Graph Representation Learning of and Its Applications

Zhang Xinyi

School of Electrical and Electronic Engineering,

Nanyang Technological University

- Graphs can be constructed from a large range of source.
  - Chemical structure, citation network, knowledge graphs…
- Two research direction.
  - Graph based graph analysis (finding graph embedding)
    - graph classification, graph clustering…
  - Node based graph analysis (finding node/edge embedding)
    - node classification, link prediction, node clustering…

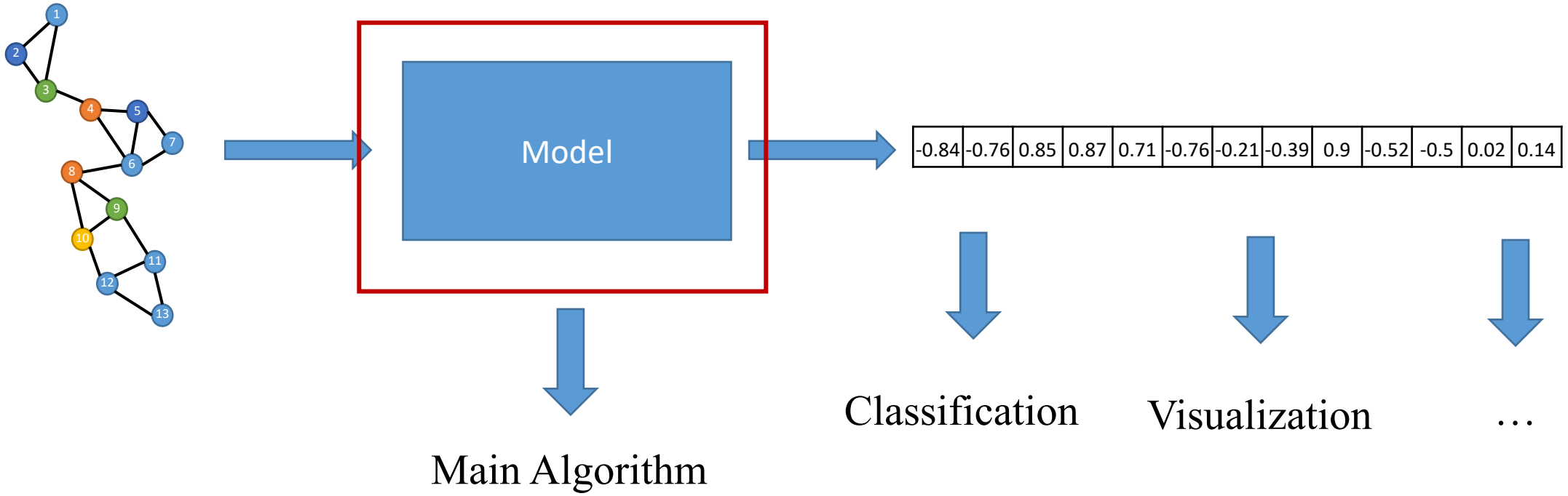
- Previous Work
  - CapsGNN (graph based)
  - Subgraph2vec# (graph based)
- Current Work
  - ➢ Metapath Analysis (node based)

Previous Work

- CapsGNN
  - A model for graph based graph representation learning.
  - Framework:
    - Basic node features extraction: GCN is applied to extract node features.
    - Graph features extraction: Dynamic Routing [1] is applied as a pooling method to compress node representations into multiple graph representations.
    - Class features extraction: Dynamic Routing is applied again to do graph classification and generate final class representations.
  - Code: https://github.com/XinyiZ001/CapsGNN

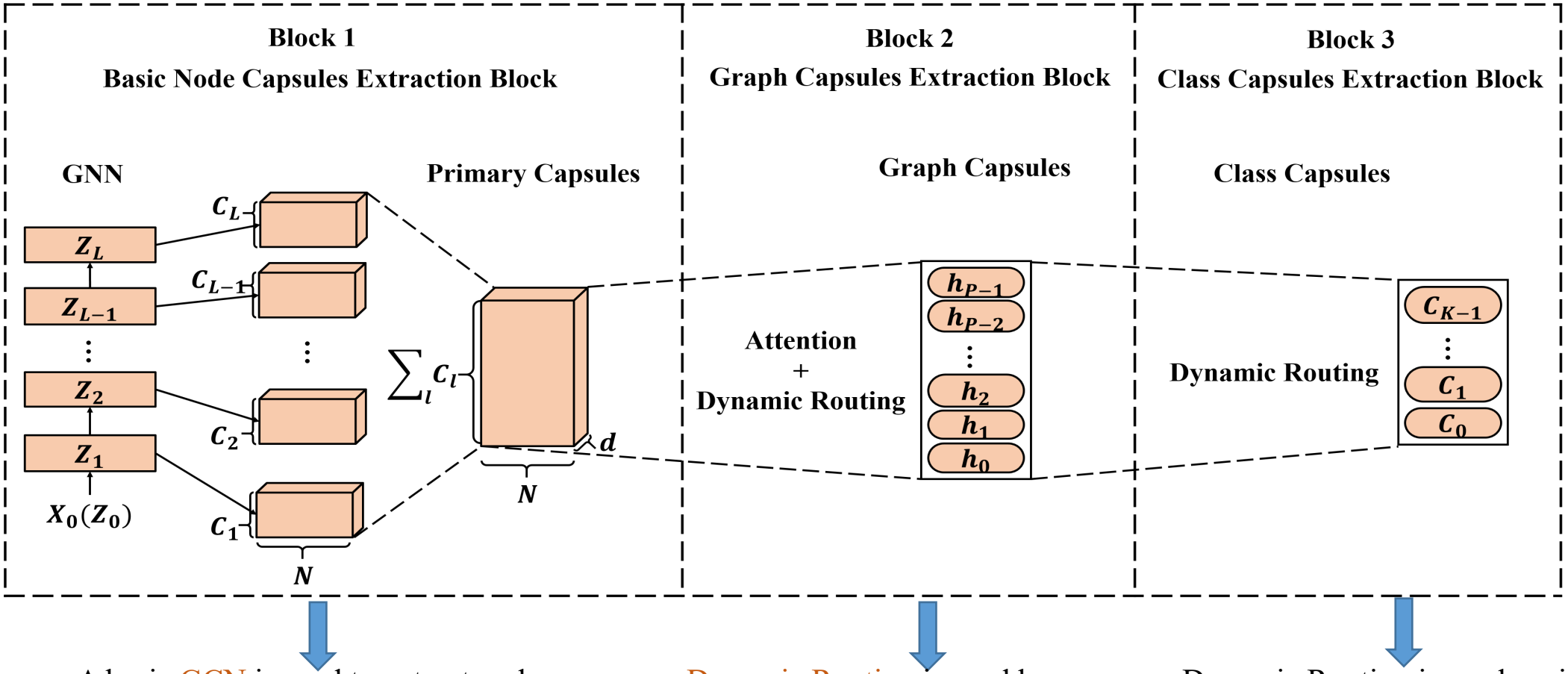
1. S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in Advances in Neural Information Processing Systems, pp. 3856-3866, 2017.

Graph Based Representation Learning



Graph based representation learning method is designed to find the suitable embedding for the whole graph. Then, graph classification, graph clustering can be done based on the graph embeddings.

Framework of Capsule Graph Neural Network

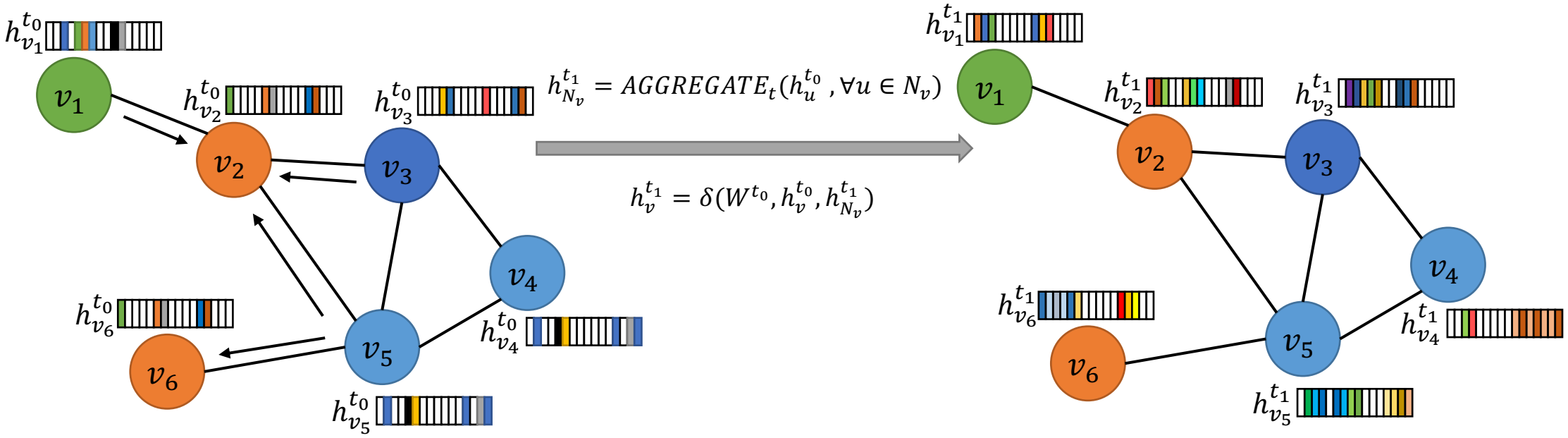


A basic GCN is used to extract node features which are then reshaped into the form of capsules (multiple vectors).

Dynamic Routing is used here as a pooling method to generate multiple graph embeddings.

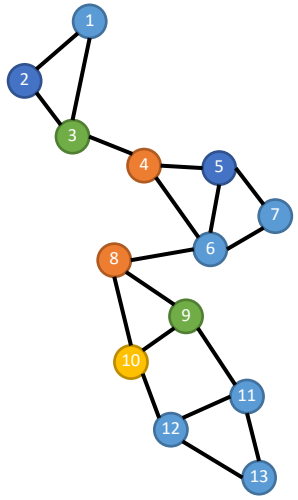
Dynamic Routing is used again to do graph classification and generate the final graph embeddings.

Basic Idea of Graph Neural Network



The basic idea of most GNN is to generate the representations of the center node by gathering the information of its neighboring nodes. The aggregation function and activation function will be used during this procedure.

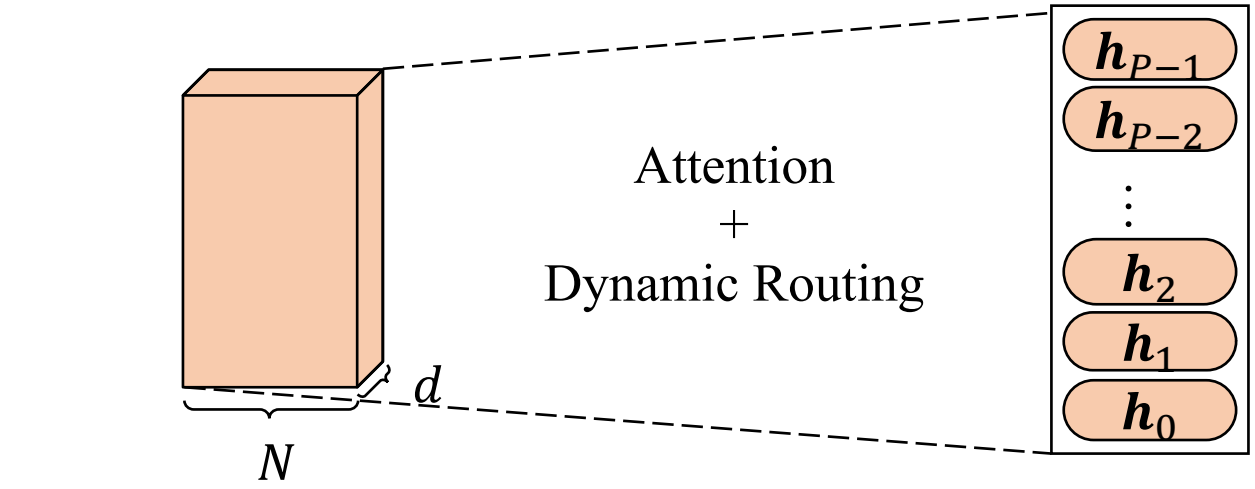
The Result of GNN



1	0.64	-0.15	-0.63	-0.47	0.64
2	0.31	0.84	-0.87	0.56	0.41
3	0.24	-0.87	-0.29	-0.32	-0.67
4	0.09	0	-0.07	0.02	-0.89
5	0.91	-0.99	-1	0.17	0.67
6	0.23	-0.49	0.56	0.48	0.82
7	0.97	-0.12	0.11	0.66	0.02
8	-0.98	0.18	1	-0.21	0.03
9	-0.07	-0.57	0.71	-0.26	0.23
10	0	0.33	-0.83	0.38	0.12
11	-0.15	0.33	0.96	-0.17	-0.91
12	-0.63	-0.37	0.43	-0.72	-0.62
13	0.64	-0.15	-0.63	-0.47	0.64

By applying GCN, graph features are extracted in the form of node features.

Result of Dynamic Routing



Then Dynamic Routing is applied to compress node representations into a fixed number of graph representations. At the last layer, the Dynamic Routing is applied again to generate class representation and do classification.

Capsule Graph Neural Network

- Current work related to CapsGNN
  - Propose more optimization methods to improve the classification performance as well as efficiency.
  - Conduct more experiments to evaluate CapsGNN and target to a journal paper.

Previous Work

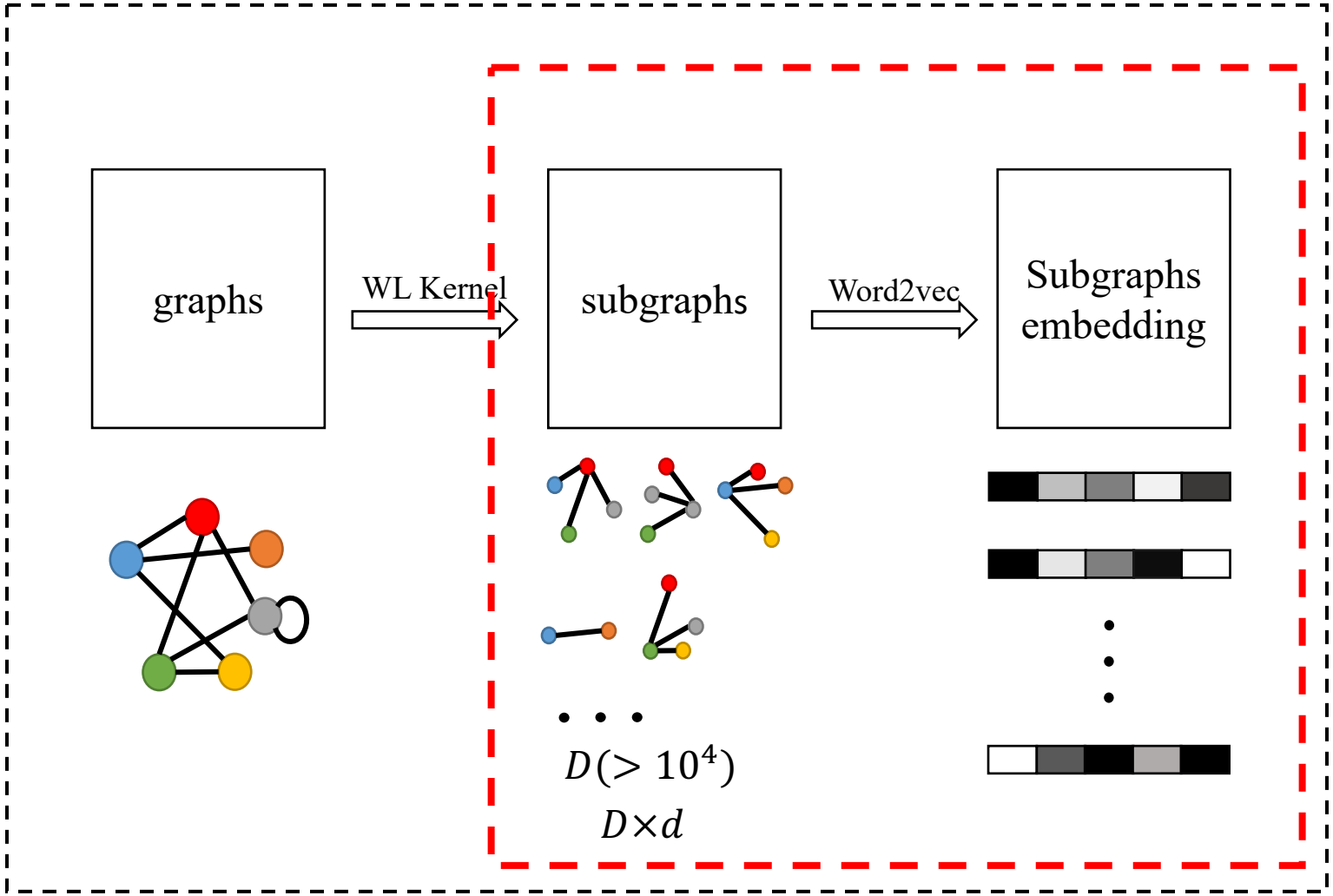
- Subgraph2vec#
  - Subgraph2vec# is a improved version of subgraph2vec which is my senior's work.
  - Framework
    - Subgraphs are extracted based on WL kernel method.
    - Word2vec is applied to learn the subgraph embeddings.
    - Graph kernel is built based on the learned subgraph embeddings.

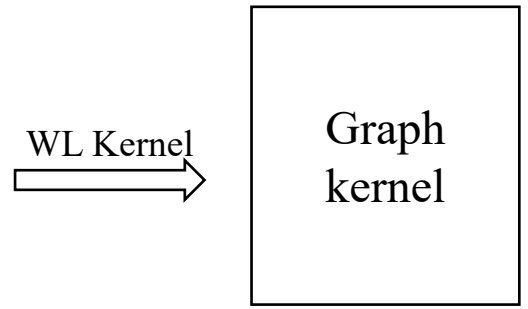
    ⇨ Basic Subgraph2vec
    - Hash technic is applied during word2vec. (Hash technic here is used to reduce the number of trainable parameters.)

Framework of Subgraph2vec

D: number of subgraphs which is also the length of subgraphs vocabulary.

d: dimension of subgraph embedding.





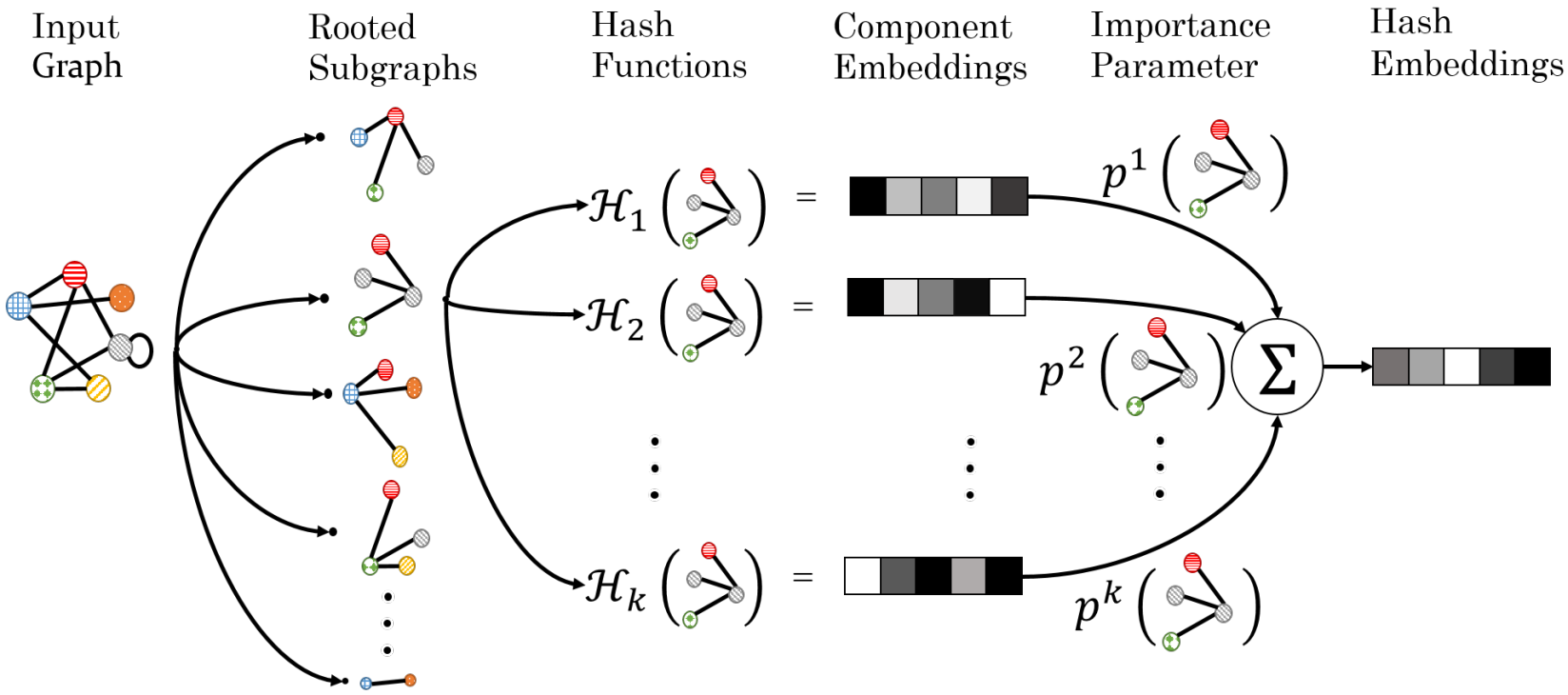
Number of trainable parameters:

$D \times d \times 2$ $(> 10^4 \times 32 \times 2)$

Subgraph2vec with Hash

D': length of subgraphs vocabulary.

Number of trainable parameters:
$D' \times d \times 2$



$D(> 10^4)$ $\qquad$ $D' = \frac{D}{B}$

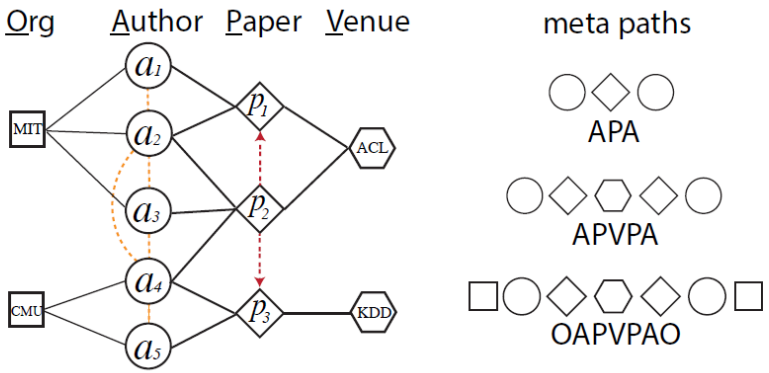
k hash functions are used to index k component embeddings for an input subgraph. Besides, k importance parameters are learned to scale each component embedding. The final subgraph embedding is calculated by adding all component embeddings.

Current Work

- Metapath Analysis
  - DeepWalk [1] (a basic way to learn node embeddings):
    - In DeepWalk, different walks will be extracted from a information network firstly.
    - Each walk can be considered as a sentence.
    - Word2vec can be applied to these extracted sentences/walks and then we can get node embeddings.
  - Metapaths
    - In heterogeneous information network (HIN), metapaths can be used to direct the walk extraction so that different types of nodes can be selected in a proper order.
    - Different metapaths can reflect different aspects of an HIN, so, I want to analyze the relationships between metapaths so that I can analyze different aspects of the HIN.

1. Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.

Metapath



(a) An academic network

Metapaths are pre-defined based on knowledge. For example, in an academic network, the meaningful metapaths can be :

- Author-Paper-Author
- Author-Paper-Venue-Paper-Author
- Organization-Author-Paper-Venue-Paper-Author-Organization

Node embeddings learned based on different metapaths can reflect different properties.