The Future of Graph-Based Machine Learning: An In-depth Analysis of GAT and GPS Models

Ricky Miura rmiura@ucsd.edu

Yusu Wang yusuwang@ucsd.edu

Gal Mishne gmishne@ucsd.edu

Abstract

This paper presents a comprehensive analysis of advanced methodologies in *Graph Neural Networks* (GNN), with a specific focus on *Graph Attention Networks* (GAT) and the *General, Powerful, Scalable* (GPS) *Graph Transformer* (GT) model. GNN have emerged as a potent tool for learning representations of graph-structured data, tackling a wide array of problems ranging from social network analysis to molecular structure interpretation. We delve into the nuances of GAT, an architecture that introduces an attention mechanism into GNN, enabling the model to emphasize critical nodes and thus learn more robust representations. Our analysis also explores the GPS framework, which addresses the scalability and efficiency challenges in GNN by combining message passing neural networks with global attention layers, thus balancing computational complexity with representational power. We compare and contrast the performance of these models of different tasks on graph-structured data, highlighting their strengths and limitations. Through extensive experiments, we demonstrate the efficacy of GAT and GPS models in capturing complex graph structures and patterns, underscoring their potential in advancing the field of graph-based machine learning.

Code: https://github.com/RickyMiura/GNN-Analysis

1 Introduction

One of the more recent developments in the field of machine learning is *geometric deep learning*; a subset of deep learning in which we focus on performing tasks specifically on graph-structured data. We begin by asking the question, "What is a graph?" A *graph* is a type of data where we have *nodes* representing an entity and *edges* between these nodes representing some sort of relation. These nodes and edges may contain individual features themselves that give them characteristics. Some common forms of graph structured data include social networks where nodes represent people and edges represent connections between people or citation networks where nodes represent documents and edges represent links cited between the documents. Common deep learning techniques meant to deal with euclidean data such as tabular data or images would not work on graphs as they are forms of non-euclidean data so we aim to find innovative approaches to performing machine learning tasks on this new form of data.

The main tasks associated with geometric deep learning are node classification and graph classification. *Node classification* is when we aim to classify missing nodes within the graph using features and labels of neighboring nodes. In a social network, we may know the existence of a relationship between two people but not know who the person is so we would perform node classification to solve this type of problem. *Graph classification* is another classification problem only this time we aim to classify the entirety of the graph by learning about the nodes' relationships and structure of the whole graph. A common application of this is the classification of molecules by looking over a whole graph of atoms (the nodes) and their bonds (the edges). Another task in geometric deep learning is *link prediction* where we predict edges between nodes. In a social network, we could use neighboring people and attributes of two people to predict whether or not we think they would be friends.

As you can see, there is a big presence of graph-structured data in the real world and the way we deal with these associated problems is different because we see different sorts of structures within their respective domains. Current deep learning techniques like *Multilayer Perceptrons* (MLP) and *Convolutional Neural Networks* (CNN) learn over data with fixed structure and their models learn over these fixed parameters. Unlike the techniques mentioned above, *Graph Neural Networks* (GNN) must add another level of generalization and adaptability where it can learn over graph networks with nodes of varying degrees and diameters.

2 Challenges of GNNs

2.1 Over-smoothing/Over-squashing

As the field of geometric deep learning develops, many researchers have found new and efficient ways to perform tasks associated with graphs. When developing algorithms for these tasks, we are faced with two challenges known as *over-smoothing* and *over-squashing*. Different forms of GNN all follow the same foundation where for each node, some sort of aggregate is performed on its neighbors and passed into a dense neural network which gives us a new embedding of that node. Like CNN, GNN also have layers but in a different form. In a GNN, layers are associated with the order neighbor of that node. To simplify, for a node, the first layer would be the new embedding using the aggregate of its first order neighbors and the second layer would be that of its second order neighbors.

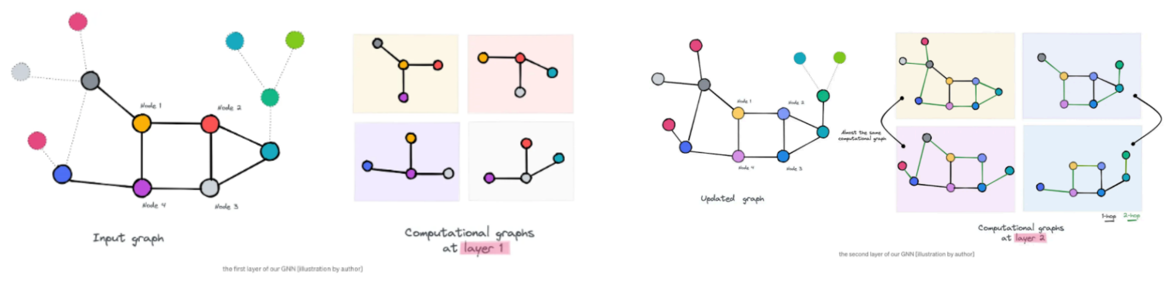

Node 1
Node 2
Node 4
Node 3
Input graph
Computational graphs at layer 1
the first layer of our GNN [illustration by author]
Node 1
Node 2
Node 4
Node 3
Updated graph
Almost the same computational graph
1-hop
2-hop
Computational graphs at layer 2
the second layer of our GNN [illustration by author]

Figure 1: The figure on the left shows how at one layer, the nodes used to aggregate for the new embedding have a recognizable difference and thus the embeddings will be different. The figure on the right shows how at two layers, we use more nodes and thus the nodes used to compute the new embeddings are very similar and will result in very similar embeddings. (AOMAR 2021)

Over-smoothing occurs when we set this number of layers to be too high and the embeddings of the nodes become very similar. At each increase in the number of layers, the number of neighbors increases exponentially so we aggregate more and more of the nodes and at a certain point, the embeddings cannot be differentiated.

This is a difficult challenge to face because while you do not want this case of over-smoothing to occur, you also want to learn from more nodes as there may be valuable information in the second or third hop neighbor of the node. Figure 2 shows a possible graph structure where more layers would be better because one or two layers would only be learning from that cluster of nodes and classify the new user to be in that cluster as well even though its features may say otherwise.

Another challenge that we must face when developing new GNN architectures is over-squashing. As we mentioned earlier, new embeddings involve the aggregation of neighboring nodes. This problem occurs when we aggregate information from a potentially large neighborhood of nodes into a single representation at a given node and distant nodes in the graph influence the computation at a particular node, leading to a situation where the representation becomes overly compressed. This large computation causes a bottleneck and loss of useful information and leads to difficulties in the model learning to differentiate between different parts of the graph.

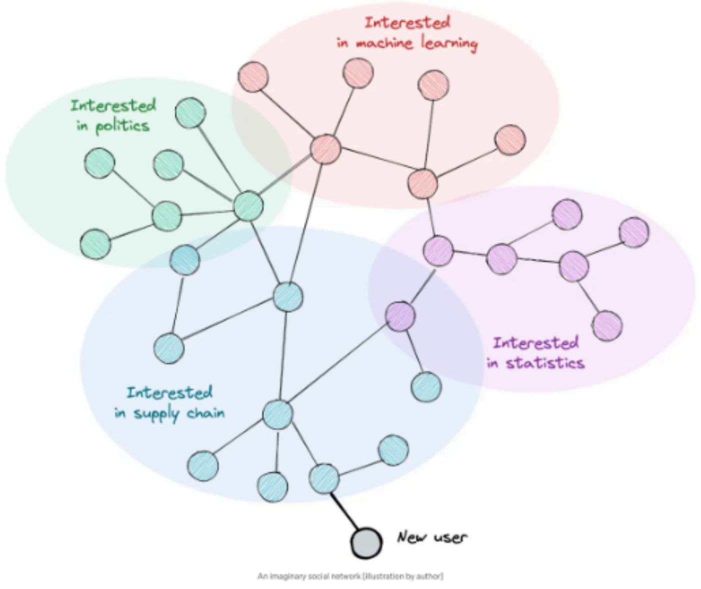

Interested in machine learning
Interested in politics
Interested in statistics
Interested in supply chain
New user
An imaginary social network [illustration by author]

Figure 2: Illustration shows how graphs with clusters could be at a disadvantage for GNN with few layers. (AOMAR 2021)

2.2 Long-Range Interactions

To alleviate the limitations of over-smoothing and over-squashing, various methods have been proposed as to how we can handle these *long-range interactions* (LRI). The development of *fully connected Graph Transformers* have gained popularity as it gives the ability to model long-range dependencies in the graphs and alleviate information bottleneck to some extent. (Dwivedi et al. 2023)

What we characterize long-range interactions as can depend on a few different factors. One factor that we should take into account is graph size or the number of nodes in a graph. As the number of nodes increases, a graph is more susceptible to over-squashing and there will be a visible effect on the information if a local message passing based GNN (MP-GNN) is used to perform graph level tasks. On the contrary, local MP-GNN would have the same success as any GT that handles LRI on graphs with smaller size and it would not be influenced by the information bottleneck. Thus, one condition of the LRI benchmark is that the graph sizes should be sufficiently large in order to separate the performance of local MP-GNN from those models which model LRIs. (Dwivedi et al. 2023) Another factor that characterizes LRI is the nature of the task. Depending on the task at hand, we may only need to look at sub-structures of the graph in which local MP-GNN can be just as effective. On the other hand, we may need to learn over the entirety of the graph in which this case, we would would categorize as a benchmark of LRI due to the nature of the task. The last factor that we must take into account is the contribution of the global graph structure to the task. In tasks where the global *positional encodings* (PE) is valuable as a source of information, local MP-GNN will fail to capture this information without some sort of influence from a bottleneck and thus handling LRI is necessary. A dataset where the learning task can benefit from the global PE can be considered as a benchmark for LRI. Additionally, if the task is dependent on some information of distance and graph features, we can also consider the dataset for LRI since the information regarding distance would require global structural information.

As we look at different models and their performance on different datasets, we would like to evaluate the model's handling of LRI. To do so, our dataset must qualify as a long-range dataset to which we will use one of the proposed *long-range graph benchmark* (LRGB) datasets by Dwivedi et al. (2023).

3 Methods

3.1 Graph Attention Networks (GAT)

Graph Convolutional Networks (GCN) have become one of the more standard applications of geometric deep learning. In short, for each node, we take its neighboring nodes and take some weighted aggregate to form a new embedding of the node. The message passed from each node can also be scaled by the number of neighboring nodes to quantify the importance of nodes with more neighbors but that is the most you can get in terms of quantifying any "stronger" relationships between two nodes. But what if a node's neighbor's features (not just the degree) has some importance to the node's new feature representation.

That is where we introduce the idea of *Graph Attention Networks* (GAT) which builds onto GCN by adding the concept of *self-attention mechanisms*. For a target node and each of its neighboring nodes, a learnable weight matrix and activation function are used to compute the attention coefficient for each pair of nodes. The attention coefficients are computed in a way that nodes which are important to the target node's features get higher weights. Unlike traditional GCN which uses predefined weights, GAT learns to assign different weights to different nodes in a neighborhood, allowing for a more flexible and powerful aggregation scheme.

The input to a single graph attention layer is a set of node features, $\boldsymbol{h} = \{\overrightarrow{h}_1, \overrightarrow{h}_2, \dots, \overrightarrow{h}_N\}, \overrightarrow{h}_i \in \mathbb{R}^F$ where N is the number of nodes, and F is the number of features in each node. (Veličković et al. 2018) The layer gives us an output which is the set of new node features, $\boldsymbol{h}' = \{\overrightarrow{h}'_1, \overrightarrow{h}'_2, \dots, \overrightarrow{h}'_N\}, \overrightarrow{h}'_i \in \mathbb{R}^{F'}$, which may or may not be in the same dimension. To obtain stronger expressivity, at least one learnable linear transformation is required so the input features can be converted into higher-level embeddings. The learnable linear transformation is parameterized by a *weight matrix*, $\boldsymbol{W} \in \mathbb{R}^{F' \times F}$, which is then applied to all nodes. The self-attention mechanism that differentiates GAT from GCN is then performed on the nodes-a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ computes *attention coefficients*

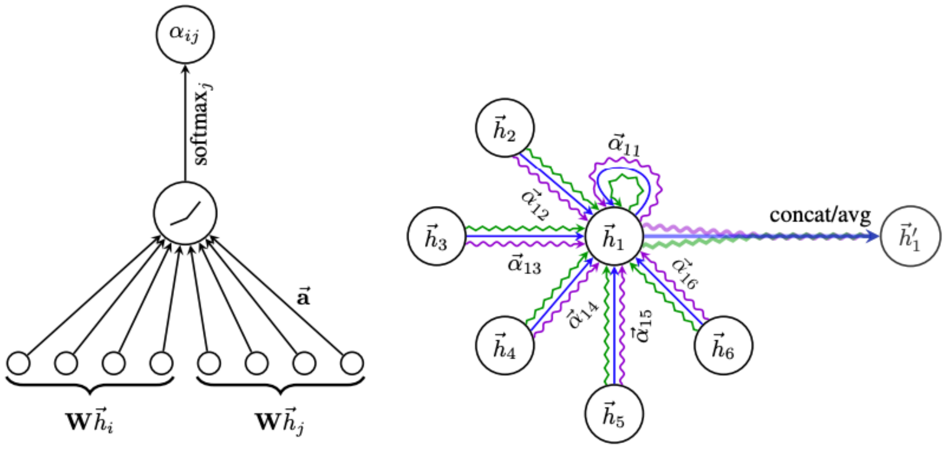


Figure 3: Illustration of Attention Mechanism used in GAT (Veličković et al. 2018)

$$e_{ij} = a(\boldsymbol{W}\overrightarrow{h}_i, \boldsymbol{W}\overrightarrow{h}_j)$$

which represents the importance of node j's features to node i. The model allows every node to attend on every other node, dropping all structural information. (Veličković et al. 2018) The structure of the graph is factored into the mechanism after performing masked attention which is where we only compute e_{ij} for nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is some neighborhood of node i in the graph. In most cases, these will be the first-order or second-order neighbors of i (including i) to minimize computational costs as the number of neighboring nodes grows exponentially. Coefficients are then normalized across all choice of j using the softmax function to make them easily comparable across different nodes:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}$$

After solving for the normalized attention coefficients, the linear combination of the features corresponding to them can be computed to obtain the new embedding of the node features. We could also apply a non-linearity function, σ:

$$\overrightarrow{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W}\overrightarrow{h}_j\right)$$

like *ReLU* or *Sigmoid* activation. Now we have our new and final output features for every node with more expressivity while maintaining some form of the information from the original node feature inputs. The aggregation process is illustrated in Figure 3.

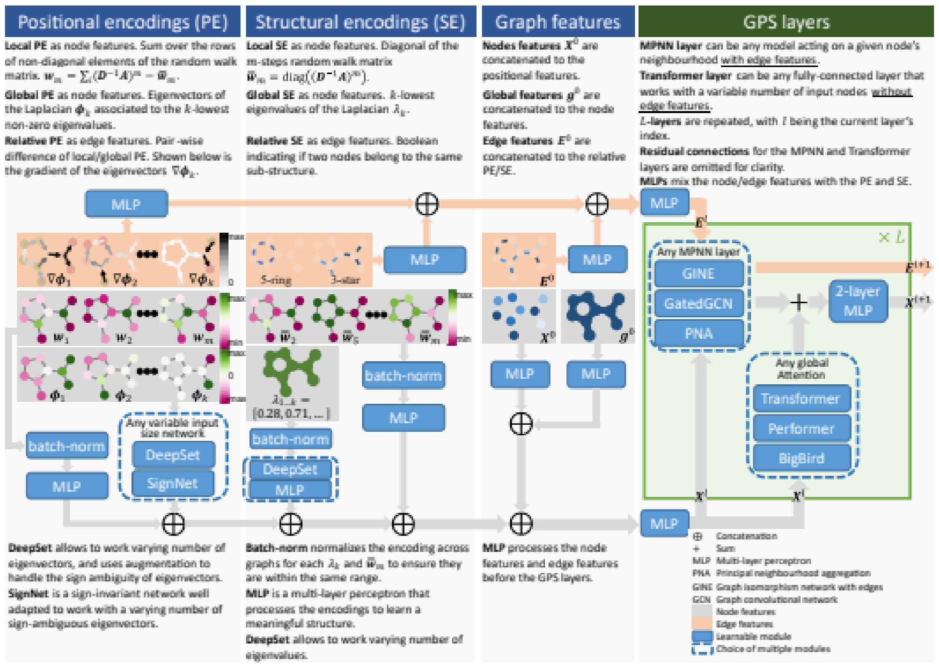
GAT are also very flexible in that they can be applied to graph nodes with varying degrees and do not require any transformation or pre-processing of the input graph structure. Many techniques used to deal with graph structured data also are built on the assumption that neighboring nodes share class labels. GAT are not built upon this assumption and can be applied to graphs that are not necessarily well-structured or regular. Their ability to specify different weights to different nodes allows them to handle graphs that may have been difficult to solve before where connected nodes may have different labels or roles.

3.2 General, Powerful, Scalable (GPS) Graph Transformers

The limitations of simple GNN models that were mentioned above, in over-smoothing and over-squashing, have been attempted to be alleviated with the use of *Graph Transformers* (GT). They do so by implementing the idea of *global attention* where all nodes are attended to in order to learn the *structural encodings* (SE) and *positional encodings* (PE) of all nodes. However, these methods require a fully-connected graph and thus new sets of limitations arise as the computational cost of a fully-connected graph transformer model is extremely costly with a complexity of $O(N^2)$ for a graph with N nodes and E edges. To alleviate these limitations, we use a *General, Powerful, Scalable* (GPS) *Graph Transformer* layer which combines a *message passing neural network* (MPNN) layer with a *transformer* layer while also leveraging different positional and structural encoding techniques to provide more *powerful* expressivity of the input graph and more *scalability* in the time complexity of the model. The *generalization* of the transformer is achieved through its flexible modularity. Different choices of the MPNN layer along with the global attention layer that combine to make the GPS layer can be made which makes this transformer model applicable to a wide range of graph structures and tasks. The initial PE and SE types, which will be elaborated on later, are also parts of the layer that can be chosen by the one training the model which adds even more elements of modularity and generalization.

Positional and structural encodings are integral components of all the various GTs that have been researched. PE are meant to give a representation of the nodes position in space by capturing the distance between nodes and its neighbors. On the other hand, SE are meant to give a representation of the structure of a graph or subgraph in order to capture similarity between in structure. GPS proposes a new way of representing the PE and SE by organizing both into 3 categories; *local, global* and *relative*. Next, we will elaborate on the key differences between the categorizations of encodings.

Local PE is encoded as a node feature and allows a given node to recognize its position and role within a nearby group of nodes. Within this group, the proximity of two nodes to one another influences the similarity of their respective local PE. An analogy of this concept is the position of a word within a sentence, rather than in the entire text. **Global PE** is also encoded as a node feature but differs in the sense that it allows a given node to recognize its global position within the graph rather than a local cluster. Within the graph, the proximity of two nodes to one another influences the similarity of their respective global PE. An analogy of this concept is the position of a word within the entire text and not just within the sentence. **Relative PE** is encoded as an edge feature unlike the previous represents the distances or directional relationships of two nodes. It is an embedding of edges that


Positional encodings (PE)
Structural encodings (SE)
Graph features
GPS layers
MLP
batch-norm
Any variable input size network
DeepSet
SignNet
5-ring
3-star
Any MPNN layer
GINE
GatedGCN
PNA
2-layer MLP
Any global Attention
Transformer
Performer
BigBird
× L
⊕ Concatenation
+ Sum
MLP Multi-layer perceptron
PNA Principal neighbourhood aggregation
GINE Graph isomorphism network with edges
GCN Graph convolutional network
Node features
Edge features
Learnable module
Choice of multiple modules

Local PE as node features. Sum over the rows of non-diagonal elements of the random walk matrix. $w_m = \sum_i (D^{-1}A)^m - \bar{w}_m$.
Global PE as node features. Eigenvectors of the Laplacian ϕ_k associated to the k-lowest non-zero eigenvalues.
Relative PE as edge features. Pair-wise difference of local/global PE. Shown below is the gradient of the eigenvectors $\nabla\phi_k$.

Local SE as node features. Diagonal of the m-steps random walk matrix $\bar{w}_m = \mathrm{diag}((D^{-1}A)^m)$.
Global SE as node features. k-lowest eigenvalues of the Laplacian λ_k.
Relative SE as edge features. Boolean indicating if two nodes belong to the same sub-structure.

Nodes features X^0 are concatenated to the positional features.
Global features g^0 are concatenated to the node features.
Edge features E^0 are concatenated to the relative PE/SE.

MPNN layer can be any model acting on a given node's neighbourhood with edge features.
Transformer layer can be any fully-connected layer that works with a variable number of input nodes without edge features.
L-layers are repeated, with l being the current layer's index.
Residual connections for the MPNN and Transformer layers are omitted for clarity.
MLPs mix the node/edge features with the PE and SE.

DeepSet allows to work varying number of eigenvectors, and uses augmentation to handle the sign ambiguity of eigenvectors.
SignNet is a sign-invariant network well adapted to work with a varying number of sign-ambiguous eigenvectors.

Batch-norm normalizes the encoding across graphs for each λ_k and $\bar{w}_m$ to ensure they are within the same range.
MLP is a multi-layer perceptron that processes the encodings to learn a meaningful structure.
DeepSet allows to work varying number of eigenvalues.

MLP processes the node features and edge features before the GPS layers.

Figure 4: General Recipe for a GPS graph Transformer (Rampášek et al. 2023)

is linked to the distance indicated by any PE, either global or local, similar to the distance observed between two words.

Local SE is encoded as a node feature and allows a node to recognize the substructures it belongs to. When considering a SE with a radius of m, the similarity of the local SE for two nodes increases as the resemblance of their respective m-hop subgraphs around them grows closer. **Global SE** is encoded as a graph feature and allows the network to recognize the global structure of the graph. As the similarity between two graphs increases, their global SE will become closer. **Relative SE** is encoded as an edge feature and measures the difference in the structure of two nodes. Furthermore, it is an embedding that is correlated to the difference of any local SE.

As seen in Figure 4, the PE and SE of the graph are a big component in the architecture of the transformer. The flexibility to choose different types of PE and SE give that much more complexity and generality that can change the expressivity of the transformer. Rampášek et al. (2023) focuses on only global PE, relative PE, and local SE in the GPS layers of their transformer model but you can see the potential and ability for innovation in the domain that arises with this new approach to embedding.

The GPS layer follows a simple updating procedure in which at a single layer, the features are updated by aggregating the output of the MPNN layer and global attention layer until you reach the number of layers you wish to have in your transformer model. As mentioned before, the MPNN layer and global attention layer is flexible in their design. Specifically, the MPNN can be any function that operates within a local neighborhood, and the global attention layer can be any type of fully-connected layer. The layer updating process goes as follows:

$$\boldsymbol{X}^{l+1}, \boldsymbol{E}^{l+1} = \texttt{GPS}^{l}(\boldsymbol{X}^{l}, \boldsymbol{E}^{l}, \boldsymbol{A}) \tag{1}$$
$$\boldsymbol{X}_M^{l+1}, \boldsymbol{E}^{l+1} = \texttt{MPNN}_e^{l}(\boldsymbol{X}^{l}, \boldsymbol{E}^{l}, \boldsymbol{A}) \tag{2}$$
$$\boldsymbol{X}_T^{l+1} = \texttt{GlobalAttn}^{l}(\boldsymbol{X}^{l}) \tag{3}$$
$$\boldsymbol{X}^{l+1} = \texttt{MLP}^{l}(\boldsymbol{X}_M^{l+1}, \boldsymbol{X}_T^{l+1}) \tag{4}$$

where $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of a graph with N nodes and E edges; $\boldsymbol{X}^l \in \mathbb{R}^{N \times d_l}$, $\boldsymbol{E}^l \in \mathbb{R}^{E \times d_l}$ are the d_l-dimensional node and edge features respectively. (Rampášek et al. 2023) These contain the structural and positional encodings of choice and are aggregated in the following processes. Lastly, $\texttt{MPNN}_e^l$ and $\texttt{GlobalAttn}^l$ are instances of an MPNN with edge features and of a global attention mechanism at the l-th layer with their corresponding learnable parameters, respectively; $\texttt{MLP}^l$ is a 2-layer MLP block. (Rampášek et al. 2023)

As shown above, GPS provides a brand new set of potential in the world of geometric deep learning. As the name suggests, the **generality** is shown through the immense flexibility of the transformer architecture and its ability to generalize to a multitude of graph structures. The **power** is shown through its potential expressivity with a different approach to the embeddings of node and edge features while also alleviating the limitations of simple GNNs that are over-smoothing and over-squashing by allowing information to spread across the graph via full-connectivity. (Rampášek et al. 2023) Lastly, the **scalability** is shown through its computational complexity of $O(N+E)$ which is an improvement from most graph transformers that have quadratic time complexity. In the following sections, we will evaluate GPS and compare the results to those of simple GNN models.

4 Datasets

Table 1: Statistics of the Datasets Used

Dataset	**Task**	**# Graphs**	**Avg Nodes**	**Avg Edges**	**Avg Diameter**	**# Class**
`Cora`	Transductive	1	2,708	10,556	3.9	7
`PascalVOC-SP`	Transductive	8,498	479.25	2,709.57	5.65	21
`IMDB`	Inductive	1,000	19.77	193.06	9.76	2
`Enzyme`	Inductive	600	32.63	124.27	3.81	6

4.1 Cora

The Cora dataset represents a citation network in which documents are represented by nodes and citation links are represented by (undirected) edges. The task at hand is transductive which means we aim to perform node classification by predicting one of seven classes that each document is labeled. The dataset consists of only one graph where each node/document contains a sparse *bag-of-words* feature vector which will be used to perform classification. We use this dataset because it has a significantly smaller shortest path length and diameter so there is no need to consider the handling of LRIs. We will experiment with different models and their ability to perform node classification on this type of smaller range data.

4.2 PascalVOC-SP

The PascalVOC-SP dataset consists of multiple graphs that represent different images from the original Pascal VOC image dataset. This dataset is different in that super pixels are extracted from the original image and represented as nodes which are then connected by edges to create a rag-boundary graph representing the original image. The task at hand is transductive which means we aim to perform node classification by predicting one of 21 semantic segmentation labels for each superpixel node in each graph. This dataset was constructed to extract a minimum of 500 superpixels for each graph in order to satisfy the 'graph size' characteristic of the LRGB mentioned in Dwivedi et al. (2023). We use this dataset in order to compare the capabilities of handling LRI between various models of GNNs we experiment with.

4.3 IMDB-BINARY

The IMDB-BINARY dataset consists of multiple graphs that each represent a movie. Within each graph, individual actors are represented by nodes and edges represent collaborations between entities from other movies. The task at hand is inductive which means we aim to perform graph classification. We will do so by using the links between entities in other movies to predict the genre of a movie which is a binary variable, action or romance. We use this dataset in order to compare the performance of graph classification of various models as opposed to the node classification we had been doing previously. Different models will perform better or worse based on the task at hand so the results obtained from this dataset will be used to compare results from other datasets with different tasks.

4.4 Enzyme

The Enzyme dataset consists of multiple graphs that represent different protein structures. Within each graph, nodes represent secondary structure elements and edges represent the interactions and spatial closeness between these elements. The task at hand is inductive which means we aim to perform graph classification by predicting the catalyzed chemical reaction of each graph which can be reflected by one of the 6 EC top-level classes. We use this dataset as another way of comparing to compare model performance for graph classification and to show the strengh and weaknesses of different models based on the given task.

5 Experiments

5.1 Experimental Setup

We will train and test four different GNN architectures (GCN, GIN, GATv2, graphGPS) on all the datasets described above. We split each dataset into train and test sets and evaluate the accuracy of our models on both sets due to the nature of the task being classification. The `IMDB` dataset was downloaded from the `Pytorch Geometric` library but contained zero node features. Because we need the models to learn some set of node features for classification, we pre-processed the graphs to include degrees for each node as a node feature. The other datasets were also downloaded from the same library but contained node features so pre-processing was not necessary. We will optimize all of our models using the Adam optimizer with a learning rate of 0.01 and a weight decay of 0.0005. Lastly, 200 epochs will be run to train the models on each dataset with the exception of `PascalVOC-SP` in which we will run 5 epochs due to the sheer number of graphs contained in the dataset.

Our GCN models consist of two `GCNConv` layers from the `Pytorch Geometric` library with the input being the number of node features of the graph and the output being the number of class labels. We will use ReLU activation and drop out for our hidden layers and a softmax for the output layer. For our datasets where we must perform graph classification, we will incorporate batching of size 32 and use global mean pooling for each batch before going into the linear output layer which is then softmaxed.

Our GAT models consist of two `GATv2` layers from the `Pytorch Geometric` library with the input being the number of node features of the graph and the output being the number of class labels. Additionally, we will use 8 heads to incorporate the self attention mechanism that differentiates GAT architectures from simple GNN architectures. We will use ReLU activation and drop out for our hidden layers and a softmax for the output layer. For our datasets where we must perform graph classification, we will incorporate batching of size 32 and use global mean pooling for each batch before going into the linear output layer which is then softmaxed.

Our GIN models consist of two `GINConv` layers from the `Pytorch Geometric` library with the input being the number of node features of the graph and the output being the number of class labels. For our models on the `Cora`, `IMDB`, and `Enzyme` datasets, both of the `GINConv` layers include a sequential layer which in itself has a linear layer with ReLU activation followed by another linear layer. After going through these two layers, For our model on the `PascalVOC-SP` dataset, both of the `GINConv` layers contain multi-layer perceptron layers instead of a sequential layer. All of our models will use ReLU activation and drop out for the feed forward process with the output layer being softmaxed. For our datasets where we must perform graph classification, we will incorporate batching of size 32 and use global mean pooling for each batch before going into the linear output layer which is then softmaxed.

Our graphGPS models consists of three `GPSConv` layers from the `Pytorch Geometric` library with each layer using GAT as its message passing mechanism and ReLU as its activation function. Lastly, one more GAT layer is added as the output layer with the same size as the number of classes in the dataset. For the feed forward process, ReLU activation and drop out will be used on the first three GPS layers and the output layer will be softmaxed to give us our one prediction label. Due to the size of the `PascalVOC-SP` dataset, we will incorporate batching of size 32 and use global mean pooling for each batch before going into the linear output layer which is then softmaxed.

5.2 Results and Analysis

Table 2: **Left**: Accuracies for *Transductive* Tasks **Right**: Accuracies for *Inductive* Tasks

	`Cora`		`PascalVOC-SP`	
Method	**Train**	**Test**	**Train**	**Test**
GCN	100%	78.9%	69.96%	69.53%
GIN	100%	72.8%	69.95%	69.43%
GATv2	100%	**81.5%**	69.89%	69.53%
graphGPS	85.71%	63.4%	68.75%	**80.13%**

	`IMDB`		`Enzyme`	
Method	**Train**	**Test**	**Train**	**Test**
GCN	66.4%	71.8%	81%	76.67%
GIN	73.1%	72.3%	96%	**94.17%**
GATv2	65.7%	50%	67.17%	70.67%
graphGPS	75.7%	**75%**	70.17%	75.67%

The results of our models on each dataset can be seen in the table above. Our GAT models seem to perform best on the `Cora` dataset which is reasonable because it is not necessary to handle LRI for this dataset due to the size and structure of each graph. Thus, a simple GNN architecture like GAT can perform as well or even better as a graphGPS model. On the contrary graphGPS performs the best on the `PascalVOC-SP` dataset. The large discrepancy in the performance of graphGPS compared to other architectures is due to its ability to handle LRI especially on a dataset that is characterized as a LRGB. As for the `IMDB` dataset, graphGPS saw the best performance but there was not as big of difference in performance compared to simple GNN architectures like GIN. This dataset consists of smaller graphs that are not characterized as a LRGB. Additionally, the task of this dataset involves graph classification which is where GIN is most powerful. We believe these two factors may be the reason for such a small gap in performance between the models but we also believe that incorporating different structural and positional encodings into our graphGPS model will increase the performance significantly. Lastly, for the `Enzyme` dataset, our GIN model saw the highest performance in terms of accuracy. Like `IMDB`, `Enzyme` also contains graphs of smaller size which is why we believe the handling of LRI was not necessary and a simple architecture like GIN could produce exceptional results. Our results show the strengths of graphGPS when the structure and task of the dataset call for it but we can also see the strengths of simple architectures like GAT and GIN on datasets where the handling of LRI is not necessary.

6 Conclusion

In this analysis, we have dived deeper into the sophisticated domain of *Graph Neural Networks* (GNN), unraveling the intricacies of *Graph Attention Networks* (GAT) and the *General, Powerful, Scalable* (GPS) *Graph Transformer* framework. Our exploration of GAT has reaffirmed its significance in enhancing node representation through the incorporation of attention mechanisms, allowing for dynamic weighting of node contributions and fostering more nuanced graph representations. The GPS framework, on the other hand, has emerged as a formidable approach in addressing the challenges when it comes to handling long-range interactions while simultaneously alleviating limitations of scalability and expressiveness that persist in traditional GNN architectures.

Through empirical analysis, we have showcased the prowess of GAT in achieving state-of-the-art performance across various benchmarks, particularly when the task necessitates discernment of fine-grained structural information. Meanwhile, the GPS model's novel blend of message-passing and global attention layers has yielded remarkable advancements in handling larger and more complex graphs, demonstrating its potential in capturing global structural features without compromising computational efficiency.

Our analysis underscores the potential of these advanced GNN architectures in a multitude of practical applications, from social network analysis to bioinformatics. While GAT stands out in scenarios where local context and node-specific relationships are key, the GPS model excels in global structure recognition and largescale graph processing. The adaptability and robustness of these models pave the way for future research directions, including the integration of GNN with other deep learning domains and the exploration of unsupervised learning frameworks within the graph domain.

In conclusion, the advancements in GAT and GPS models have not only enriched the graph-based machine learning landscape but also set a foundation for the ongoing pursuit of more sophisticated, efficient, and versatile graph neural network architectures. The field of geometric deep learning is advancing rapidly, holding promise of transformative breakthroughs in fields where understanding the underlying graph structure is critical.

References

AOMAR, Anas AIT. 2021. “Over-Smoothing Issue in Graph Neural Network.” https://towardsdatascience.com/over-smoothing-issue-in-graph-neural-network-bddc8fbc2472

Dwivedi, Vijay Prakash, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2023. “Long Range Graph Benchmark.” https://arxiv.org/abs/2206.08164

Rampášek, Ladislav, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2023. “Recipe for a General, Powerful, Scalable Graph Transformer.” https://arxiv.org/abs/2205.12454

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. “Graph Attention Networks.” https://arxiv.org/abs/1710.10903