Fraud detection with Graph Attention Networks

Project Part-01 & Part-02

Part-01: Understanding/Summarizing the data you will be using for the project utilizing some visualization techniques you have already learned.

Introduction to Fraud Detection:

The collection of procedures and investigations known as fraud detection enables companies to spot and stop the unlawful financial activity. This can include unauthorized use of credit cards, identity theft, computer hacking, insurance fraud, and other issues. When someone steals money or other assets from you by dishonesty or illegal action, it is called fraud.

Therefore, if the fraud is ongoing, having a good fraud detection system can assist institutions spot suspicious activity or accounts and reduce losses.

The fraud detection model based on ML algorithms is challenging for a number of reasons, including the fact that fraud accounts for a relatively small portion of all daily transactions, that its distribution changes quickly over time, and that the true transaction label will not be available for several days because investigators were unable to promptly review all of the transactions.

However, because most data science models ignore something really crucial like network topology, standard approaches of machine learning continue to fall short of spotting fraud.

Similar to social networks, fraud detection calls for the use of a graph's functionality. An example of a graph transactions network is shown in the following graphic, where we can see some nodes like a bank account, a credit card, and a person along with their connections:

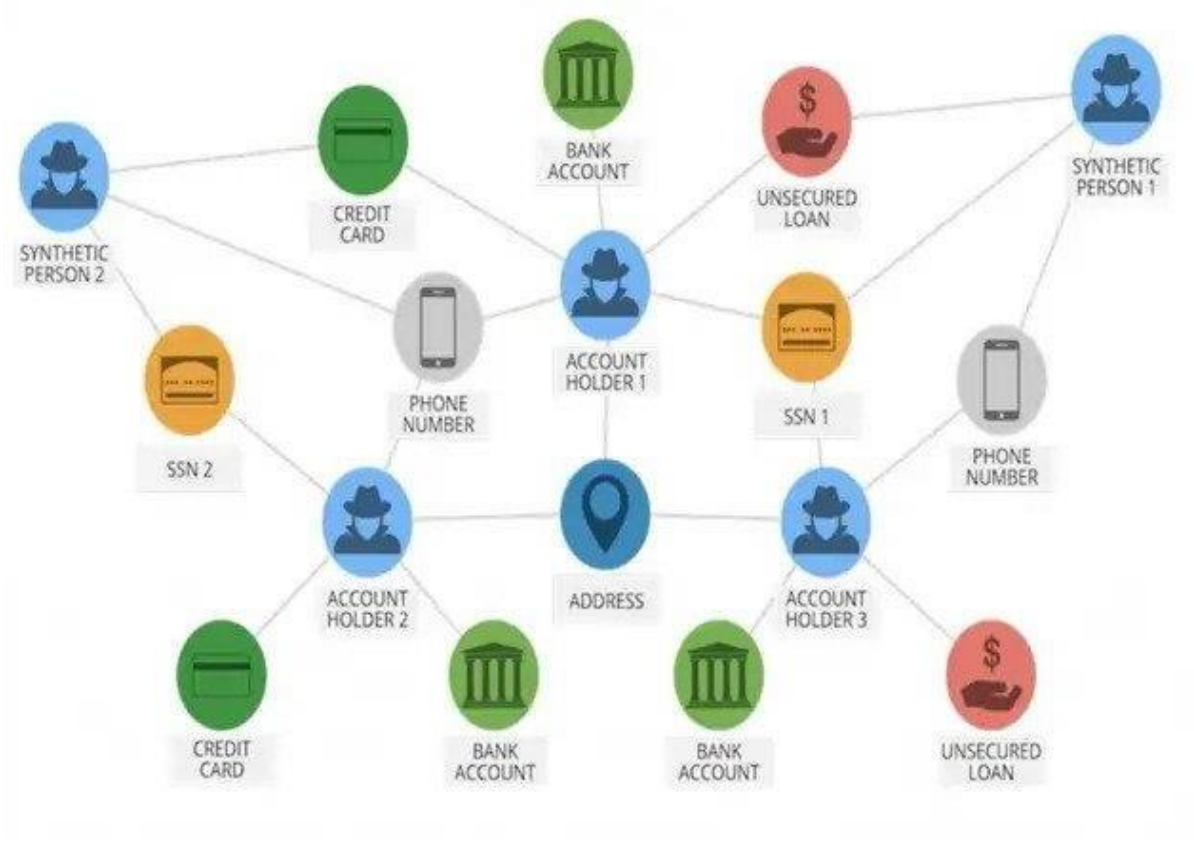


Fig: Transactions graph example.

By incorporating more pertinent feature information from the network, graph machine learning increases the accuracy of fraud predictions.

We are utilizing the Bitcoin dataset, which associates real-world objects with either legal or illegal Bitcoin transactions.

This consists of three CSV files: the first one labels nodes (licit/illicit/unknown), the second describes edges between nodes using their "Ids," and the third contains node ids with 166 attributes.

A transaction is represented as a node in the graph, and a movement of bitcoins between transactions is represented by an edge. Each node has been classified as having either 166 traits and being the work of a "licit," "illicit," or "unknown" entity.

After downloading the datasets from the kaggle, we are applying some Data Visualization techniques which we have already learned in class to understand and summarize the data, and also to draw some valuable insights from the data.

Data transformations:

We will load the three CSV files into pandas data frames when you have downloaded them. We additionally connect the features and class data frames and convert the classes from names to integers as described below.

Unknown: 2, Illicit: 1, and Licit: 0.

After reforming the classes into numerical values, it looks as shown in the below tables.

	txId	class
0	230425980	2
1	5530458	2
2	232022460	2
3	232438397	0
4	230460314	2

	txId	class
203764	173077460	2
203765	158577750	2
203766	158375402	1
203767	158654197	2
203768	157597225	2

Table: classes converted into numerical values.

Data Visualizations:

In order to help people, comprehend and make sense of massive volumes of data, data visualization is a technique that makes use of a variety of static and dynamic visualizations within a given context. In order to visualize patterns, trends, and connections that could otherwise go missing, the data is sometimes presented in a story format.

We are using a bar plot to summarize the data that was converted to numerical. A bar plot, often known as a bar chart, is a graph that uses rectangular bars with lengths and heights proportionate to the values they represent to depict a category of data.

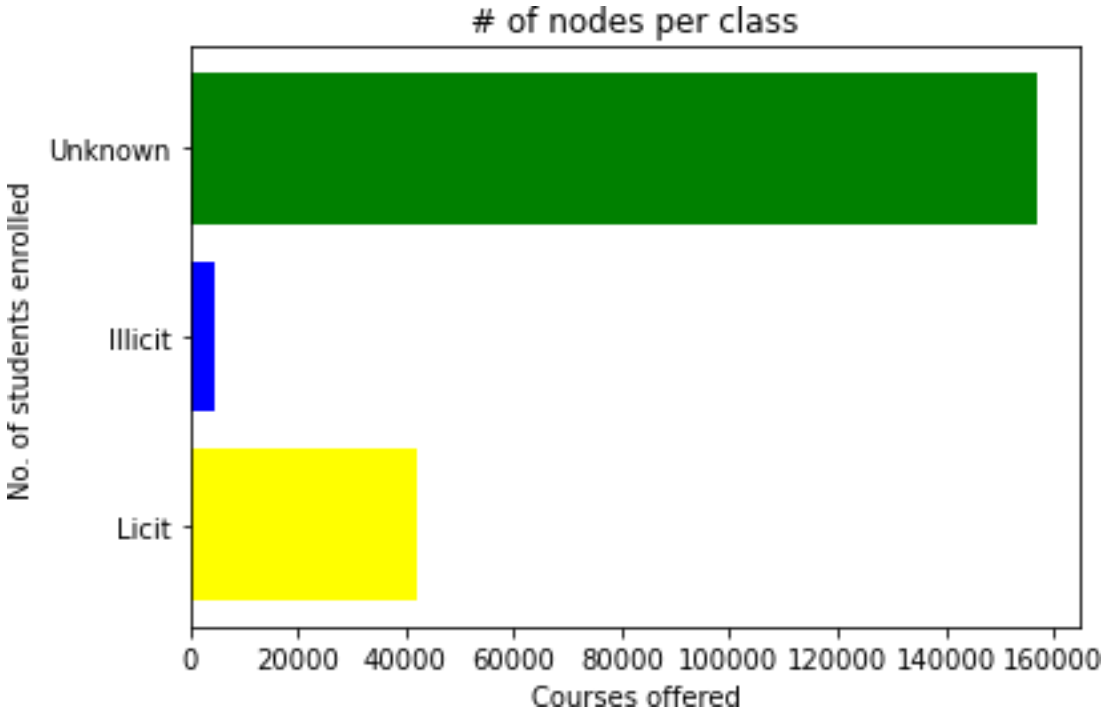


Fig: Visualization of numbers of nodes per class.

```
df_classes['class'].value_counts()
```

```
2    157205
0     42019
1      4545
Name: class, dtype: int64
```

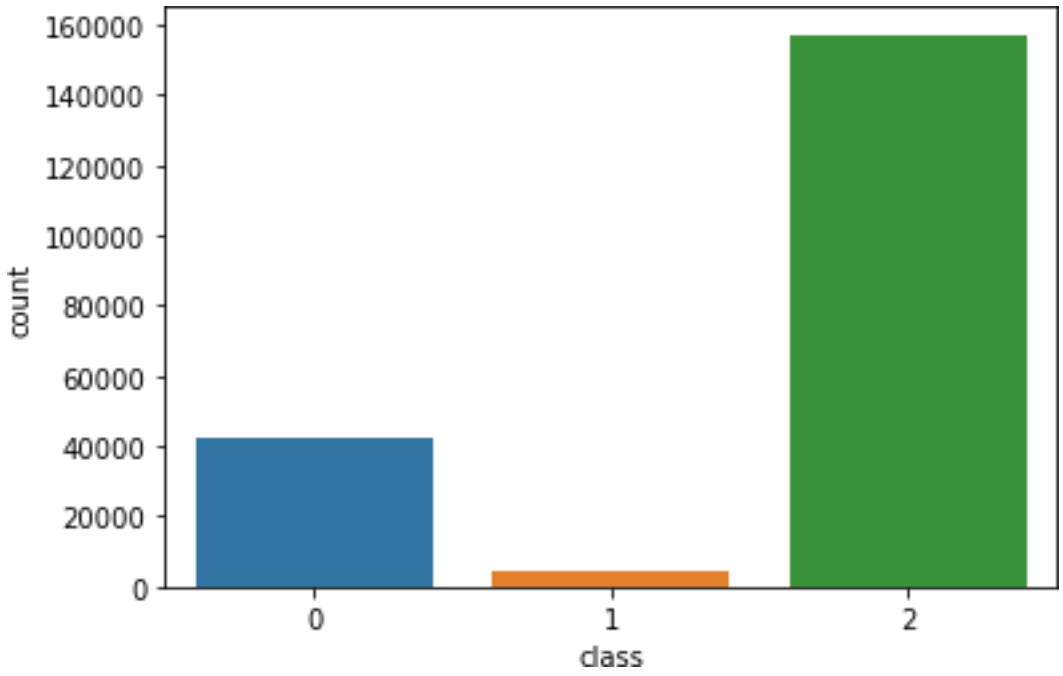
From the above Bar plot and statistics, we can summarize the below points.

we have two percent which is 4545 of the nodes that are labelled class1 (illicit), and twenty-one percent which is 42,019 are labelled class0 (licit) as we can see in Figure. And the remaining transactions of seventy-seven percent are not labelled with regard to licit versus illicit.

Unknown transactions 77% - 157205

Illicit transactions 21% - 42019

Licit transactions 2% - 4545



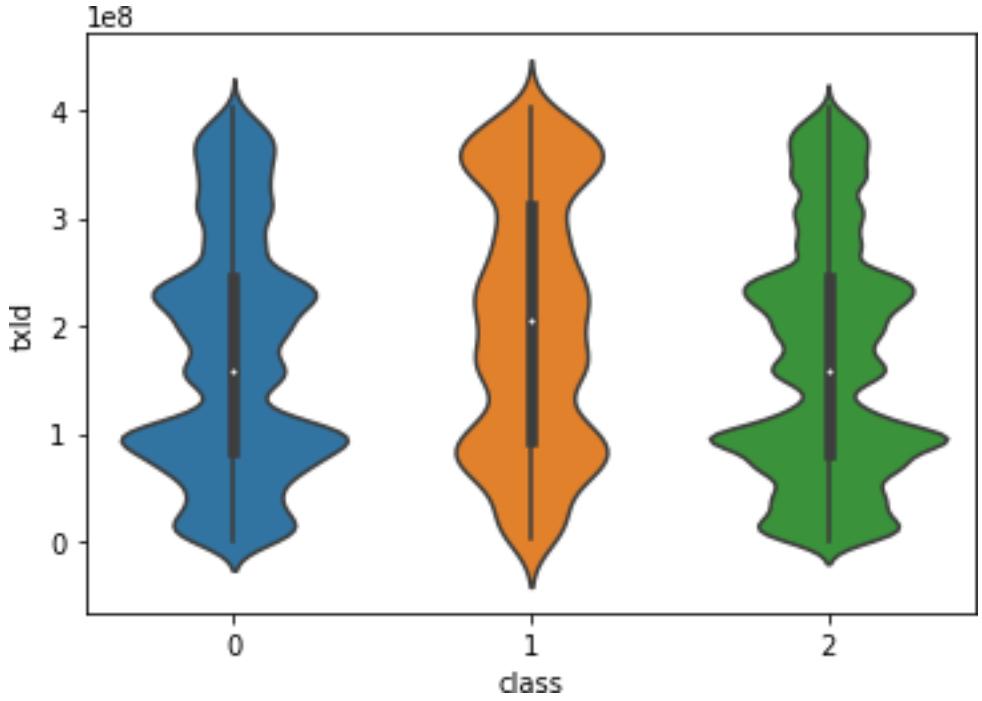


Fig: Violin plot of Class Vs Txn ID

From the above figure, we can see that violin plot for class 0 and class 2 is very similar with similar density distribution, median, lower quartile, upper quartile, maxima and minima. Median and upper quartile of class 1 is higher than class 0 and 2. But, all the classes have similar lower quartile and min.

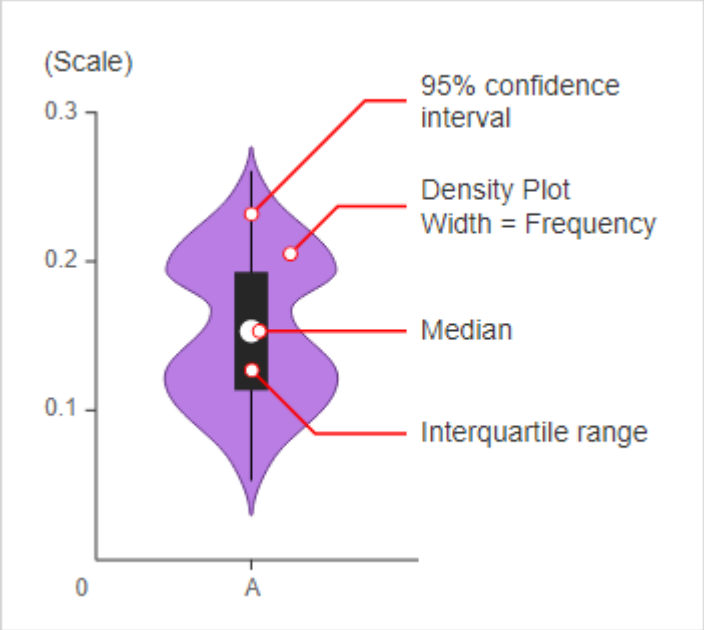


Fig: Understanding the Violin Plot

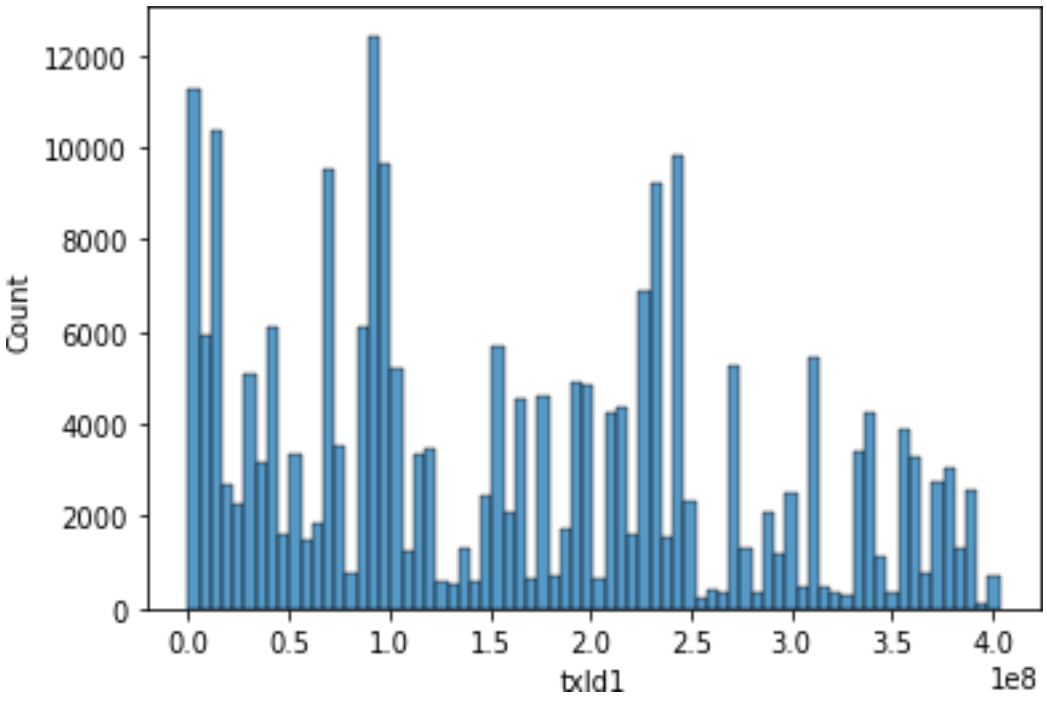


Fig: Understanding the histograms for TxId1

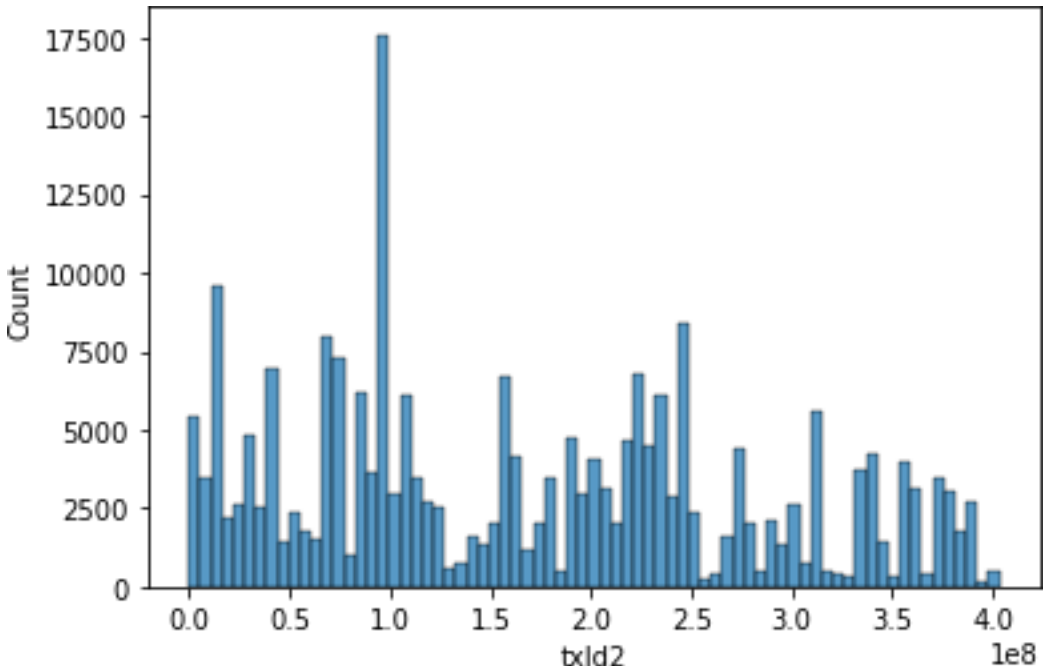


Fig: Understanding the histograms for TxId2

	txId1	txId2
txId1	1.000000	0.746848
txId2	0.746848	1.000000

Correlation:

In the given data, a change in one independent variable should have no impact on any other independent variables. However, when independent variables are correlated, it means that shifts in one variable are related to changes in the other.

Density plots:

Density plots are another quick and simple method for obtaining the distribution of each feature. Similar to a histogram, except that each bin's top has a curved line drawn through it. They are what are known as abstracted histograms.

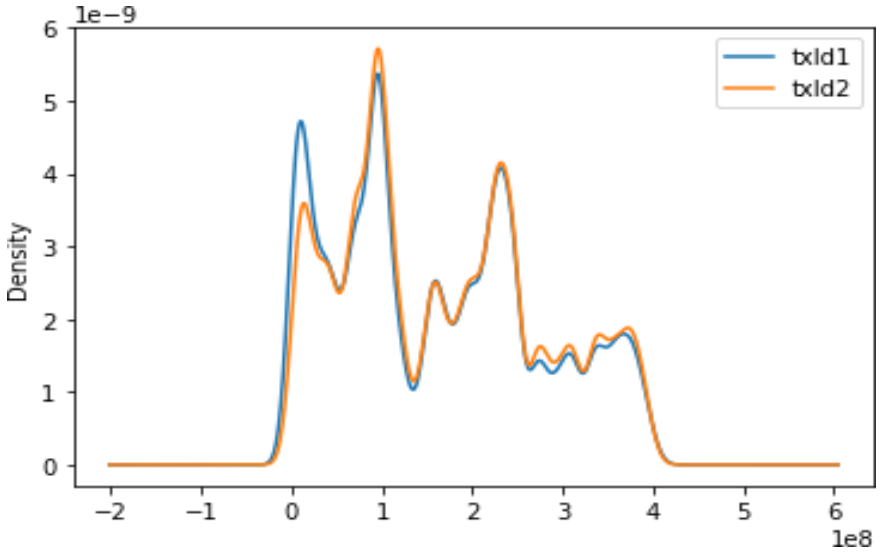


Fig: Density Plot

The above plot will show the distribution of attributes in the transactions.

Part-02: Brief description of the algorithm used or you will be using.

GRAPH ATTENTION NETWORKS:

One of the most well-known types of graph neural networks is Graph Attention Networks (GATs).

A Graph Attention Network (GAT) is a neural network design that uses masked self-attentional layers to overcome the drawbacks of earlier techniques based on graph convolutions or their approximations. It operates on graph-structured input.

Instead of calculating static weights based on node degrees like Graph Convolutional Networks (GCNs), they assign dynamic weights to node features through a process called self-attention. The main idea behind Graph Attention Networks is that some neighbors are more important than others, regardless of their node degrees.

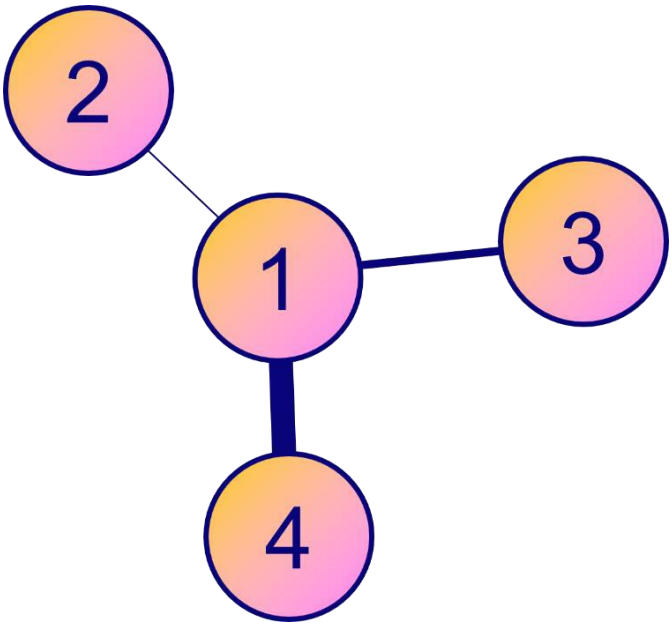
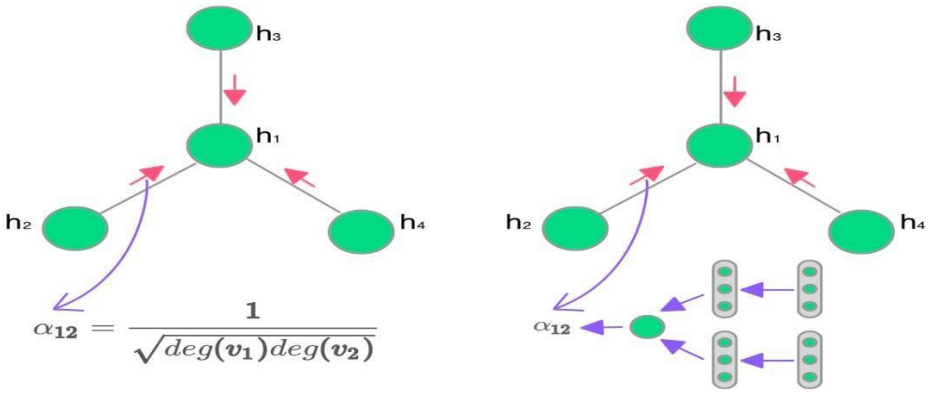


Fig: Node 4 is more important than node 3, which is more important than node 2.

A Graph Attention Network enables (implicitly) specifying different weights to different nodes in a neighborhood without necessitating any type of expensive matrix operation (such as inversion) or relying on prior knowledge of the graph structure by stacking layers in which nodes are able to attend over their neighborhoods' features.

A more understandable model in terms of the significance of neighbors results from the analysis and visualization of the learned attentional weights.



GCN explicitly assigns non-parametric weight $\alpha_{ij} = \frac{1}{\sqrt{deg(v_i)deg(v_j)}}$, via the normalization function during neighborhood aggregation.

GAT implicitly captures the weight α_{ij}, via the attention mechanism, so that more important nodes receive higher weight during neighborhood aggregation.

In place of the statically normalized convolution process, Graph Attention Network provides the attention technique. The crucial distinction is amply demonstrated in the graphic above.

The main idea behind GAT is to compute that coefficient implicitly rather than explicitly as GCNs do. That way we can use more information besides the graph structure to determine each node's "importance".

The authors behind GAT proposed that the coefficient, from now on denoted as a_{ij}*aij*, should be computed based on node features, which are then passed into an attention function. Note that edge features can also be included. Finally, the softmax function is applied in the attention weights a_{ij}*aij* to that result in a probability distribution.

A few important notes of Graph Attention Networks:

- GATs are agnostic to the choice of the attention function.
- The coefficient does not depend on the graph structure. Only on the node representations.
- GATs are fairly computationally efficient.
- The work can be extended to include edge features as well.
- They are quite scalable.

Submitted by:

1) Rohith Yamsani – 6364594
2) Sachin Sravan Kumar Komati - 6384991