Summer ML Triggers

Week 2 Report

Russell Zhang
dg22882@bristol.ac.uk

Supervisor Dr Sudarshan Paramesvaran

Particle Physics Laboratory

Tables

1 Graph Neural Networks
- Mathematical Foundations
- Introduction
- Graph Neural Networks

2 Researching...

Part 1

Graph Neural Networks

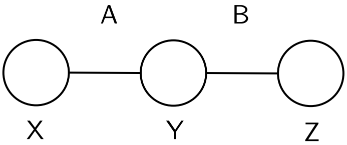
1.1 Mathematical Foundations

Graph Theory

Graph

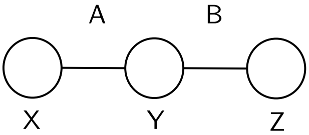
Graph

A graph is a set composed of vertices (or nodes) and edges. The nodes in the graph are connected (or not) by edges, which are represented by mathematical symbols as $G = (V, E)$, where V is all the points on the graph, and E is all the edges in the graph.

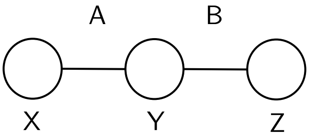


X, Y and Z (dots) are nodes, A and B (lines) are edges, nodes X and Y are adjacent, nodes Y and Z are adjacent, and nodes X and Z are not adjacent.

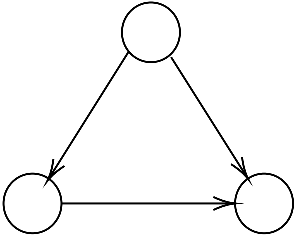
Graph types



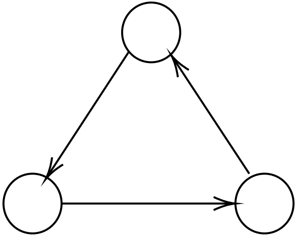
Undirected Graph



Directed Graph



DAG, Directed Acyclic Graph



DCG, Directed Cycle Graph

Algebra representations of graphs

- **Adjacency matrix:** for a simple graph $G = (V, E)$ with n-vertices, it can be described by an adjacency matrix $A \in R^{n\times n}$,where

$$A_{ij} = \begin{cases} 1 & if\,(v_i, v_j) \in E \; and \; i \neq j \\ 0 & otherwise \end{cases} \qquad (1)$$

It is obvious that such matrix is a symmetric matrix when G is an undirected graph.

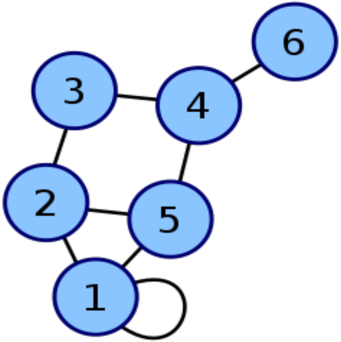
- **Degree matrix:** for a graph $G = (V, E)$ with n-vertices, its degree matrix $D \in R^{n\times n}$ is a diagonal matrix, where

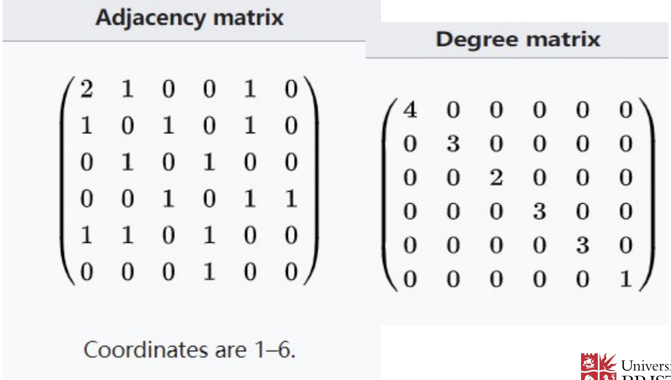
$$D_{ii} = d(v_i)$$

$d(v_i)$: the degree of a vertex counts the number of times an edge terminates at that vertex. Note that in the case of undirected graphs, an edge that starts and ends in the same node increases the corresponding degree value by 2 (i.e. it is counted twice).

Algebra representations of graphs

Example





Adjacency matrix

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Coordinates are 1–6.

Degree matrix

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Algebra representations of graphs

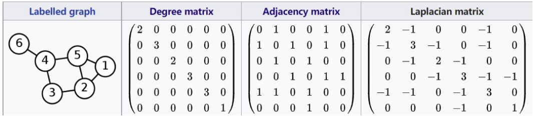
- **Laplacian matrix:** for a simple graph $G = (V, E)$ with n-vertices, if we consider all edges in G to be undirected, then its Laplacian matrix $L \in R^{n \times n}$ can be defined as

$$L = D - A \tag{2}$$

Thus, we have the elements:

$$L_{ij} = \begin{cases} d(v_i) & \textit{if } i = j \\ -1 & \textit{if } (v_i, v_j) \in E \textit{ and } i \neq j \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

Note that the graph is considered to be an undirected graph for the adjacency matrix.



Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

Algebra representations of graphs (Optional)

- **Symmetric normalized Laplacian:** the symmetric normalized Laplacian is defined as:

$$L^{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{4}$$

The elements are given by:

$$L_{ij}^{sym} = \begin{cases} 1 & if\ i = j\ and\ d(v_i) \neq 0 \\ -\frac{1}{\sqrt{d(v_i)d(v_j)}} & if\ (v_i, v_j) \in E\ and\ i \neq j \\ 0 & otherwise \end{cases} \tag{5}$$

Algebra representations of graphs (Optional)

- **Random Walk normalized Laplacian:** it is defined as:

$$L^{rw} = D^{-1}L = I - D^{-1}A \tag{6}$$

The elements can be computed by:

$$L_{ij}^{sym} = \begin{cases} 1 & if\ i = j\ and\ d(v_i) \neq 0 \\ -\frac{1}{\sqrt{d(v_i)d(v_j)}} & if\ (v_i, v_j) \in E\ and\ i \neq j \\ 0 & otherwise \end{cases} \tag{7}$$

1.2 Introduction

A Brief Summury

Reference

- Sanchez-Lengeling, et al., ”A Gentle Introduction to Graph Neural Networks”, Distill, 2021.

- 1.2.1 Graph Types Data representations
- 1.2.2 Problems of Graph Structured Data
- 1.2.3 Tasks of Using Graphs in Machine Learning

1.2.1 Graph Types Data representations

- **Image as Graph**

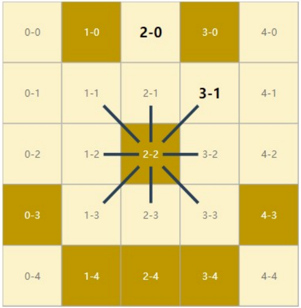
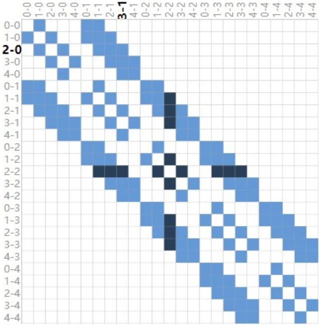
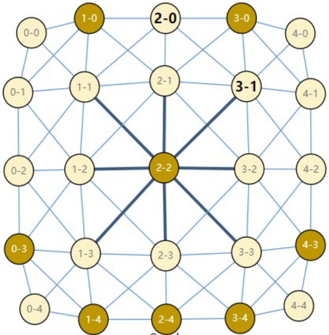


Image Pixels



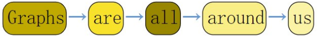
Adjacency Matrix

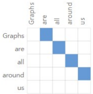


Graph

1.2.1 Graph Types Data representations

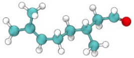
- **Text as Graph**

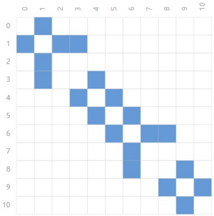




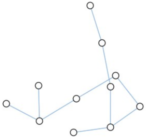
	Graphs	are	all	around	us
Graphs		■			
are			■		
all				■	
around					■
us					

- **Molecular as Graph**



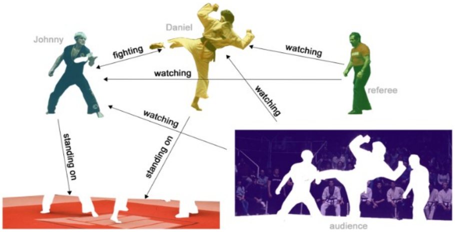


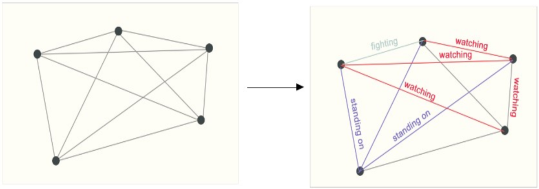
	0	1	2	3	4	5	6	7	8	9	10
0		■									
1	■		■	■							
2		■									
3		■			■						
4				■		■					
5					■		■				
6						■		■	■		
7							■				
8							■			■	
9									■		■
10										■	



1.2.2 Problems of Graph Structured Data

- Graph-level task: predict the property of an entire graph.
- Node-level tasks: predict the identity or role of each node within a graph.
- Edge-level task: predict the property or function represented by the edge.





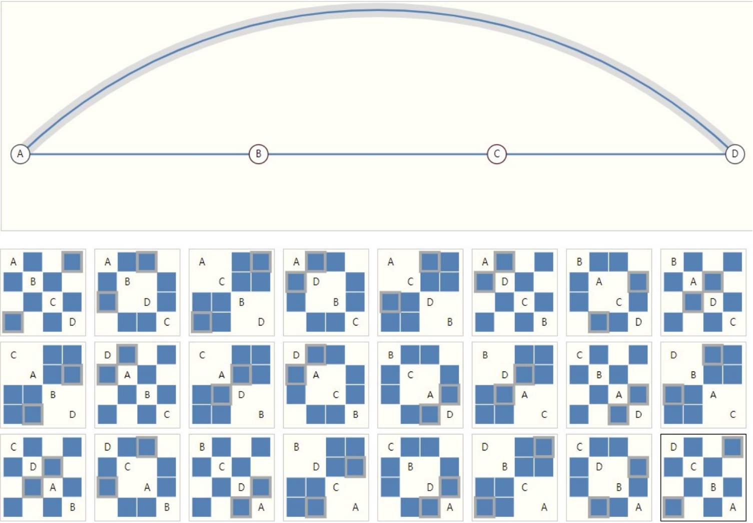
Input: fully connected graph, unlabeled edges

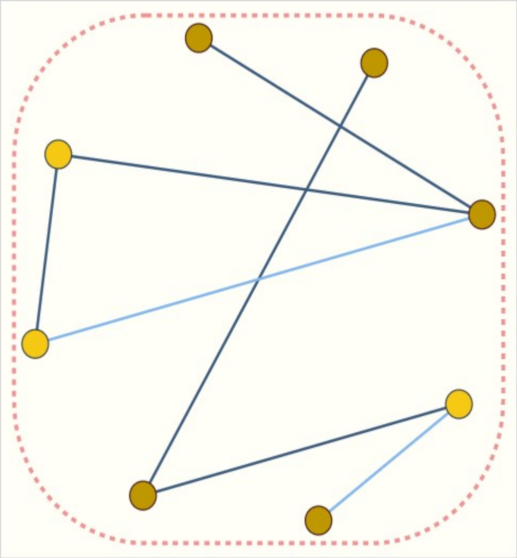
Output: labels for edges

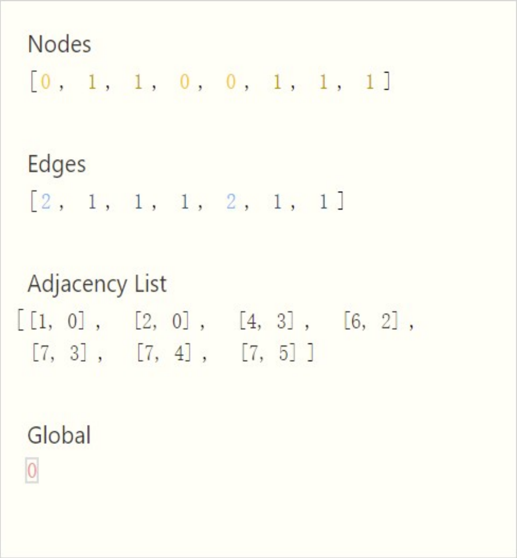
1.2.3 Tasks of Using Graphs in Machine Learning

How we will represent graphs to be compatible with neural networks

- Machine learning models typically take rectangular or grid-like arrays as input. Graphs have up to four types of information that we will potentially want to use to make predictions: nodes, edges, global-context and connectivity. So first three (nodes, edges, global-context) are relatively straightforward.
- Representing connectivity of Graphs is more complicated. Perhaps the most obvious choice would be to use an adjacency matrix, since this is easily tensorisable. But if a node needs to travel a long way to connect to another node, then our adjacency matrix will be sparse, which are space-inefficient.
- Another problem is that there are many adjacency matrices that can encode the same connectivity, and there is no guarantee that these different matrices would produce the same result in a deep neural network (that is to say, they are not permutation invariant).


A
B
C
D





Nodes

[0, 1, 1, 0, 0, 1, 1, 1]

Edges

[2, 1, 1, 1, 2, 1, 1]

Adjacency List

[[1, 0], [2, 0], [4, 3], [6, 2],
[7, 3], [7, 4], [7, 5]]

Global

0

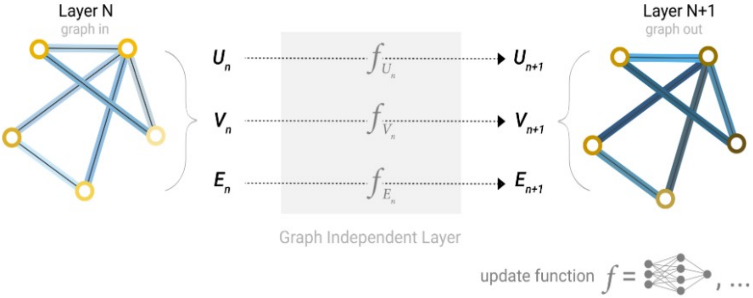
1.3 Graph Neural Networks

From Scratch

- 1.3.1 The simplest GNN
- 1.3.2 GNN Predictions by Pooling Information
- 1.3.3 Passing messages between parts of the graph
- 1.3.4 Different Combination Update Methods

1.3.1 The simplest GNN

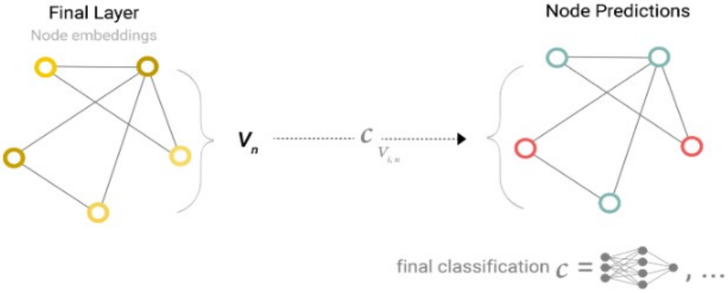
This GNN uses a separate multilayer perceptron (MLP) on each component of a graph; we call this a GNN layer. For each node vector, we apply the MLP and get back a learned node-vector. We do the same for each edge, learning a per-edge embedding, and also for the global-context vector, learning a single embedding for the entire graph. (This process does not change the structure of Graph.)



1.3.2 GNN Predictions by Pooling Information

Binary classification

We will consider the case of binary classification, but this framework can easily be extended to the multi-class or regression case. If the task is to make binary predictions on nodes, and the graph already contains node information, the approach is straightforward — for each node embedding, apply a linear classifier.

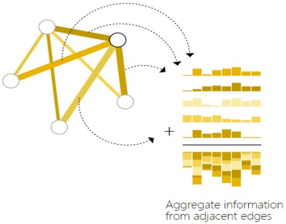


1.3.2 GNN Predictions by Pooling Information

Information Incomplete For instance, you might have information in the graph stored in edges, but no information in nodes, but still need to make predictions on nodes. We still can do predictions by pooling. Pooling in two steps:

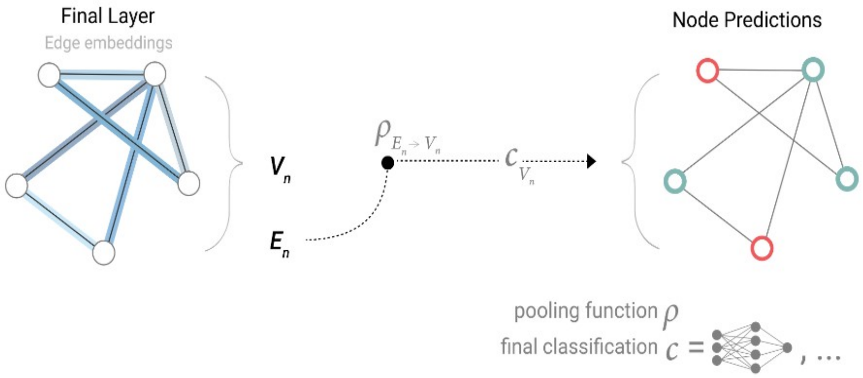
- For each item to be pooled, gather each of their embeddings and concatenate them into a matrix.
- The gathered embeddings are then aggregated, usually via a sum operation.

We represent the pooling operation by the letter ρ, and denote that we are gathering information from edges to nodes as $\rho_{E_n \to \rho_{V_n}}$.



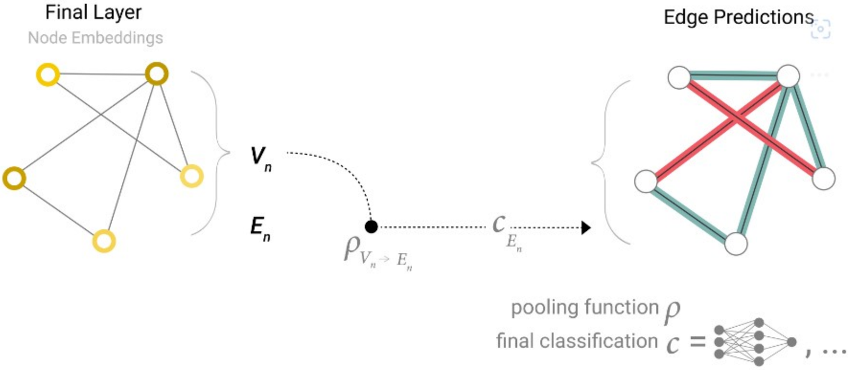
1.3.2 GNN Predictions by Pooling Information

Information Incomplete Nodes Binary classification So If we only have edge-level features, and are trying to predict binary node information, we can use pooling to route (or pass) information to where it needs to go. The model looks like this.



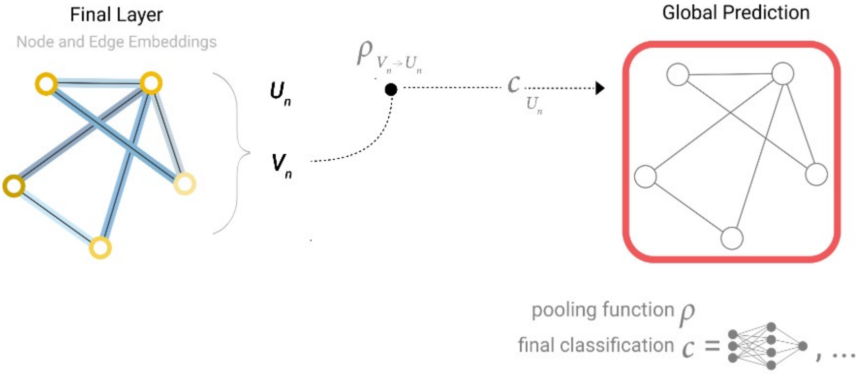
1.3.2 GNN Predictions by Pooling Information

Information Incomplete Edges Binary classification If we only have node-level features, and are trying to predict binary edge-level information, the model looks like this.



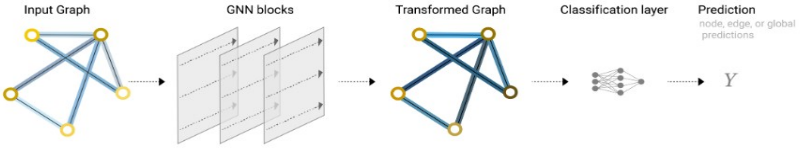
1.3.2 GNN Predictions by Pooling Information

Information Incomplete Global property Binary classification If we only have node-level features, and need to predict a binary global property, we need to gather all available node information together and aggregate them. This is similar to Global Average Pooling layers in CNNs. The same can be done for edges.



Intuition

In our examples, the classification model C can easily be replaced with any differentiable model, or adapted to multi-class classification using a generalized linear model.



An end-to-end prediction task with a GNN model.

Note that in this simplest GNN formulation, we′ re not using the connectivity of the graph at all inside the GNN layer. Each node is processed independently, as is each edge, as well as the global context. We only use connectivity when pooling information for prediction.

1.3.3 Passing messages between parts of the graph

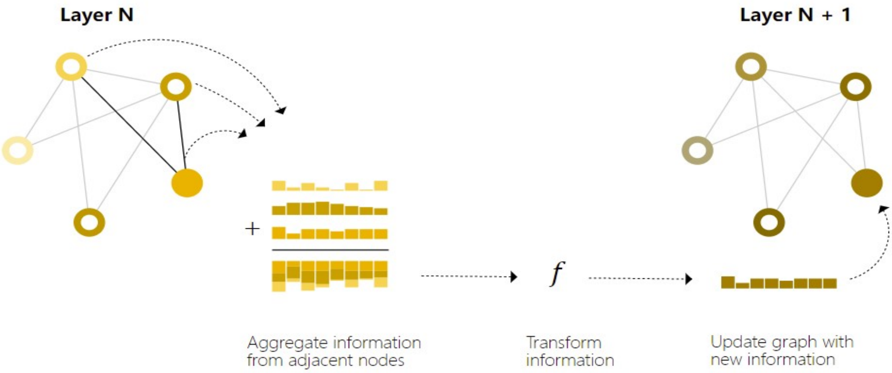
We could make more complex predictions by using pooling within the GNN layer, in order to make our learned embeddings aware of graph connectivity. We can do this using message passing, where neighboring nodes or edges exchange information and influence each other’ s updated embeddings.

Three Steps:

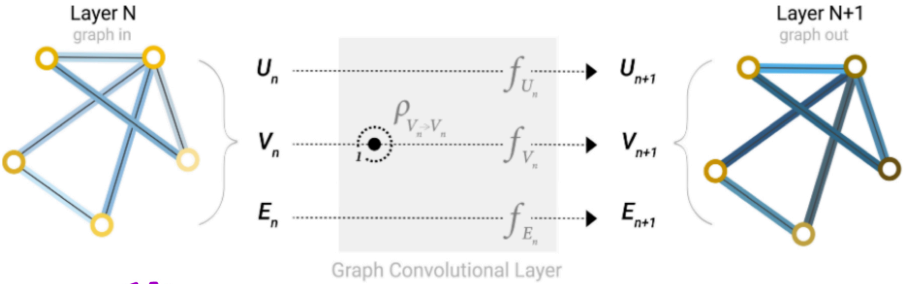
- For each node in the graph, gather all the neighboring node embeddings (or messages), which is represented as g function.
- Aggregate all messages via an aggregate function (like sum, but also can use mean or median).
- All pooled messages are passed through an update function, usually a learned neural network.

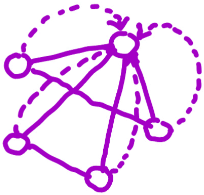
1.3.3 Passing messages between parts of the graph

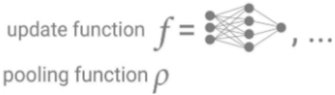
This process is really like convolution in CNN.


Layer N
Layer N + 1
+
f
Aggregate information from adjacent nodes
Transform information
Update graph with new information

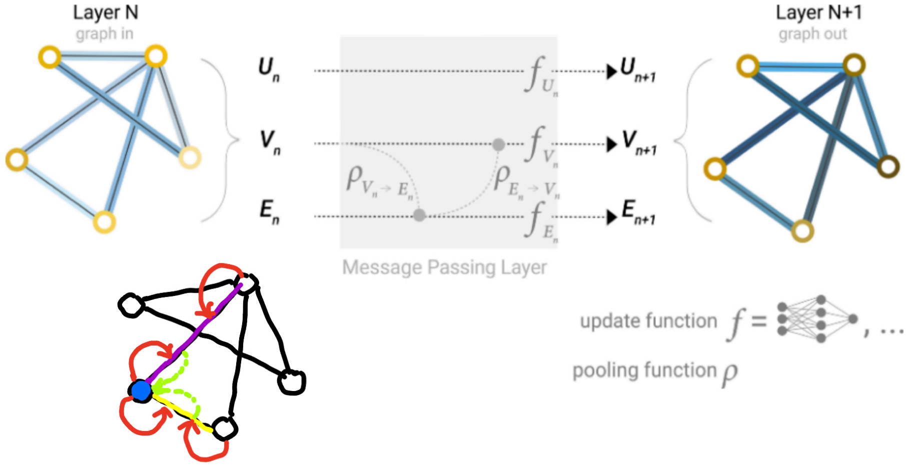
1.3.4 Different Combination Update Methods



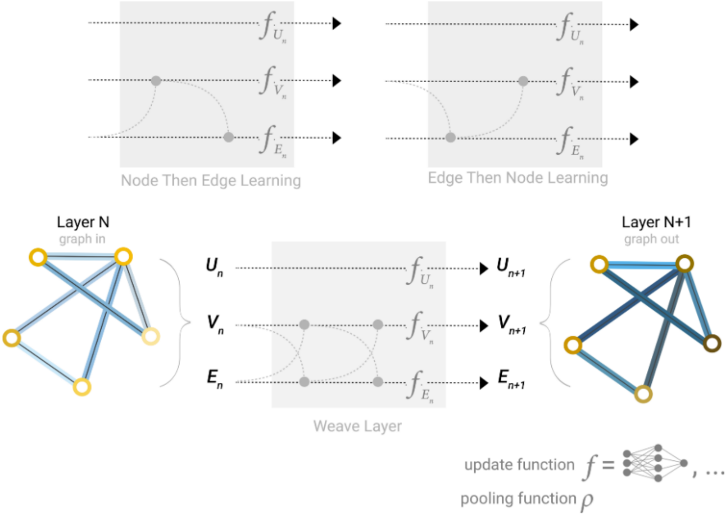




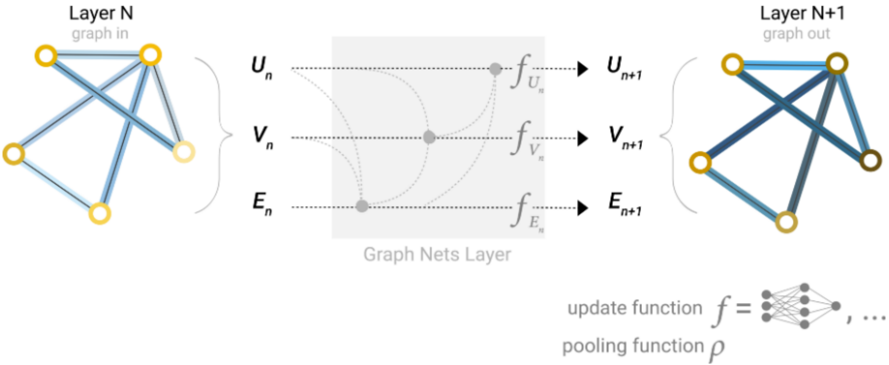
1.3.4 Different Combination Update Methods



1.3.4 Different Combination Update Methods

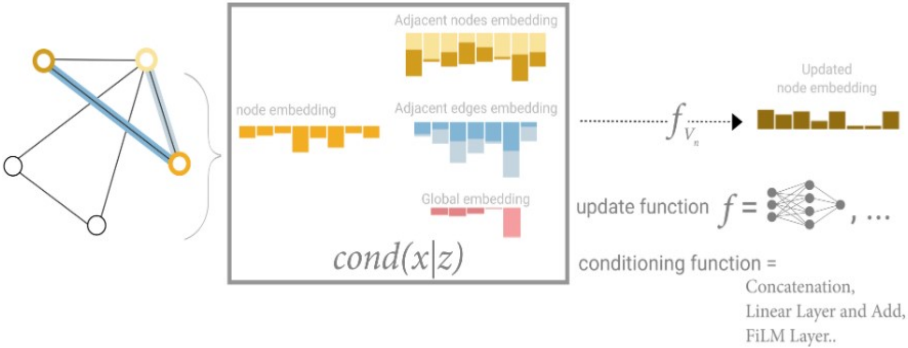


1.3.4 Different Combination Update Methods



1.3.4 Different Combination Update Methods

featurize-wise attention mechanism



Researching...

ViG: Vision Graph Neural Networks

Brand new paper from CVPR2022

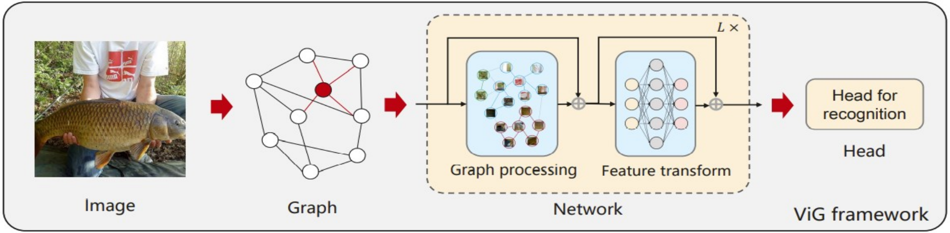
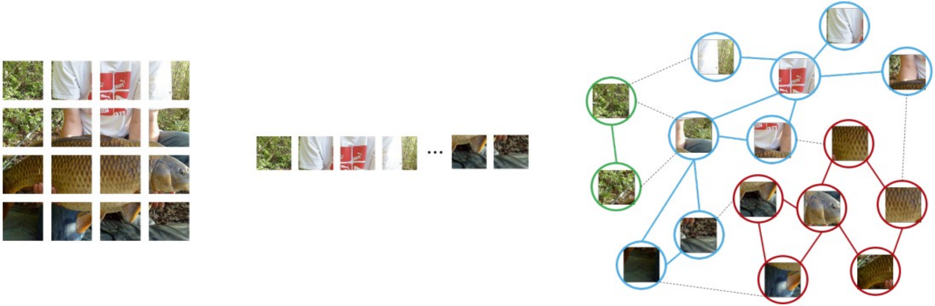


Figure 2: The framework of the proposed ViG model.

- Vision GNN: An Image is Worth Graph of Nodes Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, Enhua Wu

ViG: Vision Graph Neural Networks

Data Structure

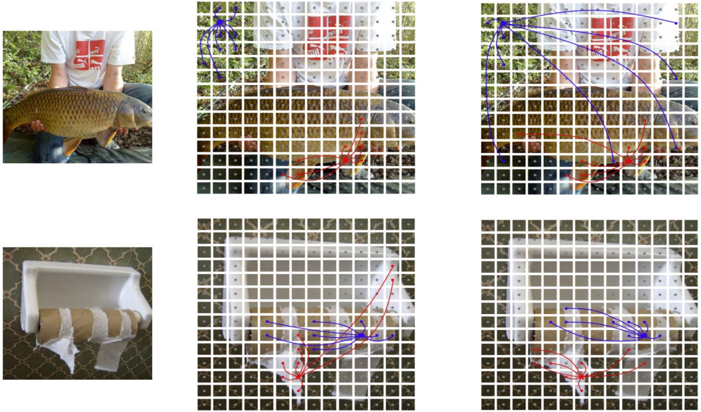


(a) Grid structure.

(b) Sequence structure.

(b) Graph structure.

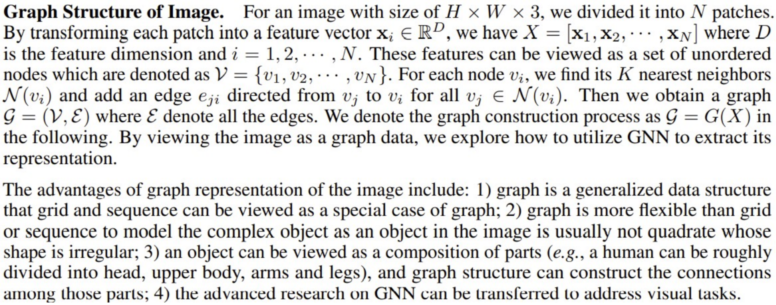
Data Processing and Pattern Learning



(a) Input image. (b) Graph connection in the 1st block. (b) Graph connection in the 12th block.

Current my work and plan

I'm working on implementing Graph Structure of Image Converting part in this paper by Python. The goal is to convert our image data to graph structure. And next step, I will use constructed dataset to train in a simple GNN architecture.



Graph Structure of Image. For an image with size of $H \times W \times 3$, we divided it into N patches. By transforming each patch into a feature vector $\mathbf{x}_i \in \mathbb{R}^D$, we have $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ where D is the feature dimension and $i = 1, 2, \cdots, N$. These features can be viewed as a set of unordered nodes which are denoted as $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$. For each node v_i, we find its K nearest neighbors $\mathcal{N}(v_i)$ and add an edge e_{ji} directed from v_j to v_i for all $v_j \in \mathcal{N}(v_i)$. Then we obtain a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E}$ denote all the edges. We denote the graph construction process as $\mathcal{G} = G(X)$ in the following. By viewing the image as a graph data, we explore how to utilize GNN to extract its representation.

The advantages of graph representation of the image include: 1) graph is a generalized data structure that grid and sequence can be viewed as a special case of graph; 2) graph is more flexible than grid or sequence to model the complex object as an object in the image is usually not quadrate whose shape is irregular; 3) an object can be viewed as a composition of parts (*e.g.*, a human can be roughly divided into head, upper body, arms and legs), and graph structure can construct the connections among those parts; 4) the advanced research on GNN can be transferred to address visual tasks.

Further

Our numpy format data can be represented as an array. So I plan to argue whether it is possible to use array format data and regard this as a speech recognition problem.