# GNNExplainer

## 1. Introduction

GNNExplainer was first introduced in november 2019 in paper “GNNExplainer: Generating Explanations for Graph Neural Networks”. This method allows us to explain singular decisions of the model (but not the model itself) by identifying the most influential nodes, edges and features that lead the model to that decision. It handles single- as well as multi-instance explanations. It can be used to obtain explanations for any graph-based machine learning task, but is mostly used for explaining node classification and link prediction tasks.

## 2. Simple explanation

Explanation provided by GNNExplainer is a subgraph of the entire graph that the network was trained on, so that the subgraph maximizes the mutual information with GNN predictions. In the process a graph mask and a feature mask are created. Graph mask selects the important subgraph of the computation graph, while the feature mask masks out unimportant node features.

To describe the process of explanation, let's start with a simple graph and look at its computational graph.

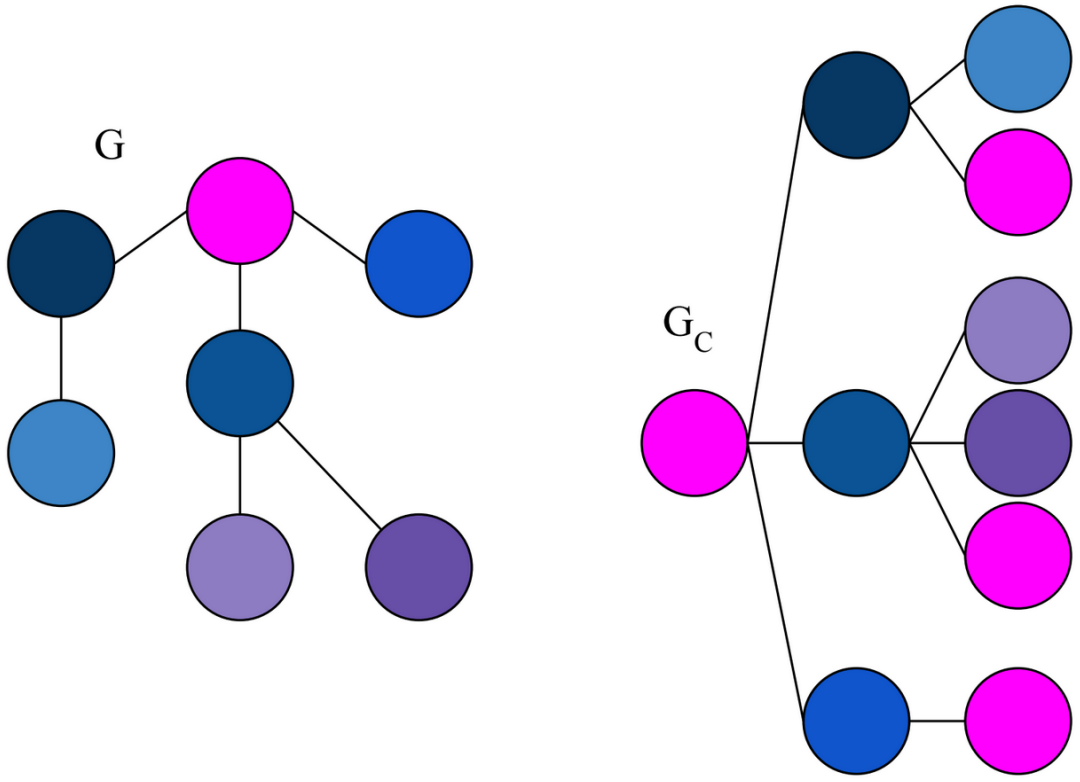


*Fig 1: A graph (on the left) and its corresponding computational graph for a task of GNN prediction on a pink node.*

Algorithm removes nodes and node features from the computation graph (creating $G_S$, computation subgraph) and looks at how the model prediction changes. By removing nodes, a node mask is created. By removing features, a feature mask is created.

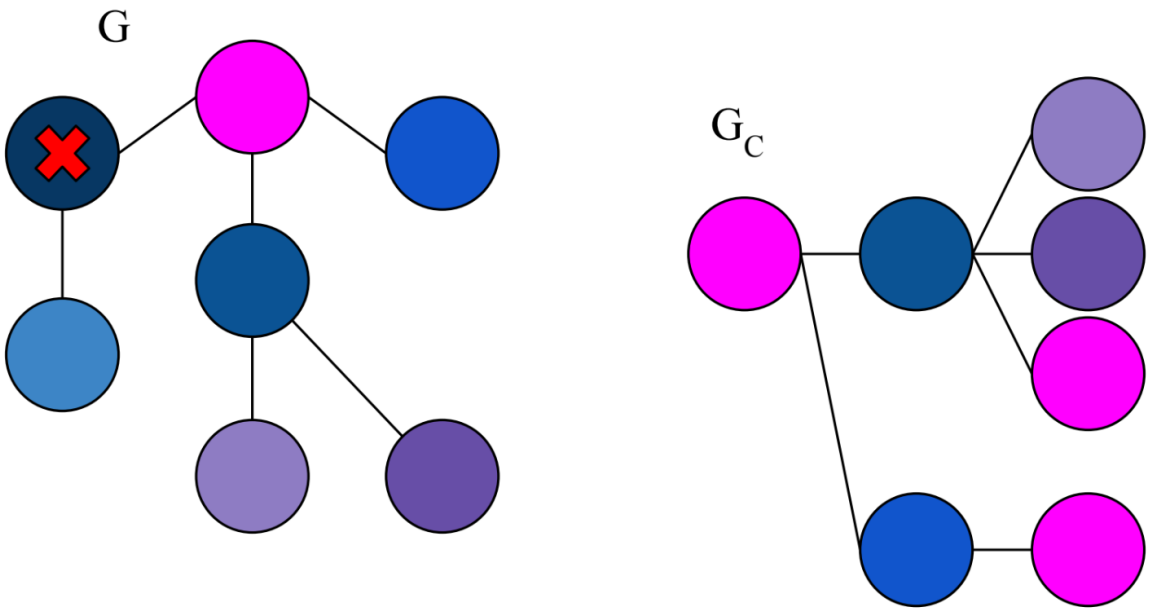


*Fig 2: Computation subgraph obtained by removing the dark blue node.*

If removing a node or a feature does not impact the model prediction, then it means that this node or feature is irrelevant for the GNN in the process of making a prediction. However, if the prediction changes dramatically - then it means that this node or this feature has a big impact on model prediction for that node.

It is computationally inefficient to check all possible permutations of computation graph $G_C$ subgraphs $G_S$ and all possible permutations of node features subsets $X_S$, so instead we use continuous masks.

# 3. Formulation

When using method to explain prediction on node $v$, the goal is to find the subgraph $G_S \subseteq G_C$ and to find the most influential features $X_S = \{x_j | v_j \in G_S\}$ for the prediction $\hat{y}$. Importance is formalized using mutual information $MI$ and GNNExplainer can be described as optimization of:

$$max\, MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S)$$

For node $v$, $MI$ measures the change in the probability of prediction $\hat{y} = \Phi(G_c, X_c)$ when the computation graph is limited to $G_S$, and node features are limited to $X_S$.

For a trained GNN model $H(Y)$ is constant, so in fact the problem is finding the minimum of equation $H(Y|G = G_S, X = X_S)$. GNNExplainer computationally efficient version is optimized using gradient

descent. For a deeper look into the math behind this method, I would advise to have a look at the original paper “GNNExplainer: Generating Explanations for Graph Neural Networks”.

# 4. Example

To visualize effects of GNNExplainer, we created a simple two-layer GCN network for a node prediction task on CORA dataset. This is a citation network dataset, consisting of 2708 nodes representing scientific publications in the field of ML, classified into one of seven classes (Theory, Reinforcement Learning, Genetic Algorithms, Neural Networks, Probabilistic Methods, Case Based, Rule Learning). Node features are embedding of keywords.

Model was trained for 200 epochs and achieved accuracy of 0.81 on the test set. Then we used GNNExplainer to explain predictions on one node of index 200 - its true class is 3 (Neural Network), and so is the class predicted by model on that node. We used GNNExplainer with 200 epochs, obtaining the following results:

a) Computation subgraph

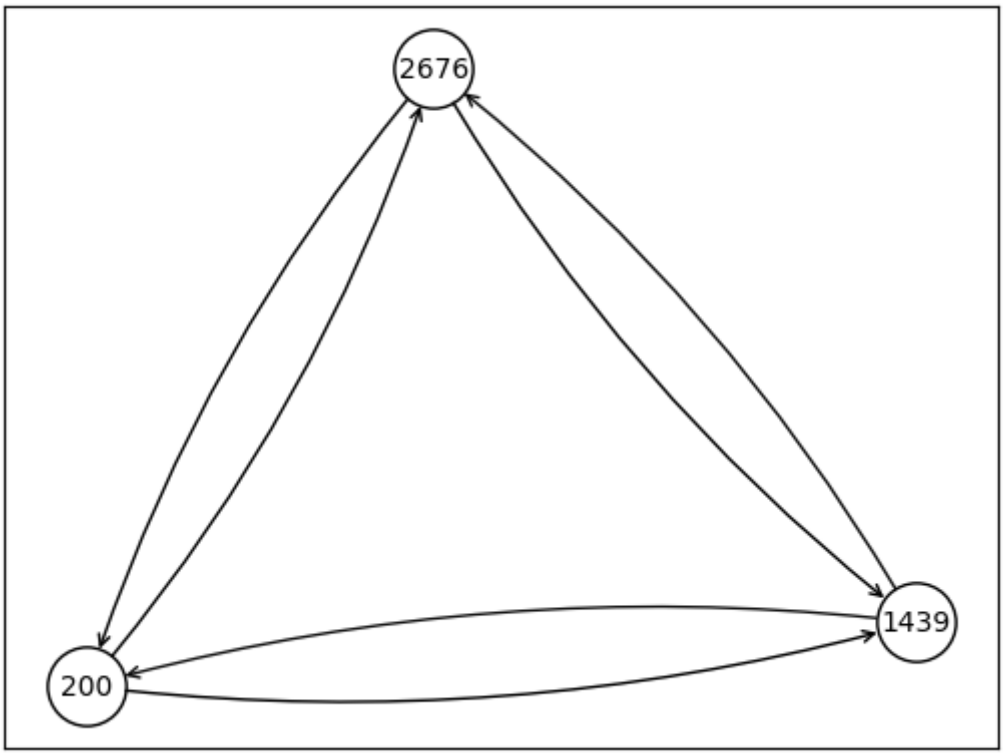


*Fig 3: Computation subgraph for node 200.*

Numbers represent node indexes. Nodes 200, 1439 and 2676 all share the same class (Neural Network).

b) Feature importance - unfortunately, we couldn’t find a dictionary of keywords for CORA dataset. However, to check if nodes of the same class share the same keywords, we checked other nodes from class 3.

Feature importance for node 200 (explained node):

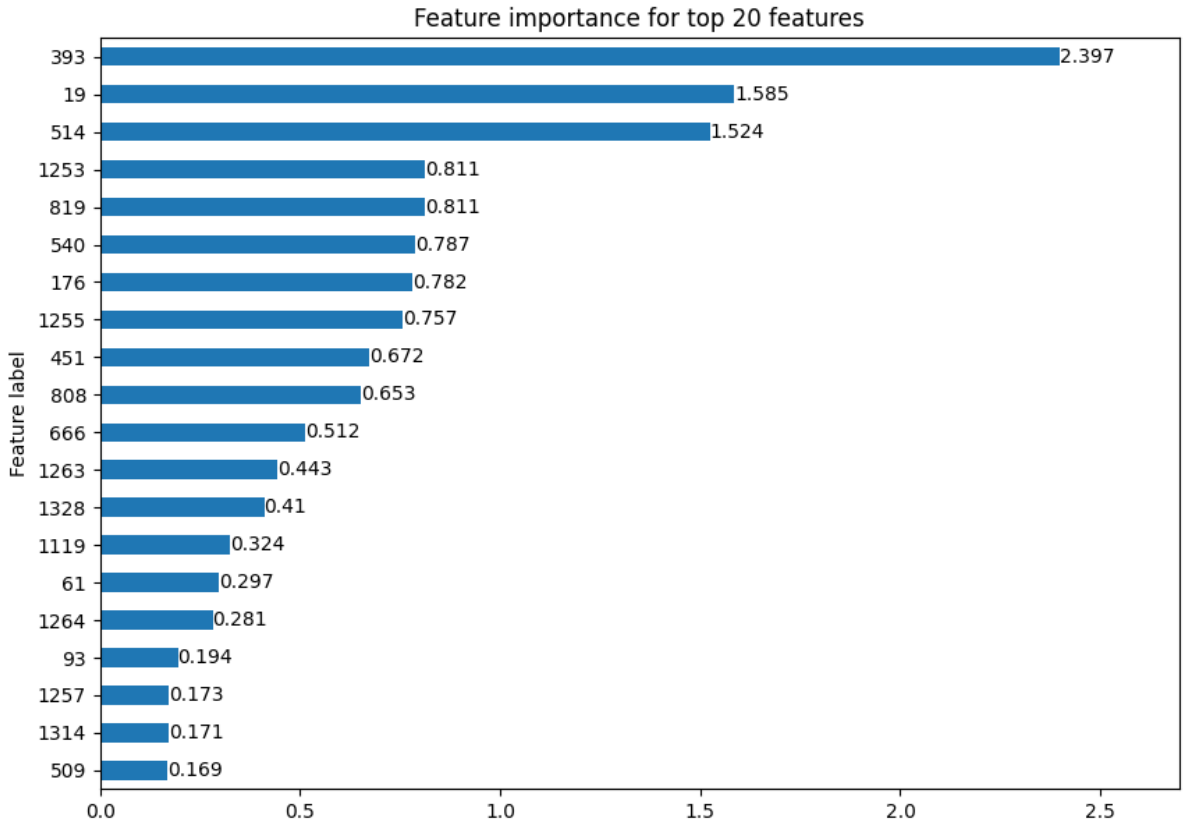


*Fig 4: Feature importance for node 200.*

Feature importance for node 1439 (neighbor of node 200, same class as node 200):

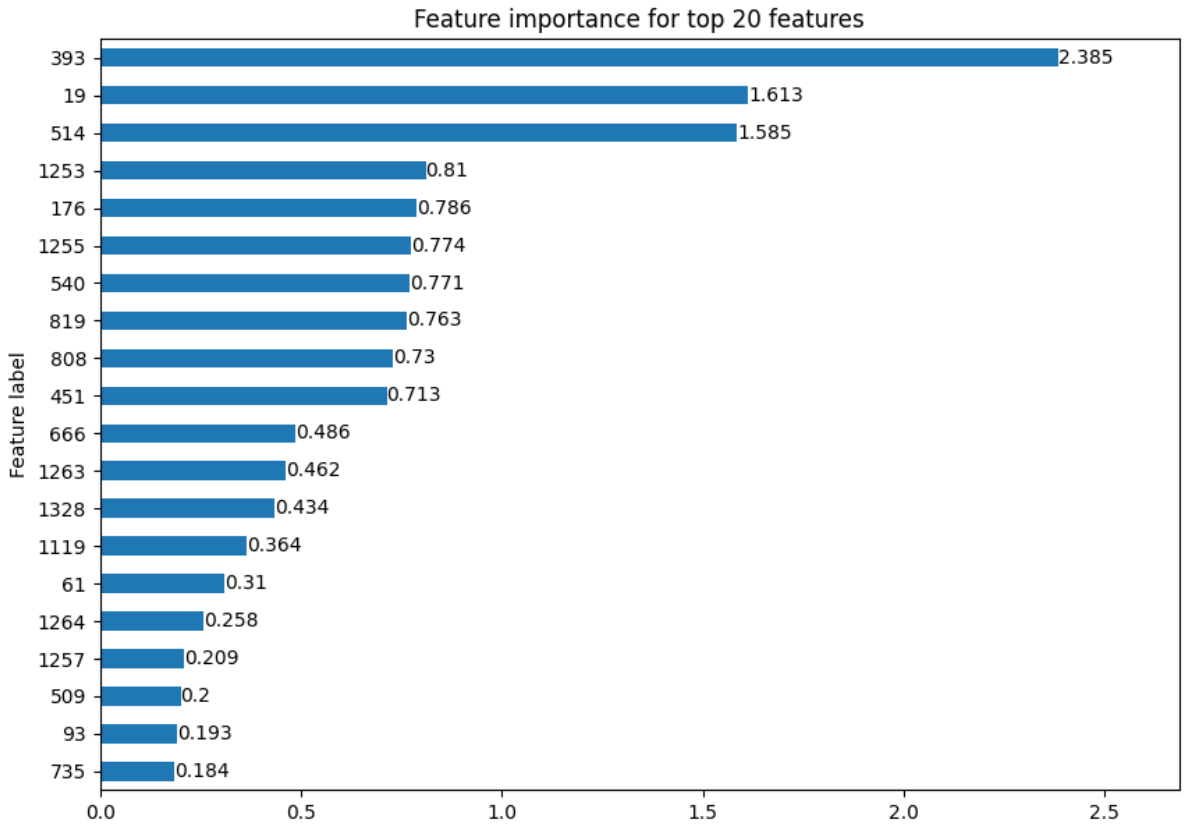


*Fig 5: Feature importance for node 1439.*

We can see that several features are marked as important for both predictions.
Feature importance for node 918 (not a neighbor of node 200, same class as node 200):

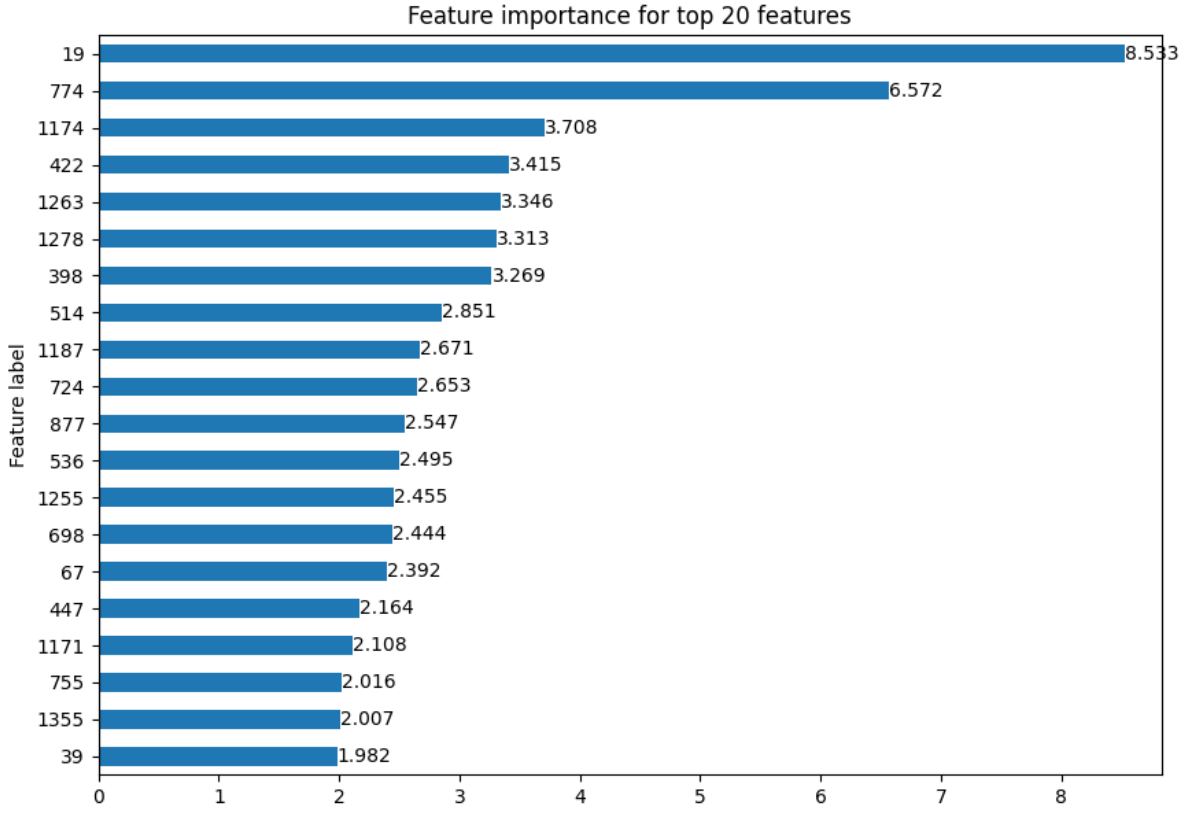

*Fig 6: Feature importance for node 918.*

Here on the other hand, both papers share only 4 same words in their top 20 important features. In all those examples feature number 19 is quite high (top 1/2), so it could be a common word for papers in the field of neural networks, like “network”, “neural” or “neuron” (or something else).

# 5. References


- R. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec: GNNExplainer: Generating Explanations for Graph Neural Networks, 2019
- T. N. Kipf, M. Welling: Semi-Supervised Classification with Graph Convolutional Networks, 2016
- P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad: Collective Classification in Network Data, 2008