLNs can be worked with other modules (especially the attention mechanism) to various video-based tasks, and also what characteristics our model. Therefore, we gives a simple evaluation result of the question answer accuracy of the proposed model against three types of representative sequential model baselines. The results with accuracy are summarized in Table 4.4. As shown in Table 4.4, the Bi-GRU shows the best result followed by CB-GLNs. As noted in the original paper (Lei et al., 2018), it is because the language (script) is much more informative than the video itself in the TVQA task.

**Qualitative Results on Attention Hierarchy with Learned Compositional Structure**

In this section, we show how the attention mechanism can be fit into the CB-GLNs to learn representations more effectively. Basically, the attention mechanism places more weight on important parts to aggregate values, given a query. By virtue of the compositionality, the attention mechanism can be naturally applied to the CB-GLNs in a hierarchical fashion. Figure 4.6 presents learned attention hierarchy in a real video clip of “Friends”. In this example, The ques-
Q: What did Chandler say he was going to get, when he got up?  
A: Chandler said he was going to get cigarettes.

Figure 4.6: An example of the learned attention hierarchy for a real video clip of “Friends” in TVQA is visualized. The question and answer for this clip are “What did Chandler say he was going to get, when he got up?” and “Chandler said he was going to get cigarettes.” each. We dropped the Scene 1 and Scene 2 parts in (a) and (b), because the attention values in these scenes are nearly identical, making them non-informative. (a): Attention maps for each layer given query. the orange rectangles in layer 1 denote the cut scenes discovered by CB-GLNs and the red rectangle in layer 2 highlights scenes with high attention values given a question. (b): Cumulative attentions to the frame-level are visualized by multiplying attention values in layer 1 & 2. (c): Visualization of the learned adjacency matrices in layer 1 & 2. The orange and red rectangles are consistent with (a).
tion is “What did Chandler say he was going to get, when he got up?” and the answer for the question is that “Chandler said he was going to get cigarettes.”.

In Figure 4.6(a), we can see that scenes with high attention values (Scene 5 and Scene 6 (coded by a red rectangle)) in layer 2 are aligned well with localized video portions relevant to a given question. Scene 4, where “Chandler is searching his pocket to find cigarettes while sitting down”, is also in the localized section. Therefore, we can say our model finds a sensitive portion of the video relevant to the given question. In layer 1 (Figure 4.6(a)), the model gives a keener attention to the frame-level within each scenes (coded by orange rectangles), such as a moment where Chandler gets up or Chandler yells ‘I gotta smoke’.

Because the attention operation for each frame is conducted hierarchically, we can calculate cumulative attention values by multiplying them in layer 1 and layer 2. Figure 4.6(b) shows the cumulative attention values for each frame. The most important frame in a viewpoint of the model is the ”getting up moment” frame because the model should answer the question by identifying the meaning of “when he got up” in the question.

In Figure 4.6(c), the learned adjacency matrices in each layer are visualized. Same color-coded rectangles with (a) are used for cut scenes (orange rectangles in layer 1) and scenes with high attention values (a red rectangle in layer 2). We also coded red rectangle in layer 1, which is corresponding to scenes with high attention values in layer 2. Even though scene 5 and scene 6 in the frame level (layer 1) are considerably short when compared to the whole video sequences, the CB-GLNs can find important moments and aggregate them in an effective way.
4.6 Conclusions

In this chapter, we proposed Cut-Based Graph Learning Networks (CB-GLNs) which learn not only the representations of video sequences, but also composite dependency structures within the sequence. To explore characteristics of CB-GLNs, various experiments are conducted on a real large-scale video dataset YouTube-8M and TVQA. The results show that the proposed model efficiently learns the representations of sequential video data by discovering inherent dependency structure of itself.
Chapter 5

Dataset for Learning Spatiotemporal Structure: Multilevel Video QA

5.1 Introduction

A story is a series of events across multiple scenes centered around a succession of character’s actions. Given that humans communicate regularly and naturally through stories, the ability to understand stories is a crucial part of human intelligence that sets humans apart from others (Szilas, 1999; Winston, 2011).

Among various kinds of narratives, video is considered one of the best narrative mediums for developing human-level AI algorithms from two points of view. Firstly, they have multiple modalities such as a sequence of images, audio (including dialogue, sound effects, and background music) and text (subtitles or added comments). Secondly, they show cross-sections of everyday life by demonstrating socioculturally appropriate behaviors through the characters. However, understanding video stories is considered to be challenging to current machine
learning methods, due to the causal and temporal relationships between events, which can be complex and are often left implicit (Riedl, 2016).

One way to enable a machine to understand a video is to train the machine to answer questions about the video (Schank and Abelson, 2013; Mueller, 2004). To train the model, a large number of question-answer pairs for video are required. While several recent studies have suggested Video Question Answering (Video QA) datasets (Tapaswi et al., 2016; Kim et al., 2017b; Mun et al., 2017; Jang et al., 2017b; Lei et al., 2018), these datasets do not give sufficiently careful consideration of “understanding of video”. For such reason, the previously released video QA datasets are highly-biased and lack of variance in question difficulty. However, the construction of QA dataset with hierarchical difficulty levels in terms of understanding is crucial, as people with different perspectives (or different intelligence levels) will understand the given video differently.

In this work, we propose two criteria, memory capacity and logical complexity for video understanding. The two criteria can leverage the understanding of developmental stages of human intelligence. Base on the criteria, we construct a multilevel Video QA dataset for more comprehensive understanding of video. We argue that the different level of questions needs different inductive bias to answer to questions. Additionally, the constructed multilevel Video QA dataset provides rich annotations for main characters such as visual bounding boxes, behaviors, and emotions of main characters and also coreference resolved scripts.

5.2 Related Work

5.2.1 Datasets for Video Understanding

Video understanding area has been actively studied and several tasks including datasets have been proposed such as action recognition (Kuehne et al., 2011;
Soomro et al., 2012; Karpathy et al., 2014; Sigurdsson et al., 2016; Caba Heilbron et al., 2015; Kay et al., 2017), video classification (Abu-El-Haija et al., 2016), video captioning (Xu et al., 2016; Miech et al., 2019; Krishna et al., 2017a), and spatio-temporal object segmentation (Perazzi et al., 2016). However, these researches focus on perceiving and recognizing visual elements so that they are not suitable for high-level visual reasoning. To circumvent this limitation, video question answering for short video clips are proposed as a benchmark for high-level video understanding (Rohrbach et al., 2017; Jang et al., 2017b; Mun et al., 2017). These works only dealt with a sequence of images about short video clips, which would not include a meaningful story. Unlike these video question answering, Video Story Question Answering focuses on the story narrative about video. A story is a sequence of events, which means video with meaningful stories contain a relatively long sequence of videos and consists of a series of correlated video events. Video Story Question Answering requires the ability to discriminate what’s meaningful in a very long video, and also requires visual processing, natural language, and additional acoustic modeling. Recently, there have been some studies proposing datasets for use in the domain of video story understanding (Tapaswi et al., 2016; Kim et al., 2017b; Lei et al., 2018).

MovieQA (Tapaswi et al., 2016) uses the plot synopses of the movies to create Video QA datasets. In addition to movie clips, the dataset contains various data such as plots, subtitles and DVS. However, since QA pairs are generated based only on the given complicated plot synopses, it is not easy to answer the questions by using an insufficient amount of the dataset. The PororoQA (Kim et al., 2017b) dataset was created by directly watching animation videos (not real-world videos), making their QAs more tightly coupled to these videos. Most of these questions were very similar to subtitles and descriptions. TVQA (Lei
et al., 2018) is a large-scale video QA dataset based on 6 TV shows. TVQA aims to utilize both visual and language information from 60-90 second video clips. They also inform users which specific part of the video is needed for answering questions, which is essential ground truth for localization. Most of their questions are focused on relatively short moments (less than 15 seconds) which is not targeted for story understanding.

5.2.2 Cognitive Developmental Stages

In this section, we explain cognitive development of human based on one of Neo-Piagetian theory (Collis, 1975b) recasting of Piaget’s theory of developmental stages (Piaget, 1972). Piaget’s theory explains in detail the process by which human cognitive ability develops, in conjunction with information processing models. In order to justify three criteria proposed in this chapter in terms of human intelligence development, we examine the details of the developmental stages of Piaget’s theory. Piaget’s original model suggests a sensory-motor stage that occurs from birth, however, that stage involves only representations related to sensory-motor activity. Thus, we focus on the later stages that follow the pre-operational stage in which a child shows understanding behavior (Collis, 1975a).

- **Stage 1 (Pre-Operational Stage; 4 to 6 years)**: At this stage, a child thinks at a symbolic level, but is not yet using cognitive operations. The child can not transform, combine or separate ideas. Thinking at this stage is not logical and often unreasonable. Associations are made on the basis of emotion and preference at this stage, and it has a very egocentric sight of one’s own world.

- **Stage 2 (Early Concrete Stage; 7 to 9 years)**: At this stage, a child can
utilize only one relevant operation. Thinking at this stage has become detached from instant impressions and is structured around a single mental operation, which is a first step towards logical thinking.

- Stage 3 (Middle Concrete Stage; 10 to 12 years) : At this stage, a child can think by utilizing more than two relevant cognitive operations and acquire the facts of dialogues. This is regarded as the foundation of proper logical functioning. However, a child at this stage lacks own ability to identify general fact that integrates relevant facts into coherent one. Moreover, thinking at this stage is still concrete, not abstract.

- Stage 4 (Concrete Generalization Stage; 13 to 15 years) : Piaget referred to this stage as the early formal stage, particularly for abstract thinking. A child at this stage, however, can just generalize only from personal and concrete experiences. The child do not have own ability to hypothesize possible concepts or knowledge that is quite abstract.

- Stage 5 (Formal Stage; 16 years onward) : This stage is characterized purely by abstract thought. Rules can be integrated to obtain novel results that are beyond the individual’s own personal experiences. However, this is not a stage that every person can reach.

5.3 Two Criteria for Multilevel Video QA

This section describes two criteria as measures of video understanding. The two criteria are as follows: memory capacity, and logical complexity. Every question in the QA dataset is classified in each level by each criterion respectively, and assigned to the cognitive development stage according to Piaget’s theory (Piaget, 1972; Collis, 1975b). We explain this process in detail in the following subsections.
5.3.1 Criterion 1: Memory Capacity

When determining the difficulty of questions collected for the video, the length of the video is crucial for reasoning and finding the correct answer in machine learning perspective. If the length of the video required for answering the question is longer, the question can be classified as more difficult, and vice versa. For example, a question targeted to short video is a lot more difficult than a question targeted to an image frame, and a question targeted to entire video is a lot more difficult than a question targeted to one segment video. This criterion also can be interpreted as memory capacity of humans. In this chapter, we define Memory Capacity as the length of the target video which has to be considered to answer given question. We use the terms defined at each level consistently with the terms in (Zhai and Shah, 2006). The classification results are as follows.

- Level 1 (shot): The questions for this level are based on a video length less than about 10 seconds without change of viewpoint. This set of questions can contain atomic or functional/meaningful action in the video. Most recent datasets which deal with video belong to this level (Jang et al., 2017b; Maharaj et al., 2017; Mun et al., 2017). The questions of this level aim to evaluate understanding of information which contains video characteristics that are not in Level 1 (frame level). One important point is that target video for questions at this level contains meaningful actions. At this level, both atomic action and meaningful action can appear, and their boundary is vague. For example, waving hands (atomic action) and a gesture to saying goodbye (meaningful action) have a similar action, However, their meaning is different depending on the situation, not depending on video length. This level contains both of actions, even if their
difficulty is clearly different.

- Level 2 (scene): The set of questions for this level is based on clips that are 1-3 minutes long without place change. Videos at this level contain sequences of actions, which augment the level of difficulty from Level 2. We consider this level as the “story” level according to our working definition of story. MovieQA (Tapaswi et al., 2016) and TVQA (Lei et al., 2018) are the only datasets which belong to this level. For example, the popular TV sitcom *Friends* has 13 scenes per episode on average, and a movie has 120 scenes on average.

- Level 3 (sequence): The set of questions at this level is related to more than two scenes, but less than entire movie. To the best of our knowledge, there are no datasets dealing with the video at this level.

- Level 4 (entire): The question set for this level is based on an entire story from beginning to end. Questions at this level are based on whole video such as an entire movie or an episode of a drama.

### 5.3.2 Criterion 2: Logical Complexity

Complicated questions often require more (or higher) logical reasoning steps than simple questions. In other words, if a question requires multiple supporting facts which have interrelations to answer, we regard that the question has high logical complexity. For story-enabled intelligence, it is required to trace several logical reasoning steps by combining multiple supporting facts to give a correct answer to a given question. In a similar vein, if a question needs only a single supporting fact with a single relevant datum, we regard that it has low logical complexity. It may need only one reasoning step or one perception step to answer the question.
This subsection describes the second criterion *logical complexity* to define the level of difficulties for questions. We define five logical complexity levels based on the Stanford Mobile Inquiry-based Learning Environment (SMILE) (Seol et al., 2011). In the SMILE project, students learn online lectures or documents via a mobile platform and generate relevant questions based on what they have learned. Each question made by a student is classified into five logical complexity levels as follows:

- **Level 1 (Simple recall on one cue)**: The question set at this level can be responded with minimal cognitive effort, involving simple recall or simple arithmetic calculations. The questions at this level require only one supporting fact. Supporting fact is a triplet form of \{subject-relationship-object\} such as \{person-hold-cup\}. As the questions at this level are too simple, they may not trigger much interaction.

- **Level 2 (Simple analysis on multiple cues)**: The question set at this level can be responded with simple analysis of the question types or problems with simple reasoning. The questions at this level ask for factual information involving recall of independent multiple supporting facts, which trigger simple inference or quick interpretation. For example, two supporting facts \{tom-in-kitchen\} and \{tom-grab-tissue\} are referenced to answer “Where does Tom grab the tissue?”. This question set begins with simple question types starting with “Who”, “What”, “When”, “Where”, “How many”, and so on. Responses come from a range of clearly defined scope with little room for dispute.

- **Level 3 (Intermediate cognition on dependent multiple cues)**: The question set can be responded with intermediate level of cognition and analysis. The questions at this level require multiple supporting facts with time
factor. Time factor is a sequence of the situations or actions. Accordingly, the questions at this level cover how situations have changed and subjects have acted. It also requires cognitive operations such as comparison, classification, or categorization in responding to given questions at this level.

- **Level 4 (High-level reasoning for causality):** The question set at this level can be responded with higher-level of analysis and reasoning rather than a lower-level thinking question. The question set covers reasoning for causality beginning with “Why”. Reasoning for causality is the process of identifying causality, which is the relationship between cause and effect from actions or situations. It requires own interpretation or synthesis in responding to given questions at this level.

- **Level 5 (Creative thinking):** The question set at this level can be responded by requiring imagination and creation of new theory or hypothesis with supporting rationale. The question at this level covers creative thinking and reasoning that may help defining a new solution or concept that has not existed previously. For example, the questions (#19 and #20) in Table 1 on appendix draw an unique solution by formulating own rational equations about not occurred situations.

**5.3.3 Interpretation as Cognitive Development Stage**

Human understanding, as Piaget stated, can be classified into different stages. We propose that the concepts from Piaget’s theory of development correspond to the three-dimensional video Q&A Hierarchy criteria.

First, the development stage can be explained from the perspective of the memory capacity criterion. Memory capacity corresponds to working memory of
the cognitive process model. (Case, 1980a) suggested that the working memory available for problems increases with age, as does the space required for higher level responses. This relationship between working memory and age leads to the proposition that cognitive developmental stages can be explained by increasing attention span, or working memory capacity (Case, 1980b; Mclaughlin, 1963; Pascual-Leone, 1969). Thus, we assume that Piaget’s theory of development of human intelligence with age can be in accordance with the memory capacity criterion. For example, understanding video within 10 seconds can be understood to be from Stage 1 (Pre-operational). However, beyond minutes (e.g., understanding a scene within 3 minutes), is possible from Stage 2 (Early concrete). Beyond this, understanding two or more scenes (e.g., understanding sequences changing time and place) is possible from Stage 3 (Middle concrete). Finally, it is possible from Stage 4 (Concrete generalization) to know and understand whole video entirely.

Piaget’s developmental stages are also consistent with the logical complexity criterion. As the SMILE project proposes, from simple recall to assumption-based reasoning, methods have a kind of hierarchy that is closely related to a person’s stage of development. For example, level 1 and level 2 is available from Stage 1 (Pre-operational) in that it needs a simple call. Specifically, level 1 requires one supporting fact (e.g., \{jacket-is-black\}), on the other hand, level 2 requires independent multiple supporting facts. Level 3 is available from Stage 3 (Middle concrete), in that this level can be understood using dependent multiple supporting facts across time. Level 4 is available from Stage 4 (Concrete generalization), because this level requires a higher thought on causality in relation to “Why”. Finally, level 5 is available from Stage 5 (Formal Stage), as it requires creativity and abstract thinking about new ideas. As such, each phase of SMILE can be expressed as roughly equivalent to the human developmental
stage postulated by Piaget (Collis, 1972).

5.4 Multilevel Video QA Dataset

5.4.1 Overview

We collected multilevel Video QA dataset on a popular Korean drama Another Miss Oh, which has 18 episodes, 20.5 hours in total. The dataset consists of sequences of video frames (3 frames per second), character-centered video annotations, and QA pairs with hierarchical difficulty levels. Figure 5.1 illustrates the DramaQA dataset. We also present a comparison of our dataset to some recently proposed video QA datasets (Table 5.1). Among the datasets, only the DramaQA provides difficulty levels of the questions and rich information of characters including coreference resolved scripts.

Table 5.1: Comparison between video story QA datasets (MovieQA (Tapaswi et al., 2016), PororoQA (Kim et al., 2017b), TVQA (Lei et al., 2018) and TVQA+ (Lei et al., 2019)). Only DramaQA dataset provides hierarchical QAs and character based visual metadata. Average target video length for single QA is divided into shot level and scene level, since we provide multi level QA.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Q #</th>
<th>Images</th>
<th>Avg. Video len. (s)</th>
<th>Textual metadata</th>
<th>Visual metadata</th>
<th>Q. lev</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieQA</td>
<td>6,462</td>
<td>-</td>
<td>202.7</td>
<td>Plot, DVS, Subtitle</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PororoQA</td>
<td>8,913</td>
<td>-</td>
<td>1.4</td>
<td>Description, Subtitle</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TVQA</td>
<td>152,545</td>
<td>-</td>
<td>76.2</td>
<td>Subtitle</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TVQA+</td>
<td>29,383</td>
<td>148,468</td>
<td>61.49</td>
<td>Subtitle</td>
<td>Obj. Bbox</td>
<td></td>
</tr>
<tr>
<td>DramaQA</td>
<td>16,191</td>
<td>217,308</td>
<td>3.7^a 91.3^b</td>
<td>Script*</td>
<td>Char. Bbox</td>
<td>✓</td>
</tr>
</tbody>
</table>

^a Average video length for shot
^b Average video length for scene
* Coreference resolved script
Figure 5.1: An example of multilevel Video QA dataset which contains video clips, scripts, and QA pairs with levels of difficulty. A pair of QA corresponds to either a shot or a scene, and each QA is assigned one out of a possible four stages of difficulty. A video clip consists of a sequence of images with visual annotations centering the main characters.
Difficulty 1 and Difficulty 2 target the length of shot video. Difficulty 1 requires single supporting fact to answer and Difficulty 2 requires multiple supporting facts to answer. Questions for these two levels need spatial-level reasoning process. Difficulty 3 or Difficulty 4 requires a time factor to answer and target the length of scene video. Especially, Difficulty 4 requires causality between supporting facts from a different time. Questions for these two levels need temporal-level reasoning process.

5.4.2 Question-Answer Hierarchy for Levels of Difficulty

To construct question-answer pairs with hierarchical levels of understanding, we have proposed two criteria: Memory capacity and Logical complexity. Memory capacity (MC) is defined as the length of the video clip, and corresponds to working memory in human cognitive process. Logical complexity (LC) is defined by the number of logical reasoning steps required to answer the question, which is in line with Piaget’s developmental stage (Piaget, 1972).

For the Memory capacity, two types of video lengths are used: shot-level
(level 1) and scene-level (level 2). The shot-level clips are less than about 10 seconds long, shot from a single camera angle. The scene-level clips are about 1-10 minutes long without location changes, which contain sequences of actions.

For the Logical complexity, four levels of question types are used: simple recall on one cue (level 1), simple analysis on multiple cues (level 2), intermediate cognition on dependent multiple cues with time factor (level 3) and high-level reasoning for causality (level 4). Questions in level 1 can be answered using simple recall; requiring only one supporting fact. Questions in level 2 require multiple supporting facts, which trigger simple inference. Questions in level 3, require multiple supporting facts with time factor to answer. Question in level 4, cover reasoning for causality usually beginning with "Why".

From the two criteria, we define four hierarchical difficulties for QA and these difficulties are consistent with cognitive developmental stages of Piaget's theory (Piaget, 1972; Collis, 1975b), and align these four difficulties with spatiotemporal reasoning. Questions classified to level 1 in MC and LC belong to Difficulty 1 which is available from *Pre-Operational Stage* where a child thinks at a symbolic level, but is not yet using cognitive operations. Questions classified to level 1 in MC and level 2 in LC belong to Difficulty 2 which is also available from *Early Concrete Stage* where a child can utilize a relevant operation between multiple supporting facts. These two difficulty levels can be solved via spatial-level structured representations and reasoning.

Questions classified to level 2 in MC and level 3 in LC belong to Difficulty 3 which is available from *Middle Concrete Stage* where a child can think by utilizing more than two relevant cognitive operations and utilize dependent multiple supporting facts across time. Questions classified to level 2 in MC and level 4 in LC belong to Difficulty 4 which is available from *Concrete Generalization Stage* where a child can just generalize only from personal and concrete expe-
Figure 5.3: (a) The number of QA pairs per episode and difficulty level. Given that the scene length is tens of times than the size of the shot, the variation between levels is small compared to the number of videos. (b) The number of 5W1H question types per difficulty level.

We also analyze overall distributions of levels per episode and 5W1H question types per difficulty level. Figure 5.3(a) shows overall distributions of levels along with the episodes. Because a single scene of a video has multiple shots, the number of questions for Difficulty 1 and 2 is naturally larger than that for Difficulty 3 and 4. In Difficulty 1 and 2, **Who** and **What** questions are the majority. In case of Difficulty 3, **How** and **What** types are the top-2 questions. In Difficulty 4, Most of questions are start from **Why**.

experience and have a higher thought on causality in relation to “why”. These two difficulty levels can be solved via temporal-level structured representations and reasoning. Examples for each level are illustrated in Figure 5.2.
Figure 5.4: Examples of visual metadata and coreference resolved scripts. Our dataset provides visual metadata containing the main characters’ bounding box, name, behavior, and emotion for each frame.

5.4.3 Rich Video Annotations

As the characters are primary components of video stories, we also provide rich annotations for the main characters in “Another Miss Oh”. As visual metadata, all image frames in the video clips are annotated with main characters’ information. Also, to give a consistent view of the main characters, all coreference of the main characters is resolved in scripts of the video clips. Figure 5.4 shows the examples of visual metadata and coreference resolved scripts.

Visual Metadata

- **Bounding Box**: In each image frame, bounding boxes of both a face rectangle and a full-body rectangle for the main characters are annotated with their name. In total, 20 main characters are annotated with their unique name.

- **Behavior & Emotion**: Along with bounding boxes, behaviors, and emotions of the characters shown in the image frames are annotated. In-
Figure 5.5: (a) The percentage of each character’s frequency in visual metadata. *Haeyoung1* and *Dokkyung* is two main characters of drama *AnotherMissOh*. *Haeyoung2* is the person who has same name with *Haeyoung1*, but we divided their name with numbers to get rid of confusion. (b) The percentage of each behavior frequency in the visual metadata. *none* behavior occupies a lot because there are many frames with only character’s face. (c) The percentage of each emotion frequency in the visual metadata.

cluding *none* behavior, total 28 behavioral verbs, such as *drink*, *hold*, *cook*, is used for behavior expression. Also, we present characters’ emotion with 7 emotional adjectives; *anger*, *disgust*, *fear*, *happiness*, *sadness*, *surprise*, and *neutral*.

In Figure 5.5, distributions of main character and their behavior and emotion in visual metadata is visualized. As shown in Figure 5.5(a), Haeyoung1 and Dokkyung appear the most frequently among all characters. For Figure 5.5(b) and (c), note that various behaviors and emotions are represented except the situations when cannot express much information due to its own trait like *none* behavior or neutral emotion. Also, due to the nature of the TV drama, it is natural that the frequency of appearance varies depending on the importance of the characters: long-tail distribution.
Coreference Resolved Scripts

To understand video story, it is crucial to understand the dialogue between the characters. Notably, the information such as “Who is talking to whom about who did what?” is significant in order to understand whole stories. We provide this information by resolving the coreferences for main characters in scripts. As shown in Figure 5.4, we annotate the characters’ names to all personal pronouns for characters, such as I, you, we, him, etc. By doing so, characters in scripts can be matched with those in visual metadata and QAs.

We show analyses of each person’s utterances in the scripts (Figure 5.6). First of all, we analyze who was the most frequent person talking with the main character and mentioned in their dialogue. As shown in Figure 5.6(a), we can see that two protagonists of the drama, Dokyung and Haeyoung1, appeared most often in their dialogue. Also, it indirectly shows the relationship between the main characters. Hun, Dokyung’s brother, is familiar to Dokyung but a stranger to Haeyoung1. Figure 5.6(b) shows the percentage of each character’s utterance from whole episodes.

5.4.4 Data Collection

For the data collection process, we hired fluent English speakers without using a crowdsourcing service because of a copyright issue on the distribution of the drama video as well as using the characteristics of the drama annotation-task. Since annotating visual information requires not only knowledge of the characters but also the stories going on the drama, utilizing a crowdsourcing service with a considerable number of part-time workers might decrease the quality of the resulting annotation dataset. Therefore, with automated visual annotation tagging tools, all annotation tasks were carried out by a small group of dedicated workers who are aware of the whole drama story-line and the
Figure 5.6: (a) **Top**: Top-3 most frequently the person who the speaker talk to, for each top 6 most spoken person. **Bottom**: Top-3 most frequently the person who the speaker talks about, for each top 6 most spoken person. (b) The percentage of each person’s utterance in the script.

During the data collecting process, the intermediate annotation results were checked and feedback was given to continue the work. Hired workers didn’t start to make their own QAs not until they fully understood our hierarchical difficulties. After the annotation procedure, data inspection was proceeded to confirm their QAs were made correctly.

For visual metadata annotation, the visual bounding boxes were created using an automated tagging tool, and workers manually annotated the main characters’ names, behaviors, and emotions. We predefined main characters, emotions, and behaviors as follows:

- **Main character:** Anna, Chairman, Deogi, Dokyung, Gitae, Haeyoung1, Haeyoung2, Heeran, Hun, Jeongsuk, Jinsang, Jiya, Kyungsu, Sangseok, Seohee, Soontack, Sukyung, Sungjin, Taejin, Yijoon
• Emotion: anger, disgust, fear, happiness, sadness, surprise, neutral

• Behavior: drink, hold, point out, put arms around each other’s shoulder, clean, cook, cut, dance, destroy, eat, look for, high-five, hug, kiss, look at/back on, nod, open, call, play instruments, push away, shake hands, sing, sit down, smoke, stand up, walk, watch, wave hands, write

The QA pairs were created with the following rules: 1) Workers must use the main characters’ names (i.e. Haeyoung, Dokyung, Deogi, etc.) instead of pronouns (i.e. They, He, She, etc.). 2) Questions and answers should be complete sentences. 3) All sentences should be case-sensitive. For hierarchical difficulty of QAs, different rules were applied for each level:

• Difficulty 1
  
  – The Question-Answering (QA) set should be based on only one supporting fact (A triplet form of subject-relationship-object) from the video.
  
  – Question can (should) be started with Who, Where, and What.

• Difficulty 2
  
  – The Question-Answering (QA) set should be based on multiple supporting facts from the video.
  
  – Question can (should) be started with Who, Where, and What.

• Difficulty 3
  
  – The Question-Answering (QA) set should be based on multiple situations/actions with sequential information. To answer the question at this difficulty, temporal connection of multiple supporting facts should be considered, differently from Difficulty 2 questions.
Figure 5.7: The overall architecture of unified learning framework for Video QA. The spatial-level structure learning module and temporal-level structure learning module are integrated to answer the questions for video story.

- Question can (should) be started with *How* (recommended) and *What*.

- Difficulty 4

  - The Question-Answering (QA) set should be based on reasoning for causality.

  - Question can (should) be started with *Why*.

5.5 Experiments
5.5.1 Unified Learning Framework

In this section, we propose an unified learning framework to learn compositional structured representations for Video QA. To answer the questions given
video stories, the learning framework integrates spatial-level structured representation learning (in Chapter 3) and temporal-level structured representation learning (in Chapter 4). The proposed architecture is composed of four modules: Visual Scene Graph Generator, Context Matching Module, Spatial Structure Learning Module and Temporal Structure Learning Module. We firstly generate visual scene graphs for each image frame in the video clip, then use Context Matching Module to get QA-aware representations for each visual scene graphs. After that, Spatial Structure Learning Module learns single vector representations for each graph and answers the question if the question needs spatial-level reasoning. If the question needs also temporal-level reasoning, then Temporal Structure Learning Module learns single vector representations for whole video sequences. The overall architecture of proposed model is illustrated in Figure 5.7.

**Visual Scene Graph Generator:** As the human-annotated scene graphs are generally not available, we suggest a simple method to generate scene graphs for images. Following the works (Anderson et al., 2018), objects and their attributes in images are predicted based on the ResNet-101 features from the detected bounding boxes. For each pair of the detected objects, relationships are predicted based on the frequency prior in the Visual Genome dataset (Krishna et al., 2017b). For all bounding boxes of human in the image frames, we replace the human-referring words in those relationships with the previously identified character names and also add behaviors, emotions and scripts as nodes.

**Context Matching Module:** The context matching module converts input graphs to query-aware contexts by using the question and answers as a query (Seo et al., 2016). Context vectors for nodes in graphs are updated with attention mechanism given queries, which is based on the dot product between each query and its corresponding node feature vector.
Table 5.2: Quantitative result on the multilevel Video QA dataset. We divide test set by difficulty level and get performance of each set. The Overall column shows the average performance of each level. The QA-only indicates a simple MLP model that inputs questions only and Our\{-SSLM, TSLM\} indicates the proposed model without Spatial Structure Learning Module and Temporal Structure Learning Module respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Diff. 1</th>
<th>Diff. 2</th>
<th>Diff. 3</th>
<th>Diff. 4</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA-only</td>
<td>30.29</td>
<td>27.73</td>
<td>28.18</td>
<td>27.43</td>
<td>28.41</td>
</tr>
<tr>
<td>Our−TSLM</td>
<td>53.66</td>
<td>51.20</td>
<td>36.18</td>
<td>38.29</td>
<td>44.83</td>
</tr>
<tr>
<td>Our−SSLM</td>
<td>44.42</td>
<td>41.83</td>
<td>38.19</td>
<td>44.33</td>
<td>42.19</td>
</tr>
<tr>
<td>Our (SSLM+TSLM)</td>
<td>72.21</td>
<td>63.75</td>
<td>51.78</td>
<td>50.12</td>
<td>59.46</td>
</tr>
</tbody>
</table>

**Spatial Structure Learning Module (SSLM):** The SSLM learns representative feature vectors for scene graphs of each image frames. We adapt the Spectrally Similar Graph Pooling model introduced in Chapter 3, which learns the representations of graph by capturing compositional hierarchies of input graphs.

**Temporal Structure Learning Module (TSLM):** The TSLM learns a final feature vector for a video given sequences of image frames representations. We adapt the Cut-Based Graph Learning Networks model introduced in Chapter 4, which learns the representations of video by discovering the compositional structure of video.

### 5.5.2 Experimental Result

Table 5.2 shows the quantitative results of the proposed model on the multilevel Video QA dataset. The QA-only model concatenates the average of question’s word embeddings and the average of corresponding candidate answer’s word embeddings and gets scores with simple MLP networks. The performance of
QA-only is slightly better than random selection¹ and it shows the dependencies between question and answer so that the answers cannot be reasoned easily without considering the video contents. Our\{-SSLM,TSLM\} shows the inference results of our model without spatial structure learning module and temporal structure learning module respectively. Note that spatial structure learning module is helpful to infer the answers for questions in difficulty 1 and 2. Also, the temporal structure learning model is essential for reasoning the answers in difficulty 3 and 4. Finally, integrating two modules (Our (SSLM+TSLM)) boosts the performance for all difficulty levels.

5.6 Conclusions

To develop video story understanding intelligence, we proposed Multilevel Video QA dataset with hierarchical difficulty levels. Also, coreference resolved script and rich visual metadata for character-centered video are provided. We suggested the dataset for spatiotemporal structure learning and argued different inductive biases, spatial-level reasoning and temporal-level reasoning, are needed to solve this problem.

¹Because the dataset contains five candidate answers and we control the indices of true answers, the random selection performs around 20% accuracy.
Chapter 6

Concluding Remarks

6.1 Summary of Methods & Contributions

In this dissertation, we studied to learn compositional and hierarchical structure from data. We divided the problem into two levels, spatial-level and temporal-level, and suggest methods to learn structured representations in each level. As Graph Neural Networks (GNNs) are good at structured representations but lack of property for compositional hierarchy, we propose two compositional GNNs architectures to learn this property.

First, for the spatial-level structured representation learning, we proposed Spectrally Similar Graph Pooling (SSGPool) algorithm, which can be adapted GNN architectures and build the compositional hierarchy in an end-to-end fashion. Although existing graph pooling approaches take either feature-based pooling or structure-preserving pooling, our proposed method considers both properties simultaneously by using spectral characteristics of the graph. As results, the SSGPool achieved the best result for image-retrieval task with scene-graph
and showed promising properties in graph coarsening. Also, with various graph benchmark datasets, the SSGPool showed consistent advantages compared to strong baselines.

Second, for the temporal-level structured representation learning, we proposed Cut-Based Graph Learning Networks (CB-GLNs) which learn structured representations of video by discovering the compositional structure along with temporal axis in multilevel graph forms. To learn compositional hierarchies of video, we integrated graph-cut mechanism with graph neural networks by applying temporal constraint for graph-cut procedure. With CB-GLNs, a set of semantic units is found and merged to construct hierarchies, and final representations are learned based on this hierarchical structure. With the video theme classification and the video question answering task, the CB-GLNs showed promising results compared to previous approaches.

Finally, we proposed a multilevel video question answering (Video QA) dataset to learn spatiotemporal structured representations. For multilevel QA, two criteria are proposed based on the cognitive developmental stages: memory capacity and logical complexity. Based on the criteria, we constructed the hierarchical QA and aligned them with spatial-level and temporal-level.

6.2 Discussions for Future Directions

Although we introduced promising methods to learn structured representations based on the Graph Neural Networks (GNNs), there are a number of unanswered questions regarding the best ways to learn truly human-like structured representations.

One of the unsolved question is the “where do the graphs come from that the models operate over?”. Even though success of deep learning has been coming from its ability to perform complex computations over raw sensory data,
yet it is unclear the best ways to convert sensory data into more structured representations like graphs. As introduced in Section 2.2, visual scene graph is one of the favorable approach for structured representations of images. However, current researches for generating scene graphs are mostly performed in a supervised way with ground truth scene graphs. To take it a step further, we need address the problem of inferring an explicit structure from the raw sensory data in an unsupervised way. One possible approach is to assume a fully connected graph structure, then sparsifying and coarsening the graph in appropriate ways. However, this representations may not correspond exactly to the true entities and there is no single method which can reliably extract discrete entities from sensory data. Developing such a method is an exciting challenge for future research, and once solved will likely open the door for much more powerful and flexible algorithms.

The other open question is that "How to integrate spatial and temporal structured information in a single form?". As noted above, image-based structured representations like scene graphs have been improved across multiple image tasks. Yet, extending it in a temporal way, such as temporal events, has not been explored much, which lead to be one step closer to human cognition. One possible approach is to define the spatiotemporal scene graphs as recently proposed work (Ji et al., 2020). With the evidence from Cognitive Science and Neuroscience that people actively encode activities into consistent hierarchical part structures, (Ji et al., 2020) introduces Action Genome dataset which has a representation that decomposes actions into spatio-temporal scene graphs. The other feasible approach is to model dynamical transitions over time of spatial scene graphs. It could be done by adaptively modifying graph structures during the time. (Kipf et al., 2018) gives an attractive insight by suggesting model inferring an explicit interaction structure while simultaneously learning
the dynamical model of the interacting system.

The last question for discussion is the "How to model the reasoning process given the structured representations of data?" As the ability to assimilate structured knowledge and use it to draw reasoning is one of the key characteristics of human cognition, it is worthwhile to discuss the modeling reasoning process given structured representations. One of the possible approach is to view reasoning process as programs. As the reasoning process could be expressed by notions like recursion, control flow, and conditional iteration, the programs can be good options which offer representational and computational expressivity with respect to these notions (Parisotto et al., 2016). There are other works seen as similar approaches such as partial tree traversals in a state-action graph (Guez et al., 2018; Farquhar et al., 2018), hierarchical action policies (Andreas et al., 2017). The reasoning process also can be expressed by finding answers to questions given structured representations. Recent works for visual question answering (Santoro et al., 2017; Hudson and Manning, 2019a,c) gives great insight for reasoning the answer given structured representations of images.
Bibliography


Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D.,


tection with deep structural ranking. In Thirty-Second AAAI Conference on Artificial Intelligence.


Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G.


초록

인간의 인지 능력은 실세계를 다양한 요소들과 그들 사이의 관계로 표현하고, 요소들 사이의 구성적 계층을 구축하는 구조적 표현에 기인한다. 비록 기계학습, 특히 심층 신경망 모델의 발전이 컴퓨터 비전, 자연 언어 처리 등 다양한 분야에서 인간수준에 근접한 성능을 이끌었지만, 아직 인간의 이러한 구조적 표현 능력을 모사하는 것은 도전적인 문제로 남아있다. 최근 구조적 표현 학습을 위해, 그래프의 형태로 구조화된 데이터를 학습하는 그래프 신경망(Graph Neural Networks, GNNs)이 제안되었다. 하지만, 일반적인 GNNs는 메세지 전달(message passing) 방식의 flat한 연산을 이용하여 그래프 내 요소(정점) 표현 학습에 중점을 두기 때문에 데이터의 전체적인 구성적 계층을 학습하는 것은 어렵다.

본 학위논문에서는, 데이터로부터 구성적이고 계층적인 구조를 학습하는 것을 근본적인 문제로 상정하고, 이러한 구성적 계층구조를 학습하는 방법론에 대해 연구한다. 이를 위해, 본 문제를 공간-수준과 시간-수준 두 단계로 나누어, 공간적 데이터와 시간적 데이터 각각에 대해 구성적 구조 표현 학습을 위한 두가지 구성적 그래프 신경망을 제안한다. 더 나아가, 공간-수준, 시간-수준의 구성적 구조 표현 학습을 통합하기 위한 더층수준 비디오 질의응답 데이터셋을 제안한다.

첫번째로, 공간-수준에서의 구성적 구조 표현 학습을 위해 새로운 그래프 플링 알고리즘인 Spectrally Similar Graph Pooling (SSGPooll)을 제안한다. 제안하는 SSGPool 알고리즘은 원본 그래프의 노드에서 축소된 그래프의 노드 사이의 매핑을 나타내는 축소 행렬(Coarsening matrix)를 학습하는 것을 핵심으로 한다. 이때, 축소 행렬은 노드의 특징벡터를 기반으로 학습하면서 동시에 원본 그래프와 축소된 그래프 사이의 스펙트럼적 특징을 유지한다. SSGPool의 효과를 검증하기 위해, 다양한 그래프 벤치마크들에 대해 다른 강력한 기준모델들과 비교실험을 수행한다. 또한, 실세계 문제인 scene graph 기반의 이미지 검색 문제에 적용하여
제안하는 방법을 검증한다.

다음으로, 시간-수준에서의 구성적 구조 표현 학습을 위해, Cut-Based Graph Learning Networks (CB-GLNs)을 제안한다. CB-GLNs는 비디오 내 다양한 길이의 의미적 흐름과 그들의 구성을 내포하는 복잡한 의존적 구조를 발견함으로써 비디오의 표현을 학습한다. 이를 위해, 그래프의 정점과 간선이 각각 비디오 내 이미지 프레임들과 그 사이의 의존성을 표현하는 하나의 그래프 구조로 비디오를 나타낸다. CB-GLNs은 그래프 컷과 함께 매개변수화된 커널(parameterized kernel)과 메세지 전달 프레임워크를 이용하여 데이터의 구성적 계층 구조를 다중레벨 그래프 형태로 찾는다. 성능평가를 위해 대표적인 두가지 비디오 이해 문제인 비디오 주제 분류 및 비디오 질의응답에 대해 실험을 수행한다.

마지막으로 공간과 시간의 구조 학습을 통합하기 위해 다중레벨 비디오 질의응답 데이터셋을 제안한다. 다중레벨 질의응답을 위해, 두가지 기준인 메모리 용량(memory capacity)과 논리적 복잡도(logical complexity)를 정의하고 이를 기반으로 질문의 계층적 난이도를 제안한다. 계층적 난이도는 앞서 제안된 시공간-수준과 정렬된다. 데이터셋은 TV 드라마 “또 오해영”을 기반으로 구축하며 다중레벨 난이도 기반 질의응답 및 다양한 길이의 비디오 클립을 포함한다. 통합된 시공간-수준의 구성적 구조 학습의 평가를 위해, 앞서 제안한 두가지 구성적 그래프 신경망을 결합하여 다중레벨 비디오 질의응답 실험을 수행한다.

주요어: Compositional structure learning, Deep neural networks, Graph neural networks, Video representation learning, Video Question Answering
학번: 2014-21783