Convolutional Graph Neural Networks for

David Enrique Nieves Acarón

ABSTRACT

As neural networks become an increasingly important field of application for many modern-day use cases, there naturally arose a need to address the subject of non-eucledian data. Most data types processed in conventional neural networks are "eucledian" in nature or which have distances that obey the laws of eucledian geometry. Such examples include images, text, and audio. In contrast, non-eucledian data does not necessarily obey these laws. Examples of abundant sources of non-eucledian data include: social networks, citation networks, traffic networks, satellite networks, 3D meshes, and molecules, to name a few.[1] In this project, a brief exploration of the research regarding Graph Neural Networks will be conducted, and some experiments will be conducted showcasing their effectiveness in performing graph-related machine learning tasks.

Keywords: Graph Neural Networks, Deep Learning, Geometric Deep Learning

1. INTRODUCTION

Early work on Graph Neural Networks (GNNs) can be traced back to the work of Sperduti and Starita in 1997 where a neural network was applied to a directed acyclic graph. Reference[1] denotes four types of GNNs: Recurrent GNNs (RecGNNs), Convolutional GNNs (ConvGNNs),Graph Autoencoders (GAEs) and Spatio-Temporal GNNs (STGNNs).

To elaborate briefly on the four types, they can be summarized as follows: RecGNNs attempt to learn node embeddings using recurrent neural architectures and are distinguished for inspiring later research on ConvGNNs; ConvGNNs aim to generalize the idea of convolutions as seen in image processing in order to be able to perform convolutions on graphs much like in Convolutional Neural Networks (CNNs); GAEs work much like the auto encoders found in deep learning literature in that by training a neural network to reconstruct graph data using a compressed form of the input, it can therefore lean network embeddings or generate new graphs; finally, STGNNs attempt to address graphs that are constantly changing attributes, such as social and traffic networks.

From a foundational standpoint, most of the theory of Graph Neural Networks seeks to utilize the recent developments in Deep Learning to improve accuracy on Graph classification tasks. It should be noted that the principal way of graph classification was Graph Kernel methods which use a kernel function to measure the similarity between pairs of graphs

However,[1] note that one problem with Graph Kernels have is that they suffer from computational bottlenecks, which hinders their usability.

In this paper, convolutional graph neural networks are divided into two categories: spectral models and spatial models. Spectral models have a more theoretical foundation rooted in graph signal processing. In contrast, spatial models seek to generalize the notion of a traditional convolution as commonly taught in signal processing classes, except with the use of nodes on a graph instead of pixels or values in an array.

The authors list three main limitations of spectral CNN's: the fact that any perturbation causes a change in the eigenbasis, second; the learned filters can only be applied to the graph they are trained on; third, the $O(n^3)$ complexity of eigen-decomposition is prohibitively expensive for some applications.

The main idea in spatial-based ConvGNNs consists of convolving a node's representation with the aggregation of the node's neighbors

First, there is the topics of choosing how many nodes to select for aggregation, as in some graphs, the number of nodes in the neighborhood of a given node could be in the tens of thousands, if not millions.

There are two approaches to handling this: sampling and clustering. Sampling naturally involves sampling a subset of nodes in some determined fashion, such as a random sample. For example, some approaches such as[2] opt to select a random sample of the node's neighbors. Other approaches could take into account a subset of

significant nodes by perhaps defining a significance measure that takes into account the node feature vector of a subset of the nodes in the neighborhood of the root node. On the other hand, clustering involves joining nodes together to reduce the total number of nodes that need to be processed. To this end, a variety of well-known clustering algorithms could be utilized, such as DBScan or k-means clustering, to name a few.

Both approaches have the potential to improve Graph Neural Network (GNN) execution speed. However, this comes at the cost of undermining the integrity of the graph information. Hence, a suitable trade-off between graph integrity and GNN execution speed must be selected when desigining a GNN.

Then, there is the topic of how to aggregate these nodes. The work presented in[3] represents graph convolution as a message passing process where nodes can pass information in a way defined by the message passing function U_k as referenced in.[1] This message passing scheme is described in the aforementioned source like so:

$$h_v^{(k)} = U_k(h_v^{(k-1)}, \sum_{u \in N(v)} M_k(h_v^{(k-1)}, h_u^{(k-1)}, x_{vu}^e))$$

... with U_k and M_k in being learnable and usually in the form of Multi-Layer Perceptrons (MLPs).

Drawbacks of GNN's

Some notable GNNs such as GCN and GraphSage cannot distinguish different graph structures and that depending on the readout function used, a GNN may only be as effective as the Weisfeiler-Lehman test

In addition, in the work of Li et al.,[4] it is shown that a GNN of the convolutional kind (i.e. a **ConvGNN!** (**ConvGNN!**)) will decrease its performance as the number of graph convolutional layers increases due to these kind of layers causing close by nodes to come closer together. In fact, the same authors claim that with an infinite number of convolutional layers, the representations of the nodes should converge to a single node.

The rest of this work is organized as follows: **Problem** describes the specifics and importance of the problem faced, and the models which are being investigated along with their associated training and testing phases, their complexity, and their advantages and disadvantages; **Experimental Outcomes** explains the nature of the problem, the data, and the experimental setup along with some results and some light discussion; **Discussion** describes the experience in summary, and discusses some ideas about enhancements to the models used.

2. PROBLEM

The problem of social media analysis is one that is of interest to many companies due to their

the problem of suggesting friends or videos to watch can easily be converted into a Graph Neural Network formulation by asking which edge between the current user's node and any other set of nodes of interest has the highest probability. For that reason, the GNN needs to predict the probability of that connection.

A large amount of the data used in this project consists of datasets from the Stanford Large Network Dataset Collection by the research group Stanford Network Analysis Project (SNAP).[5] In addition, other datasets from the Torch geometric benchmark datasets are used,[6] such as the `TUDataset`.

In total, adapting the data to be used in the `torch`$_{geometriclibraryhasbeenanareaofpre-processingthatshouldbeconsidere}$

`Ideally, most of these vectors`

`These datasets were downloaded first`

3. EXPERIMENTAL OUTCOMES

4. DISCUSSION

`ONGOING`

`Fitting data to run properly with graph-level, edge-level, and node-level tasks`

- `run graph-level tasks, node-level tasks, and possibly edge-level tasks`
- `demonstrate the above with more than one dataset`
- `demonstrate the above with several architectures of ConvGNNs`

REFERENCES

[1] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S., ‘‘A comprehensive survey on graph neural networks,’’ *IEEE Transactions on Neural Networks and Learning Systems* **32**, 4--24 (jan 2021).

[2] Hamilton, W. L., Ying, R., and Leskovec, J., ‘‘Inductive representation learning on large graphs,’’ (2018).

[3] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E., ‘‘Neural message passing for quantum chemistry,’’ (2017).

[4] Li, Q., Han, Z., and Wu, X.-M., ‘‘Deeper insights into graph convolutional networks for semi-supervised learning,’’ (2018).

[5] Leskovec, J. and Krevl, A., ‘‘SNAP Datasets: Stanford large network dataset collection.’’ http://snap.stanford.edu/data (June 2014).

[6] Fey, M. and Lenssen, J. E., ‘‘Fast graph representation learning with PyTorch Geometric,’’ in [*ICLR Workshop on Representation Learning on Graphs and Manifolds*], (2019).