

[CS39-5: APPLICATION OF DEEP LEARNING ON GRAPH EDIT DISTANCE]

Final Report



THE UNIVERSITY OF
SYDNEY

Information Technology Capstone Project

COMP5703/5707/5708

Group Members

1. Shaokun Cao (520252055)
2. Ziter Yang (500071119)
3. Hao Zhou (5000678446)
4. Jiayu Shao (450069710)
5. Ananda Aziz Hardjanto (470469051) (Group Leader)

CONTRIBUTION STATEMENT

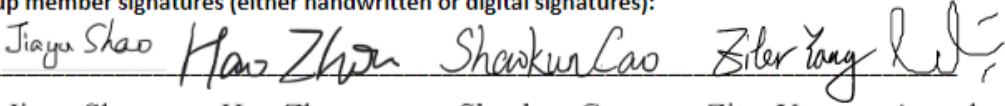
Our group, taking project CS39-5, with group members Shaokun Cao, Ziter Yang, Hao Zhou, Jiayu Shao, and Ananda Aziz Hardjanto, would like to state the contributions each group member has made for this project during semester 1 2021:

- [Shaokun Cao]: 100% (Tried coding of the innovation of TagSim with SimGnn, the A* for GeNN initial plan and did parts of the reports.)
- [Ziter Yang]: 80% (Kept the minutes from the meetings and did part of the reports.)
- [Hao Zhou]: 80% (Did parts of the reports.)
- [Jiayu Shao]: 100% (Did the main SimGNN model and parts of the reports.)
- [Ananda Aziz Hardjanto]: 100% (Did data preprocessing, ran testing of the model and parts of the reports.)

All group members agreed on the contributions listed on this statement by each group member.

Signatures:

All group member signatures (either handwritten or digital signatures):


Jiayu Shao Hao Zhou Shaokun Cao Ziter Yang Ananda Aziz Hardjanto

ABSTRACT

The term graph similarity search is an important technique in any graph based applications. This technique can be applied in many scenarios, such as finding diseases in a bunch of graph based molecules, finding similarities in social graphs, identifying chemical compounds and its similarities, and many more. In addition to the abundance of applications that graph similarity search can be used for, there are plenty of ways in which it can be implemented. Some ways are advantageous for simple search, but would not work well on heavy workload for the more difficult tasks. While other techniques might be too computationally extensive when used, and may require expensive hardware as well as plenty of time for it to run properly. In this research, the method of SimGNN neural network will be implemented for a deep learning approach to solve for Graph Edit Distance (GED). GED is one of the basis and most frequently used methods for any graph similarity search. However, GED can be very computationally extensive, and become an NP-Hard problem as it lacks the ability to adapt to different sets of data structures. In this research, the aim is to solve this by using the deep learning approach in implementing the SimGNN [9].

The neural network of SimGNN takes a couple of approaches. Firstly, the entire pair of graphs is mapped into an embedding vector by using a learnable embedding function. This provides information on the graphs being learned as a whole. Following this, node comparisons of node pairs are done in each graph level embedding. For the purpose of this research, the model is implemented on synthetically generated graph pairs for its simplicity and future adaptability of other datasets. The performance of the model itself is then measured by using mean squared error to measure its performance as well as its efficiency in solving for GED.

Upon running the model, it can be seen to be able to achieve a small number of errors, making it insignificant. This shows that the model is capable of performing GED on the given dataset with ease. Since the entire research is done by only using the hardware from Google Colab, it shows that the model itself does not require intensive computing power in order to achieve these results. This also gives the opportunity for further research to be done based on SimGNN for future works that may require a more thorough model for a more complex dataset. An example of this

would be the combination of TagSim with SimGNN, for a type aware graph similarity deep learning model [9].

TABLE OF CONTENTS

Abstract	i
Table of Contents	ii
1. INTRODUCTION	1
2. BACKGROUND / LITERATURE	1
2.1 Literature Review	1
3. RESEARCH/PROJECT PROBLEMS	2
3.1 Research/Project Aims & Objectives	2
3.2 Research/Project Questions	2
3.3 Research/Project Scope	2
4. METHODOLOGIES	2
4.1 Methods	3
4.2 Data Collection	3
4.3 Data Analysis	3
5. RESOURCES	3
5.1 Hardware & Software	4
5.2 Materials	4
5.3 Roles & Responsibilities	4
6. MILESTONES / SCHEDULE	4
7. RESULTS	5
8. DISCUSSION	6
9. LIMITATIONS AND FUTURE WORKS	6
REFERENCES	7

1. INTRODUCTION

Graph Edit Distance is one of the most popular approaches in graph similarity search. In a simple GED, it is well known for its cost-effective approach in transforming one graph, to another to solve for the GED. However, in using it for a complex dataset, it becomes a very costly approach as high computational power is necessary to gather any results. Hence, the basic approach of GED often suffers from an issue of scalability as the datasets being used increases in complexity. This causes GED to become a NP-Hard problem as the dataset becomes large and complex. Having a GED model that is scalable would be necessary when computing important datasets that are complex. The GED approach can be used on multiple useful datasets for graph similarity search such as on molecule disease comparisons and identification, chemical compounds research, and many more. This shows the importance of finding a solution for this scalability issue, as this would open for GED, and the opportunity to use this approach to solve for various complex datasets without the need of a high computation power.

To solve this issue, a deep learning approach is used by using a neural network approach to solve for the GED. One of the neural network structures that can be used for this purpose, is called the SimGNN. By using the SimGNN neural network to solve for the GED, it is able to do so efficiently, whilst still maintaining its performance effectiveness.

2. RELATED LITERATURE

2.1 Literature Review

In the sciences and social sciences, graph-structured data is commonly employed. Relational induction must be incorporated into deep learning frameworks if computers are to learn from this type of data. Deep graph embedding approaches,

the use of CNNs on graph-structured data, and the development of neural information transfer systems inspired by belief propagation have all lately received attention.

Graphs are a standard data format and language for depicting complicated systems. A graph is just a collection of items that interact with one another. A graph may be used to depict a social network by using nodes to represent people and edges to indicate two people who are friends. A biological network's nodes can represent proteins, and its edges can represent the interactions of numerous species.

The graphical formalism is useful because it focuses on the interactions of points rather than on the mass of points. Graphs, on the other hand, give more than just a visually pleasing theoretical foundation. They give a mathematical foundation for studying, analyzing, and learning from complicated real-world systems..[1]

Machine learning is a problem-solving discipline. Our models should be able to learn from data and solve specific challenges. We use supervised learning when we want our model to predict a label based on the input data; we use unsupervised learning when we want our model to cluster points. There is a difference between these two types of graph-based machine learning..

Learning similarity metrics across graphs is a popular problem in several domains including social networks, and it may help in various learning tasks such as classification, grouping, and similarity search. Deep graph similarity learning is becoming increasingly popular. The key concept is to construct a deep learning model that maps the input graph to the target space such that the distances in the target space are close to the structural distances in the input space.

Graph neural networks[3] have grown into a strong tool for learning graph representations as deep learning techniques have improved. GNNs have been processing graph-related tasks since the beginning, performing both representation learning and goal learning tasks. [2]. Compared to graph embedding techniques, GNN deep models can better exploit the properties of graphs to meet specific learning goals.

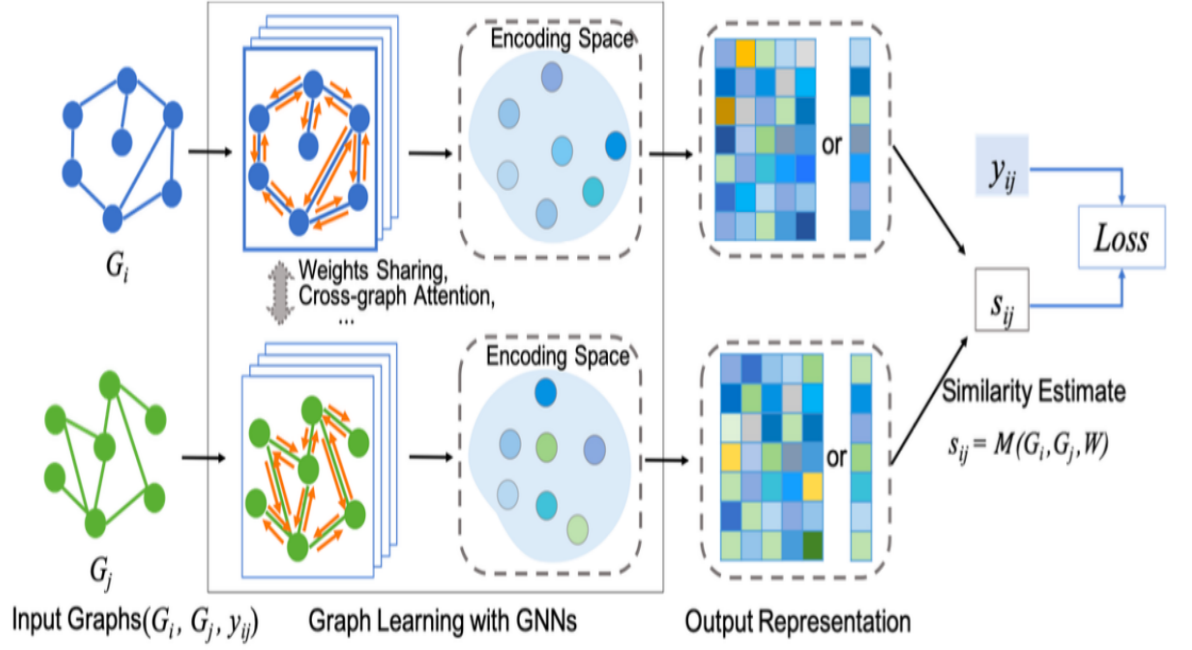
The GNN messaging system's fundamental idea is straightforward: each node gets information from its immediate neighborhood during each iteration, and as these iterations advance, each node's embedding incorporates more and more information

from the distant end of the graph. Each node embedding contains information from its neighborhood after the first iteration (k), or the features of the graph database that can be managed to reach via a path of length 1 in the graph. After the second iteration (k), or the 2-hop neighborhood, or after k iterations, or the k -h neighborhood, or after, each node embedding contains information from its neighborhood.[4]

This data is often provided in two ways. There is graph structural information on the one hand. For example, node u 's embedding may include information about the degrees of all nodes in u 's k -hop neighborhood after k rounds of GNN message forwarding. This structural information may be used to a number of situations. When looking at molecular graphs.

In addition to structural information, another important type of information collected by GNN node embeddings is feature-based information. Each node's embedding incorporates all feature information of its k -hop neighborhood after k iterations of GNN information forwarding. In terms of local feature aggregation, GNNs behave similarly to convolutional kernels in convolutional neural networks (CNNs). while CNNs acquire data from geographically defined regions in an image, Using neighborhood-based local maps, GNN aggregates data..

The goal of GNN-based similarity learning approaches is to learn graph representations while carrying out complete similarity learning tasks. A common method for GNN-based graph similarity learning models is shown in Figure 2. GNN-based GSL methods start by using a multi-layer GNN with weight W to learn the representation of G_i and G_j in the encoding space, where in a Learning on graphs may influence each other through mechanisms like weight sharing and cross-graph interaction between two GNNs. Given a pair of input graphs, where y_{ij} represents the true similarity label or score of G_i , G_j > To calculate or forecast similarity scores between two graphs, fully-connected or dot-product layers can be added after GNN layers.[4]



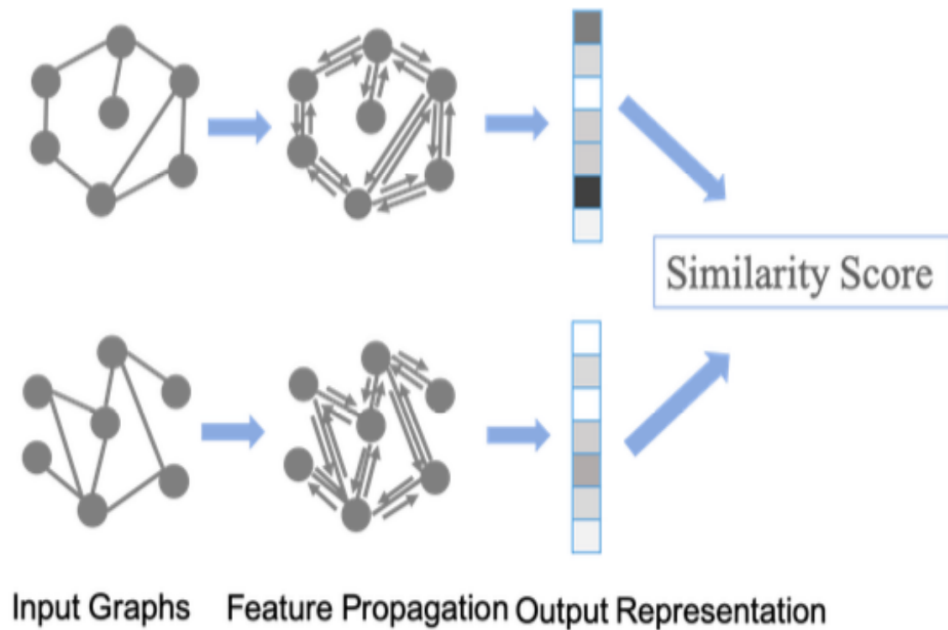
Using Siamese GCNs, each picture is transformed into a region adjacency graph with each node denoting a segmented portion of the image for content-based remote sensing image retrieval. Creating an embedding space that connects semantically coherent pictures while removing other samples is the objective. Contrast loss is applied during model training.

Because the weights of the two GNN models in the Siamese model are the same, the Siamese GNN models will make sure that the networks process the two input graphs in the same way. As a result, equivalent input graphs would be embedded in the latent space in the same way. As a result, Siamese GNNs can distinguish between two input graphs in the latent space or estimate their similarity.

In addition to the proper GNN models in the twin networks, a sufficient loss function must be specified. Another often used loss function for Siamese networks is the triplet loss [104]. In the instance of a triplet (G_i, G_p, G_n), G_p is of the same class as G_i , but G_n is not. The triplet loss is discussed more below[4][5]

$$L_{\text{Triplet}} = \frac{1}{K} \sum_K \max(d_{ip} - d_{in} + m, 0)$$

where K is the number of triplets used in training, $d(G_i, G_p)$ is the distance between G_i and G_p , and $d(G_i, G_n)$ is the distance between G_i and G_n . [5]



[4]

There are also other models like: Graph matching network[6], NeuralMCS[6][7], Hierarchical Graph Matching Network[8].

3. RESEARCH/PROJECT PROBLEMS

3.1 Research/Project Aims & Objectives

In modern society, graphs play a pivotal role in many frontier fields, including social media, chemical molecular structure, transportation planning, and bioinformatics. In the applications mentioned above, the core of the application of graphs is the calculation of graph similarity, which needs to be efficient and accurate.

Graph similarity is most commonly measured by graph edit distance (GED), defined by the minimum number of graph edit operations between a pair of graphs. However, the existing GED solvers used to calculate accurate graph edit distances have very fatal problems. When the complexity of the graph becomes higher to a

certain extent, the precise GED calculation will become an NP-Hard problem. Therefore, an efficient and accurate method to calculate GED is urgently needed. In summary, the objective of the project is to use the neural network to build a model to obtain a relatively accurate and efficient GED.

3.2 Research/Project Questions

The main question of this research paper is how the model that have been chosen, SimGNN can be used as a neural network for deep learning, to solve the NP-Hard problem of Graph Edit Distance. In addition, it is also important to create a SimGNN model that can perform and solve this problem effectively and efficiently. Therefore, it is also important to find a method to apply this deep learning method, without sacrificing any efficiency and increasing its computational cost.

To be able to measure the research question to an extent and find its solution, it is important to be able to find a method of performance measurement that is consistent, to ensure that the research question is solved, in an unbiased manner. The model is then measured in a consistent way by using the similarity score of mean squared error for evaluation. In addition, to avoid any inconsistency, the same dataset will be used throughout the research to ensure that the variety in results is not due to the different complexity in datasets.

3.3 Research/Project Scope

The scope of this research is to implement the SimGNN model on a dataset, that will be explained further in detail in the following section. The model that will be implemented is based on the paper of “SimGNN: A Neural Network Approach to Fast Graph Similarity Computation” by Yungshen Bai. In this research, it will focus its performance on only one dataset, in order to create a consistent environment and focus on the model, and how it can be improved upon to create a more effective and computationally inexpensive way to run towards simple datasets.

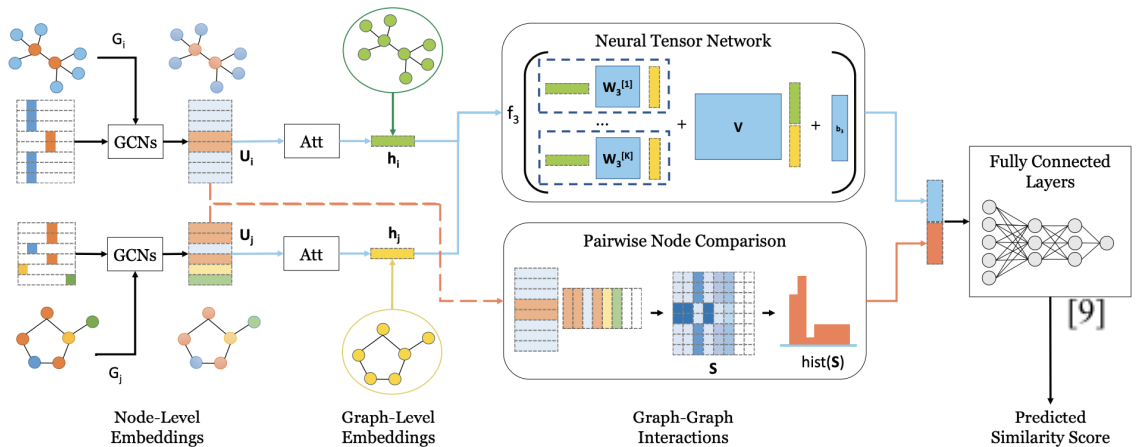
However, even though the research is focused on the model and a simple dataset, it can be used further on other datasets, without changing the majority of the model, and only some of the data pre-processing. This model is done this way to

create adaptability, for when in future instances, a different dataset needs to be used with the model for other research purposes.

4. METHODOLOGIES

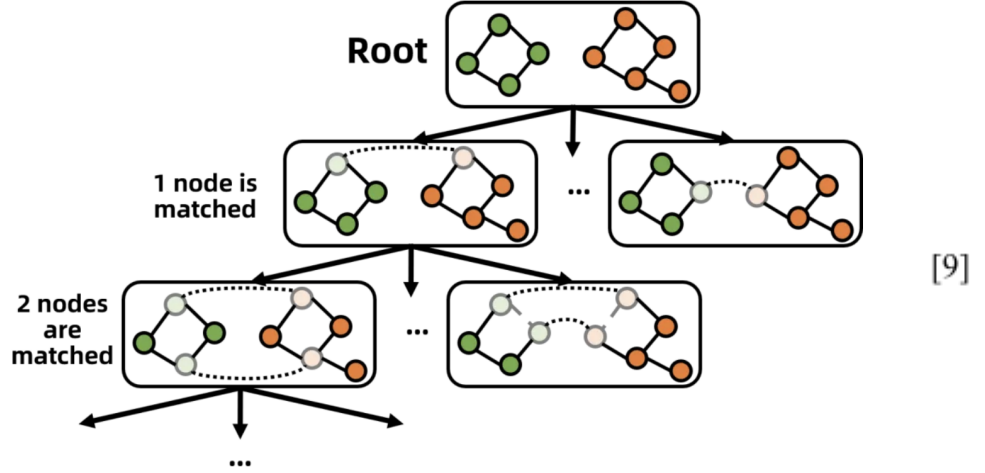
4.1 Methods

In this project, the SimGNN method will be applied to solve the problems raised in section 2. A relatively accurate GED will be obtained by the SimGNN model more efficiently than the traditional method of searching for an accurate GED. SimGNN is an end-to-end neural network model. After training, it has the ability to map a pair of graphs to GED Score. The model structure of SimGNN is shown in Figure 1,2,3. It takes several steps to convert a pair of graphs into GED. First, since the graph cannot be read directly by the model, the graph information needs to be converted into a vector, which will be encoded according to the characteristics of all nodes in the graph and the structure of the graph and then converted into a vector. Then, two strategies will be used, so that the information of the pair of graphs can interact at two levels, the two levels are the interaction at the node level and the interaction at the graph structure level. After the interaction of these two levels, a vector will be obtained, and then this vector will be passed to a fully connected neural network to obtain a GED score. Following paragraphs would be used to introduce two strategies in detail.



4.1.1 Nodal-level Embedding Interaction Strategy

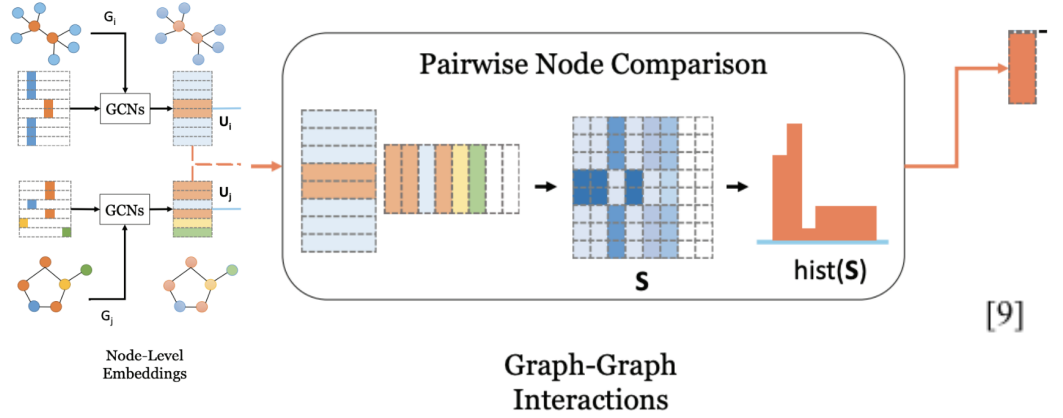
In traditional computer theory research, scientists have proposed a tree search-based algorithm for solving the optimal solution to the graph edit distance



problem. In the conventional tree search algorithm, as shown in the figure above, each state in the tree is a graph structure with two partial matches, and when each time the search tree becomes deeper, a pair of matched nodes will be added, so the interaction of nodes is an integral part of this algorithm. Therefore, nodal-level embedding with nodal-level interaction of a pair of graphs would be implemented in SimGNN model.

In order to obtain nodal-level embedding, the graphs need to be converted into vectors that can be passed into the model. The convolutional graph network, which is one of the most commonly used graph embedding approaches, would be used as the aggregation function to aggregate the neighbor information. As shown in the figure below, different colors of the node indicate the types of nodes. Hence, the first step for deriving nodal level embedding is to apply the one-hot encoding method based on the types of nodes. Meanwhile, the adjacent matrix that represents the information of the graph structure would be derived. Then, the one-hot encoded matrix of the node features would be used to conduct matrix multiplication with the adjacent matrix to generate the preliminary graph embedding.

First strategy: nodal level interaction



[9]

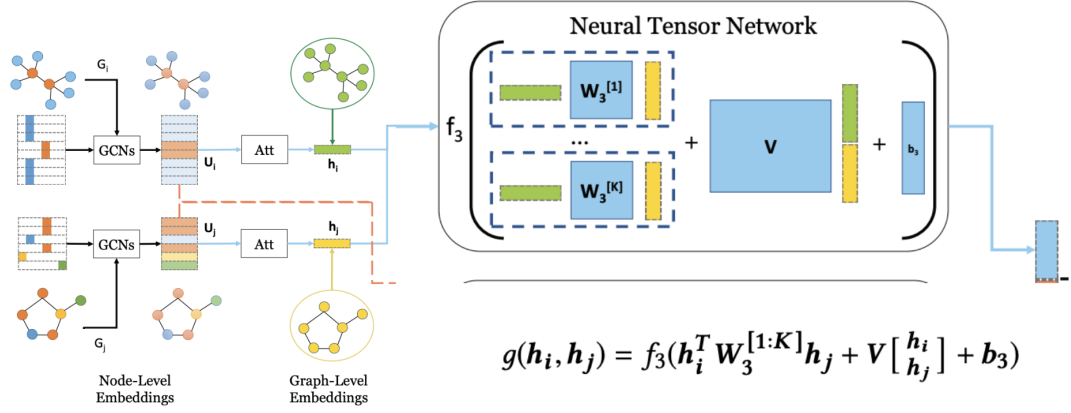
The preliminary graph embedding matrices G_i and G_j will be fed into GCN (3 layers of GCN) to aggregate neighbour information, and then the nodal-level embedding matrices of the graph pair will be derived from the multiple GCN layers, which is shown in the figure as U_i and U_j . Then, the first step of nodal-level interaction is to multiply two nodal-level embedding matrices, which would result in a matrix S shown in the figure and defined as the pairwise node similarity matrix. Following, the matrix needs to be vectorized so that it can be fed into the fully connected network to compute the GED score. Therefore, the SimGNN model utilizes the histogram to extract the features from the pairwise node similarity matrix(S), and the function is defined by $hist(S) \in R^B$ where B as the hyper-parameters is the control of bin number. After that, the histogram feature vector is normalized and ready to be fed into the fully connected network(FCN).

4.1.2 Graph-Level Embedding Interaction

Since only the information on node interaction is not enough to represent all the information of the graph, an embedding method that can represent the graph's global information is critical to improving the model's performance. The SimGNN model implements the Attention mechanism to further aggregate the information from the nodal-level embedding. The reason why the attention mechanism is selected is that the attention mechanism is a global context-aware approach, and it is capable

of resulting in weight matrices that weigh heavier on the parts of the graph having better interpretability.

Second Strategy: Graph level interaction

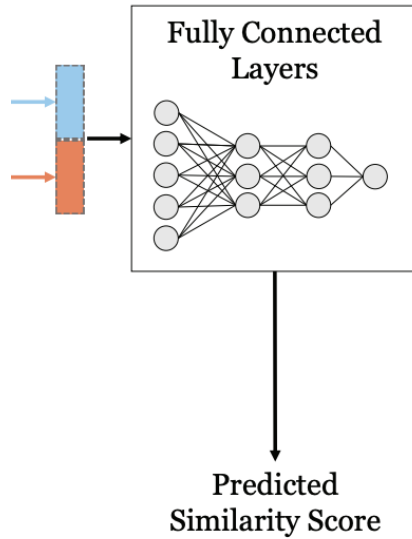


[9]

As shown in the figure above, the nodal-level embeddings of the graph pair are passed into the attention layer individually. Following that, two graph-level embedding vectors h_i and $h_j \in R^D$ are then passed into the so-called Neural Tensor Network (NTN) to conduct graph-level interaction. The NTN layer could be formulated as $g(h_i, h_j) = f_3(h_i^T W_3^{[1:K]} h_j + V \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b_3)$ where parameters $W_3^{[1:K]} \in R^{D \times D \times K}$, $V \in R^{K \times 2D}$ and $b_3 \in R^K$ are all trainable. The most commonly use matrices interaction is matrix multiplication which is $h_i \times h_j^T$ in this case. However, it usually results in interacting insufficiently and weakly so that the parameters W , V and b are introduced to strengthen the interaction between two matrices. Furthermore, the K hyperparameter is used to control the size of the graph-level similarity score vector of the pair of graphs. By passing two graph-level embeddings into the NTN layer, the graph-level similarity score vector would be generated and would be fed into the FCN for computing the GED score.

4.1.3 Computing the GED Score

After the similarity score vectors from both nodal-level and graph-level interaction, these two similarity score vectors would be concatenated to form one vector, as shown in the figure below. The concatenated similarity score vector is then fed into the Fully connected layers to give a prediction of the similarity score.



4.2 Data Collection

For the research on the application of SimGNN deep learning on GED, a large amount of data was reviewed to find the right dataset that would fit and support the research purpose. For this specific research, a synthetically generated dataset was chosen. The dataset was retrieved from a GitHub repository that is linked in the resources below. The dataset was used for a PyTorch implementation of a similar research on SimGNN and its use as a neural network for graph similarities.

The dataset downloaded consists of 11,000 json files, separated to test and train sets. There are 1,000 test json files and 10,000 train json files. Each json files consists of two graphs, considered to be pair graphs in which will be analyzed by the model. Information on these graphs are their nodes along with their respective labels. In the json file, there is also additional information on the pair graph's GED ground truth, which will be used to calculate the mean square error of the SimGNN model.

This was useful to get a precise evaluation on the performance of the deep learning model. The graph nodes, labels and GED are represented by integer values, to make any calculations of GED easier to interpret.

4.3 Data Analysis

Since there is a vast amount of graphs and information to be analyzed from the dataset, some data preprocessing was required in order for it to be a suitable input for the model to train and test from. To import and store the json files and its data, two lists marked as train and test are created. In each list, their respective data is imported into them, with each json file being in each index. An example, index one from train, would pull the pair graph and all its included information from the first json file in the train set. This was done in order for further specific information to be extracted in future processing for ease of working.

Following this, it is important in deep learning of GED, for the unique labels to be identified beforehand, in order to find the proper calculations. This was done by creating a new list of combined data from train and test, and extracting all the labels included. From here, the list was then formatted to a set, to get all the unique labels that are in the dataset. Further processing was done by sorting the labels from the smallest integer, and creating a dictionary set for the labels with indexes enumerated to it for easy access.

Lastly, an adjacency matrix is created. The adjacency matrix is needed to create a one hot encoding matrix of the positions of the nodes. This is important as the model will take in this matrix as an input. This was done by creating a new list to store all the converted graph pairs into its adjacency matrix. To do this, the train and test dataset is run through a function that was built and was inspired from the same github repository. This gives the SimGNN model a way of identifying each node and its location easily.

5. RESOURCES

5.1 Hardware & Software

The software will run on google colab which can be implemented in a virtual environment on all major operating systems (Windows, Mac and Linux) on all modern hardware architectures (iPhones and Android smartphones). We used Python for this project because data scientists frequently utilize Python and Python is a general-purpose high-level programming language, to create deep learning algorithms. It has many power techniques for the programmer to built the model such as :numpy, matplotlib, tensorflow and pytorch.

We used the Google colab to build our model as it is easier for us to work together.

CPU: Intel(R) Xeon(R) CPU @ 2.20GHz

Memory:13297228 kB

5.2 Materials

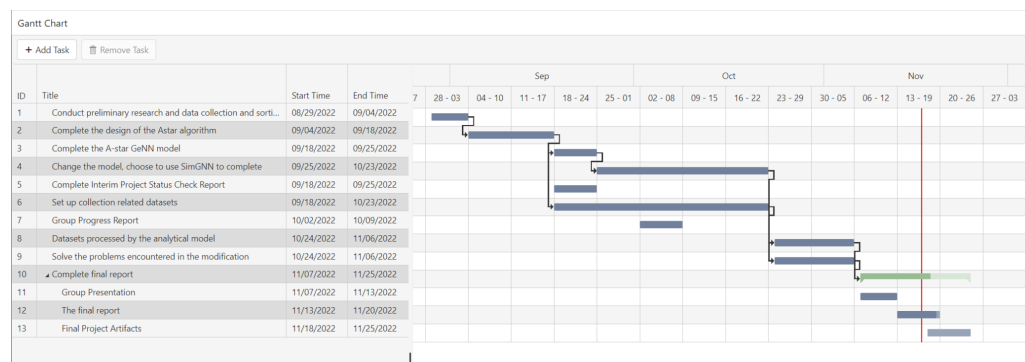
For this research, to run the program, there are several python packages that were installed and downloaded to aid with the program. Some of these packages are mentioned below. Further explanation on each packages can be found on each source page online.

Python Packages: Glob, Numpy, TQDM, argparse, json, random, tensorflow, OS, Torch, Keras, and Sklearn

5.3 Roles & Responsibilities

- Ananda: Group leader, Data preprocessing, training and testing the Simgnn model and do some part of the report.
- Jiayu: Build the Simgnn model and train the Simgnn model and do some part of the report.
- Ziter: Document the content of the meeting and the progress of the project and do some part of the report.
- Hao: Do some part of the report
- Shaokun: Build the A-star algorithm and combination of Tagsim with Simgnn(didn't figure it out) and do some part of the report.

6. MILESTONES / SCHEDULE



The whole project is mainly divided into five stages to complete.

29/08/2022-04/09/2022 is the first stage, in this stage, the group reads and learns relevant materials, and selects a suitable model from the materials for further research.

From 04/09/2022 to 18/09/2022 is the second phase. At this stage, after the team had learned and understood several models, the Astar model was selected for further study.

The third stage is from 18/09/2022 to 23/10/2022. The team needs to complete the GENN model of the Astar algorithm and collect the relevant data sets. And complete the mid-term project status review report and team progress report during this period. At this stage, it was finally decided to change the model due to the

complexity of the previously selected model and the fact that the model involved some programming languages that the team members did not know.

Then enter the fourth stage, from 24/10/2022 to 06/11/2022, the team needs to process the collected data set and analyze the results, modify and improve according to the problems that arise in the process.

Finally, in the fifth phase, from 06/11/2022 to 18/11/2022, the team worked together to complete the final project report.

Milestone	Tasks	Reporting	Date
Week-1	Analysis and selection of research topics	Topic selection and staff confirmation	07-08-2022
Week-2	Determining group meeting times and details in the study	None	14-08-2022
Week-3	Learn and analyze research-relevant models	None	21-08-2022
Week-4	Choose one of the models for further study and design a work plan	None	28-08-2022
Week-5	Conduct preliminary research on selected models and complete group proposal reports	Group Proposal Report	04-09-2022
Week-6	Complete the design of the Astar algorithm	None	11-09-2022
Week-7	In the process of completing the model, we found it to be more difficult than expected and beyond the knowledge of the team members.	None	18-09-2022

Week-8	After discussion we decided to replace the model and speed up the process to complete it.	Mid Semester Project Status Check Report	25-09-2022
Week-9	Complete the Group Proposal Report	Group Proposal Report	09-10-2022
Week-10	Finish the model & Datasets processed by the analytical model	Report to supervisor to give input on our model.	16-10-2022
Week-11	Solve the problems encountered in the modification	None	23-10-2022
Week-12	Solve the problems encountered in the modification	None	30-10-2022
Week-13	Completion of Final Report	None	06-11-2022
Week-14	Final Presentation	Group Presentation Slides and Video	11-11-2022
Week-15	Final Report (thesis) Submission	Group Final Report	18-11-2022
Week-16	Final Deliverable Submission	Project Artifacts, Codes, Datasets, Reports or Outcomes	25-11-2022

7. RESULTS

To gather the results and measure the performance of the SimGNN model to solve for GED with the dataset provided, a couple of methods were applied. The baseline error and the test error was calculated for measurement. To achieve this, the

training data was run through the SimGNN model to train it, and the trained model is then used on the test data set to see how well it performs.

Firstly, the baseline error of the ground truth was generated, in order to find the variation of the GED ground truth. This was done by calculating the difference between the mean of the GED and its ground truth value. The baseline error was created to have a deeper understanding of the dataset.

The second method of measurement, the evaluated score that was generated from running the test dataset on the trained model was used to generate a mean squared error of the performance of the model. These measurements identify the similarities between the generated GED from the model, with the similarity of the GED of the ground truth that was given from the dataset. This measurement helps evaluate the performance of the overall model, to see how precisely the model can predict and generate the correct GED, which is the goal of this research. A smaller model test error or mean squared error, would mean that the model is performing relatively well and effectively.

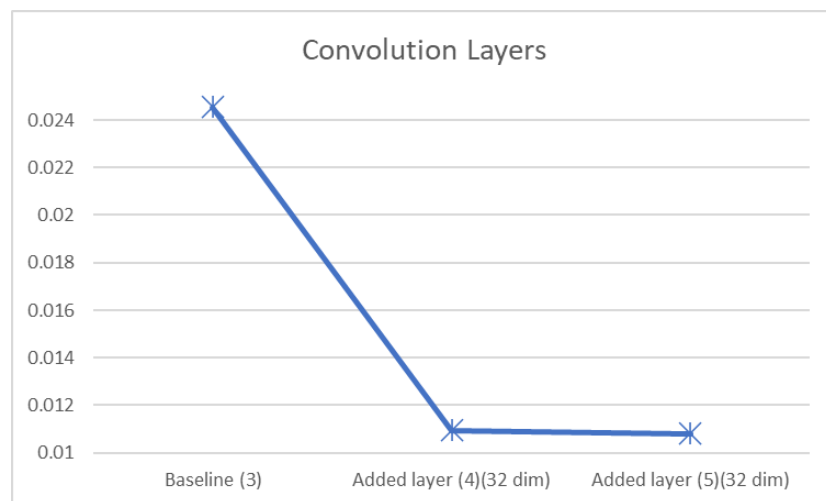
To gather more information on performance of the model, it is run six times, each with different aspects of the model being changed to identify what can be done to improve its performance. The first set of tests was on different structures of convolution layers on the main SimGNN structure. There is the baseline structure with three convolutional layers, followed with four and lastly five convolution layers. All the added convolution layers and the baseline last layer is with dimension of 32 to keep it constant. The second set of tests was on different type of activation function for the activation layer to see which would work best on this specific dataset. The three activation functions tested were the ReLU, Sigmoid and Tanh.

Convolution Layers	Model Test Error
Baseline (3)	0.02455
Added layer (4)(32 dim)	0.01095

Added layer (5)(32 dim)	0.01081
--------------------------------	---------

Table 1: Result of different Convolution Layers

The above table shows the results gathered on the test of different convolution layers. The model test error as previously stated is the mean squared error of the results from the SimGNN model. The graph below shows a graphical interpretation on how the performance of the model changes on each added convolution layer.

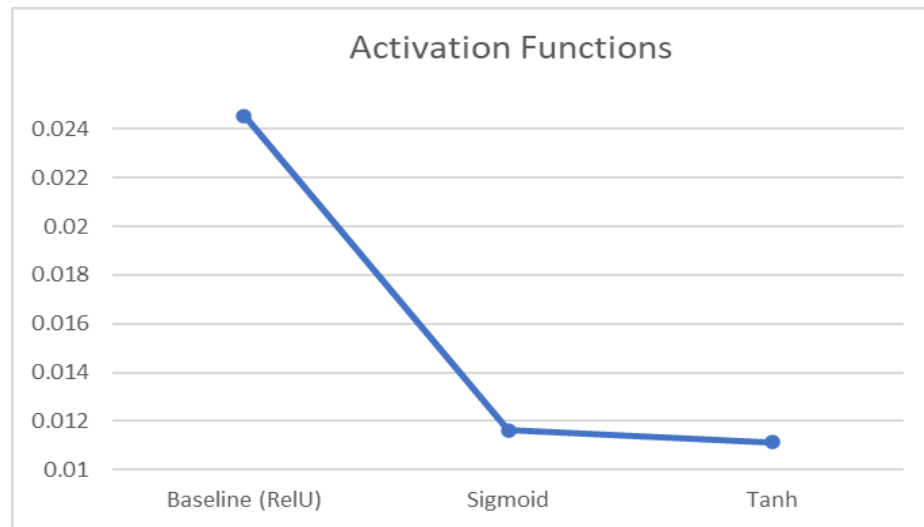


Graph 1: Result of different Convolution Layers

Activation Layers	Model Test Error
Baseline (ReLU)	0.02455
Sigmoid	0.01162
Tanh	0.01115

Table 2: Result of different activation functions

The table above is the results from the different activation functions. There is the baseline function of ReLU, then the Sigmoid and Tanh function. The activation function changed is on the main SimGNN model itself only, to create a consistent environment to ensure that it is the only thing that is affecting the changes in the performance of the model. As for the graph below, it shows graphically how the performance of the model changes in relation to the different activation functions used.



Graph 2: Result of different activation functions.

8. DISCUSSION

8.1 The results

Our model compares the Baseline model with the model with different layers and the model with different activation functions.

From Table1 and Figure 1, we found that adding four layers would significantly reduce the error of the model, from 0.02455 to 0.01095. However, when adding the fifth layer, the improvement would be very small.

From Table 2 and Graph 2, we can find that changing the activation function from ReLU to sigmoid greatly improves the performance. The model test error of

ReLU is 0.024. Is more than double the model error when using sigmoid. But the change from sigmoid to tanh leads to small improvement, and the difference in error is less than 0.005.

8.2 Implications and significance

In this research, we mainly studied the graph similarity calculation method based on graph neural network, which is used to solve the graph similarity problems in many cross fields such as science, medical imaging, character recognition and computer science. Through our work, we implement SimGNN, and get good model results with low model error, which can be helpful to calculate the graph similarity model.

In a previous study(Y.S.Bai et al.), concluded that the error result of SimGNN applied in the graph similarity calculation was about 0.018-0.015, which was close to our result. And the result of our model would be better. Based on these data, we think our research results are good and can be applied to some graph similarity calculations.

At the same time, as mentioned earlier, we also found some small details. First, adding more layers can improve the accuracy of the model, but this improvement will not be significant in a certain depth, and will only increase the complexity of the model. Therefore, we should choose the appropriate number of layers to avoid the model being too complex and difficult to generalize. Secondly, when we added nonlinear unit activation functions, we found that some activation functions, such as ReLU, performed very poorly, but Sigmoid and Tanh functions performed almost well. This may be related to the data set, so we may change this unit after trying more data sets.

8.3 Response to project expectations

Finally, we completed the requirements for the project initiator, including but not limited to the following:

1. We can well understand and master the basic principle of graph neural networks. As early as the first few weeks, we have studied and discussed the literature. We use PPT to share our understanding and experience, and can achieve the expected goals.
2. We understand the GED model framework based on the depth model, including A*, SimGNN, TagSim, GENN, etc. We have all studied and studied to understand their basic principles and central ideas, and can achieve the expected goals.

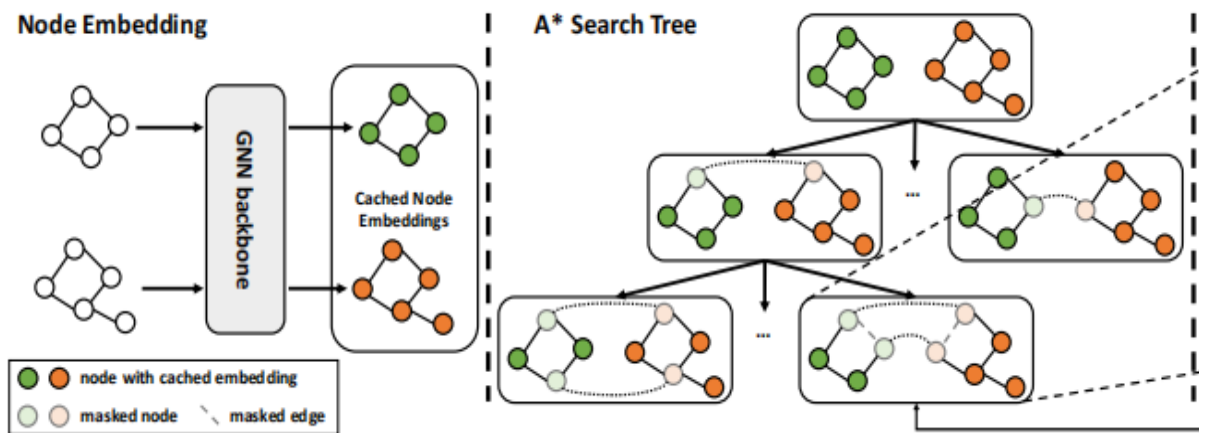


Fig.3 One of the GED model we learned(A*-GENN) [10]

3. We have successfully realized the excellent GED model: the combination of TagSim and SimGNN models has achieved good results and achieved the expected goals.
4. At the same time, we can understand the meaning of the model algorithm, use and combine the model flexibly, and achieve the expected goals.

9. LIMITATIONS AND FUTURE WORKS

9.1 Limitations

Our limitations are mainly in two aspects: on the one hand, the selection of data sets; on the other hand, the limitations of time and personnel make our model limited.

First of all, we are limited to only using one dataset. We can not comprehensively complete the test of the model and study the excellence and adaptability of the model in different data sets. We lack the ability to process different types of data sets. When we try to use some commonly used data sets, such as LINUX data sets, it will not work properly because we need the file in .json format, but no member can convert the data set into the type we need.

Secondly, Due to the limited number of members and time, the combination of SimGNN and TagSim that we plan to complete has not been completed. In the process of implementation, we found that the attention layer in SimGNN needs to consider the relationship between nodes, but when we try to integrate it into TagSim, we are faced with the problem of node relabeling, which will cause problems in the connection of node information between SimGNN and TagSim, and this idea has not been successfully implemented.

9.2 Future works

In view of the limitations in our work, we need to learn more data preprocessing techniques and try to solve the problem of model combination in future study and research.

For data set preprocessing, we can focus on data set format processing, and convert various data sets into different types of format files to meet different model requirements. We can select more datasets to learn and compare our model to verify the generalization of the model. At the same time, good data sets can train good models. We also need to learn and study how to evaluate the excellence of a data set, and conduct reasonable screening when selecting data sets. Using better data sets will benefit our model.

As for the combination of models, we need more time and research on the binding model in some details. If we can complete the combination of models, we can better apply it to the study of graph similarity, which can improve the generalization of models and reduce the prediction error. And we can also try many other combined models, such as A * GENN model. The existing literature shows that it is a better complex model, which requires us to spend more time to study.

In general, graph similarity computing is a relatively complex and profound learning field. There is still much learning work to do in the future. We need more studying, as well as continuous innovation to make the graph similarity computing work progress with the development of the times.

REFERENCES

- [1] Hamilton, William L. "Graph representation learning." *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, no. 3 (2020): 1-159.
- [2] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [3] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.
- [4] Ma, G., Ahmed, N.K., Willke, T.L. *et al.* Deep graph similarity learning: a survey. *Data Min Knowl Disc* 35, 688–725 (2021). <https://doi.org/10.1007/s10618-020-00733-5>
- [5] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [6] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. *ICML*, 2019.
- [7] Yunsheng Bai, Derek Xu, Ken Gu, Xueqing Wu, Agustin Marinovic, Christopher Ro, Yizhou Sun, and Wei Wang. Neural maximum common subgraph detection with guided subgraph extraction. <https://openreview.net/pdf?id=BJgcwh4FwS>, 2019.
- [8] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Chunming Wu, and Shouling Ji. Hierarchical graph matching networks for deep graph similarity learning. <https://openreview.net/pdf?id=rkeqn1rtDH>, 2019.
- [9] Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., & Wang, W. (2020, March 01). SimGNN: A neural network approach to fast graph similarity computation. Retrieved November 18, 2022, from <https://arxiv.org/abs/1808.05689>

[10] Bai, J., & Zhao, P. (n.d.). Tagsim: Type-aware graph similarity learning and computation - VLDB. Retrieved November 18, 2022, from <https://www.vldb.org/pvldb/vol15/p335-zhao.pdf>