AML Project Documentation

Overview and comparison of XAI methods dedicated for GNN networks.

Adam Łaba, Jakub Kraśniak

1. Topic

Our goal was to explore XAI methods suited for explaining predictions of graph neural networks. This included creating descriptions and simple code examples for several different explainers, we also prepared descriptions of metrics specifically suited for evaluating explanations of GNN models, that we later utilized in a final part of the project which was comparison of explanations obtained by different explainers (also including classic explainers like SHAP or Saliency).

2. Tools

Whole project was implemented in Python. Libraries that we used consist of:

- PyTorch [1],
- PyG (PyTorch Geometric) [2],
- Captum [3],
- Matplotlib [4],
- NetworkX [5].

3. Explainers

List of explainers for GNN networks that we used in the project:

- **GNNExplainer** [6], a method that utilizes the mutual information to compute the importance of input graph entities (nodes/node features/edges) for the model prediction,
- **PGExplainer** [7] which computes the importance of input graph structure (nodes/edges) for the model prediction using reparametrization,
- **PGMExplainer** [8] that creates Bayesian network explaining model prediction (please note that this explainer was described but not used in the final comparison),
- **GraphMaskExplainer** [9] that trains classifier for each explanation to find unimportant edges in graph,

- **Attention Explainer** which uses attention coefficients generated by a network in order to explain its prediction.

Others explainers used in final comparison:

- **InputXGradient** [10], a method that computes the attribution by multiplying the gradient with respect to input,
- **Saliency** [11] which returns the gradients with respect to inputs,
- **SHAP** [12] that uses Shapley Values from game theory.

4. Metrics

To evaluate explanations we will use several metrics specifically designed for explanations of GNN models predictions. These metrics include:

- **Fidelity** (phenomenon) [13]:

$$fid_{+} = \frac{1}{N}\sum_{i=1}^{N} |\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{G_{C\setminus S}} = y_i)| \tag{1}$$

$$fid_{-} = \frac{1}{N}\sum_{i=1}^{N} |\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{G_S} = y_i)| \tag{2}$$

Where y_i is the original prediction of graph i, N is the number of graphs, $\hat{y}_i$ is the prediction made by the gnn, $\hat{y}_i^{G_{C/S}}$ is prediction made by the gnn on graph without the explanatory subgraph, $\hat{y}_i^{G_S}$ is prediction made by the gnn on the explanatory subgraph and $\mathbb{1}$ is the indicator function.

- **Characterization score** [13]:

$$charact = \frac{w_{+} + w_{-}}{\frac{w_{+}}{fid_{+}} + \frac{w_{-}}{1 - fid_{-}}} \tag{3}$$

Where w are weights that should be picked from range (0, 1). Original paper used a value of 0.5 for both weights and so did we.

- **Unfaithfulness** [14]:

$$GEF(y, \hat{y}) = 1 - e^{(-KL(y||\hat{y}))} \tag{4}$$

Where y is prediction probability vector obtained by processing the original graph through net, $\hat{y}$ is prediction probability vector obtained from the masked subgraph, and KL is the Kullback-Leibler divergence score that quantifies the distance between the two probability distributions.

For discrete probability distributions P and Q defined on the same sample space, where X is the relative entropy from$Q$ to P, KL divergence is defined as:

$$KL(P||Q) = \sum_{x \in X} P(x) log(\frac{P(x)}{Q(x)}) \qquad (5)$$

5. Datasets

CORA [15] is a citation network dataset, consisting of 2708 nodes representing scientific publications in the field of machine learning, 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. Edge between nodes means citation.

Minesweeper [16] dataset is a minesweeper game field transformed into a graph. Each node is a single square, containing encoded information of the number of bombs around that square (50% of nodes) and another 50% of nodes is unknown. Each node has 8 edges to its neighbors. This is a binary classification task to determine if this square is a bomb or not.

MNIST SuperPixel [17] is a dataset of MNIST images transferred into the SuperPixel graph. Each graph contains 75 nodes, and the task is the same as on regular MNIST dataset.

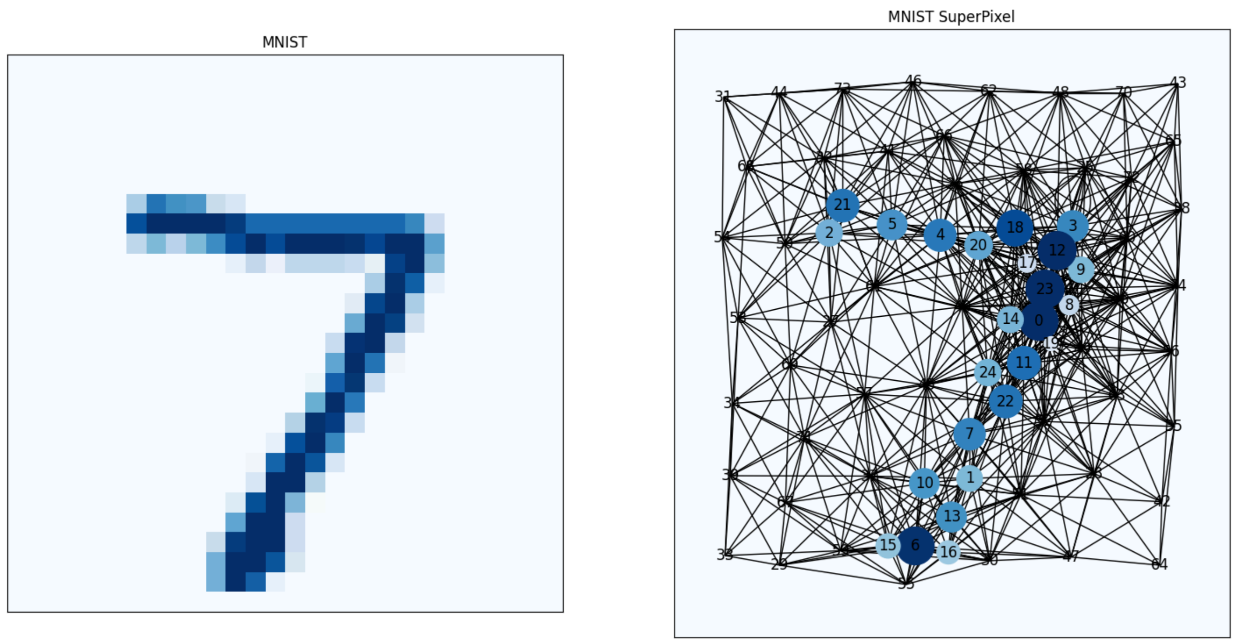

Fig 1: MNIST and MNIST SuperPixel.

6. GNN architectures

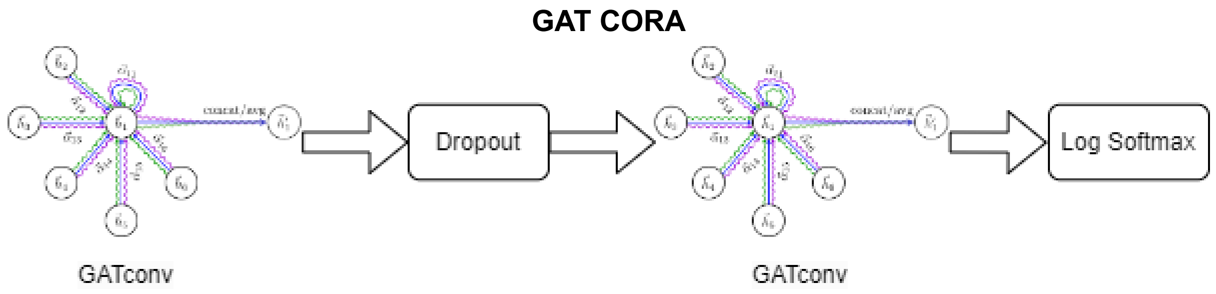

Fig 2: CORA GAT architecture.

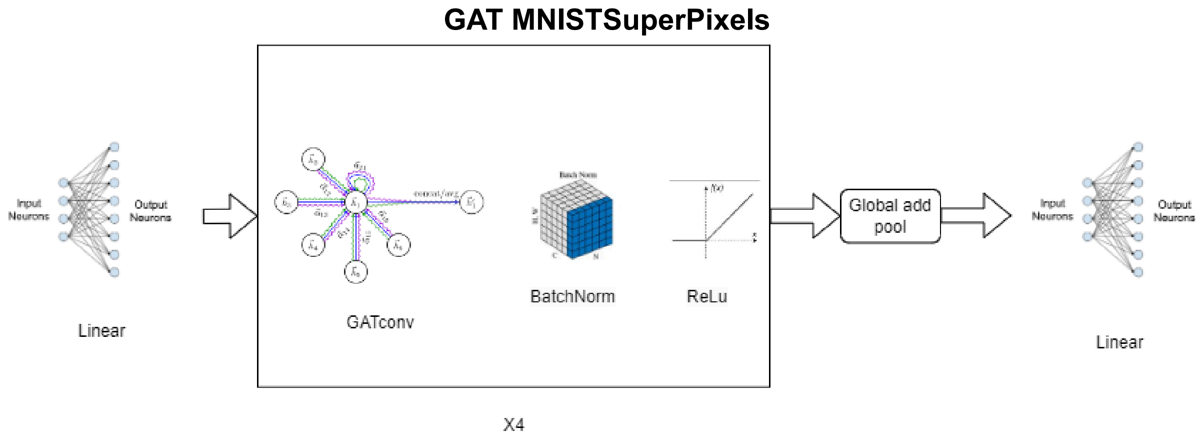

Fig 3: MNIST SP GAT architecture.

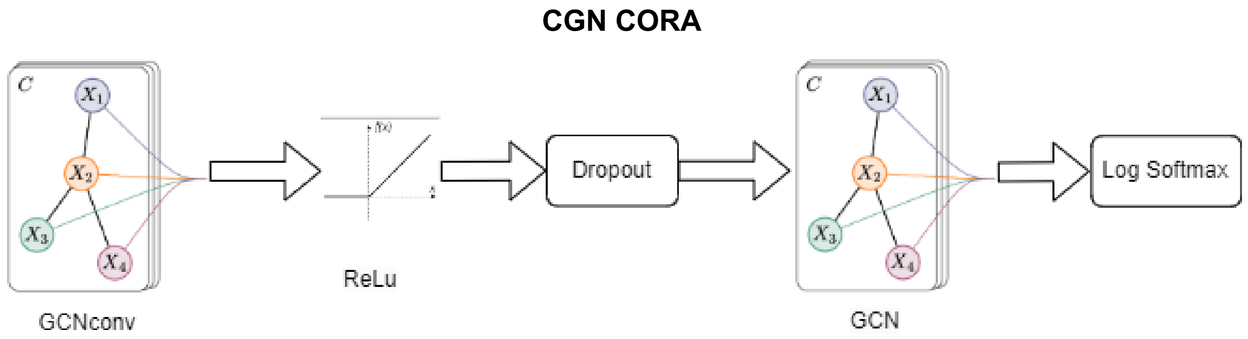

Fig 4: CORA GCN architecture.

GCN MNISTSuperPixels

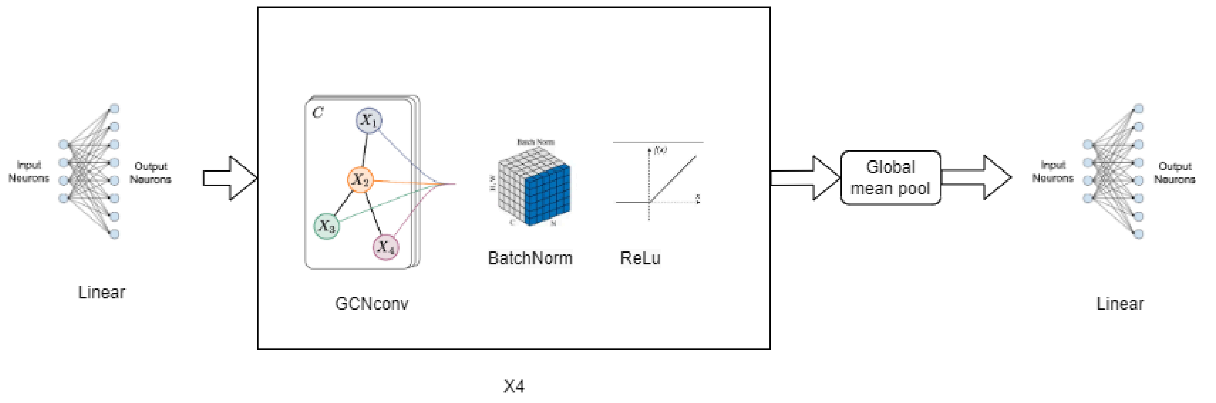

Fig 5: MNIST SP GCN architecture.

7. Results

CORA (edge-mask explanations)

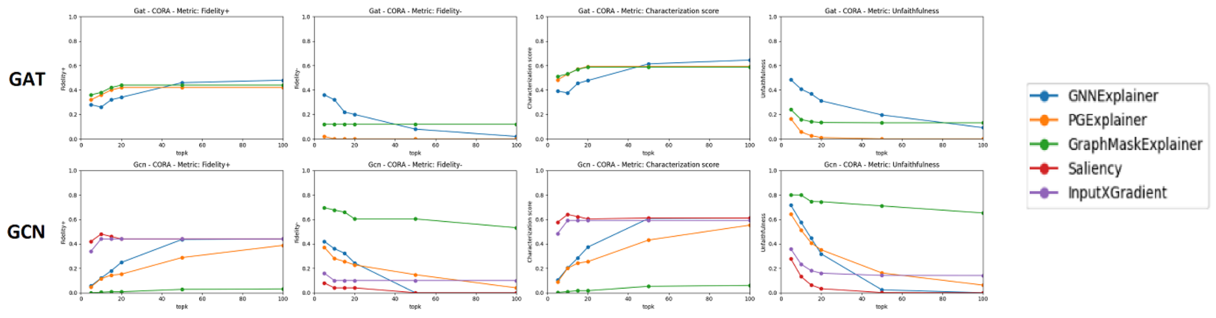

Fig 6: CORA edge-mask explanations results.

MNIST SuperPixel (edge-mask explanations)

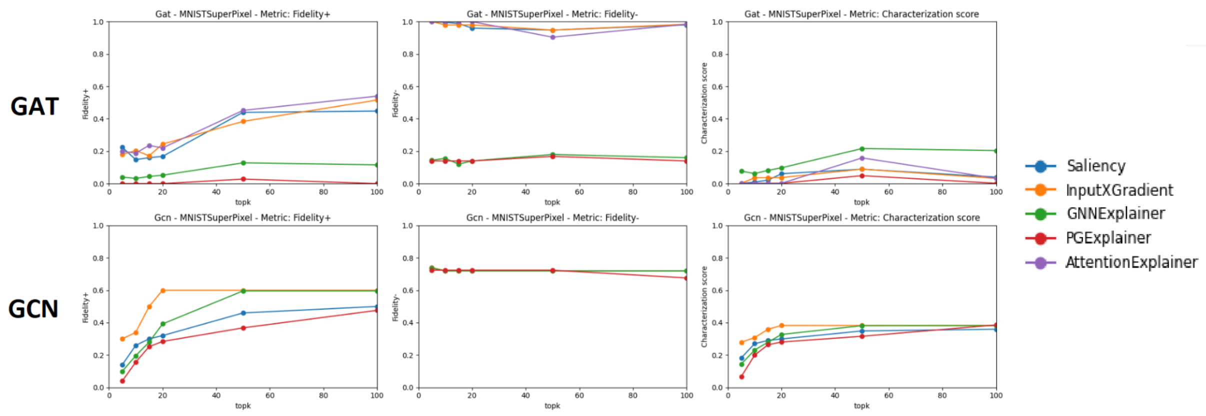

Fig 7: MNIST SP edge-mask explanations results.

MNIST SuperPixel (node-mask explanations)

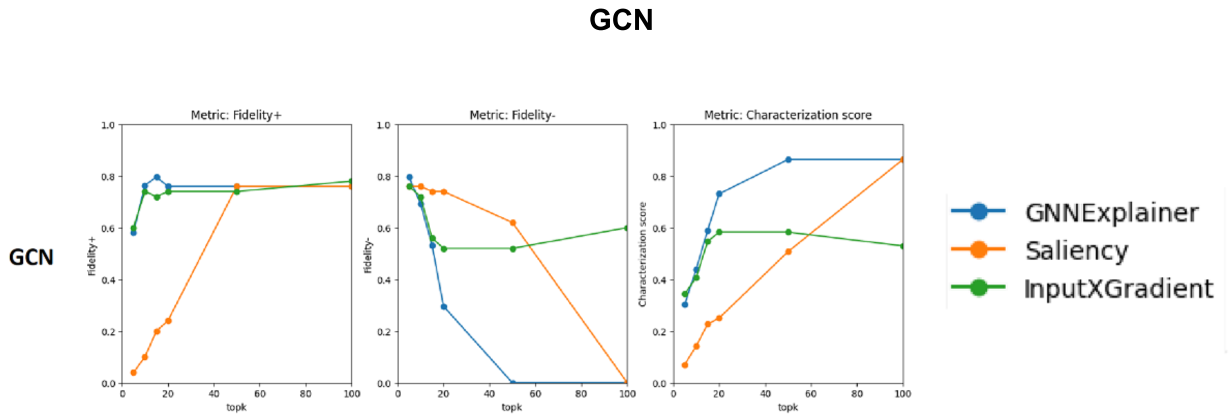

Fig 8: MNIST SP node-mask explanations results.

GCN - most influential nodes

We also visualized which nodes different explainers labeled as the most influential for the network prediction (each time selecting top 10 influential nodes for that explanation). This helps to evaluate some explanations for human observers.

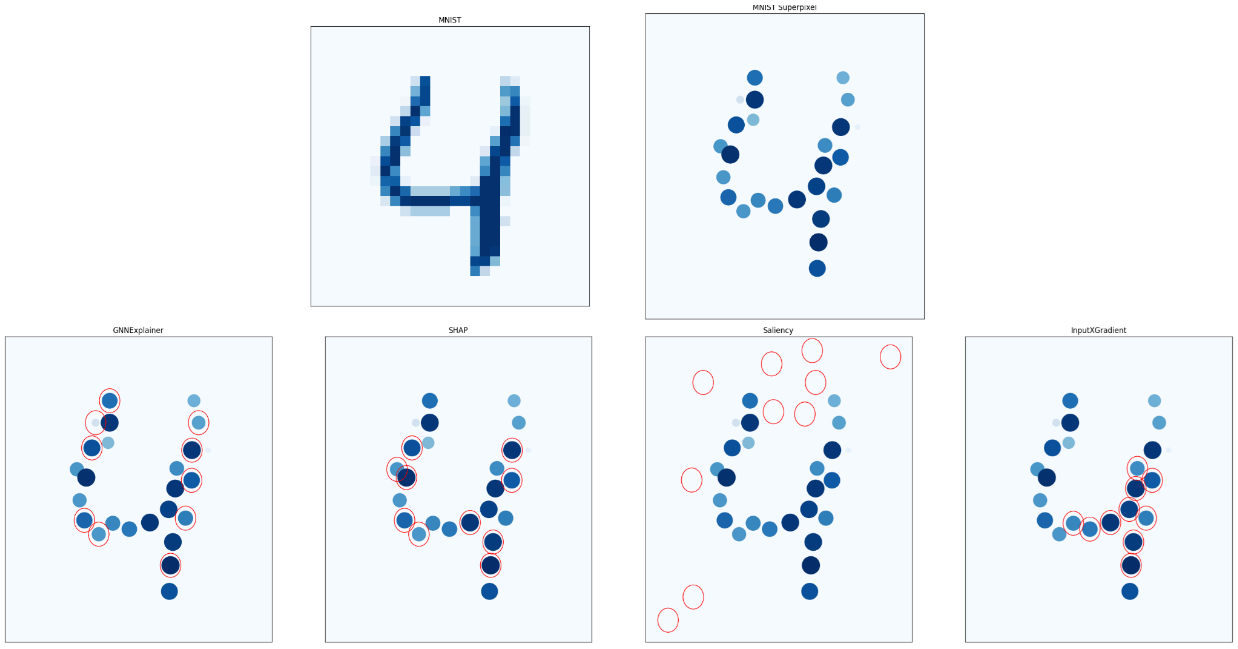

Fig 9: 10 most influential nodes, class: 4, model prediction: 4.

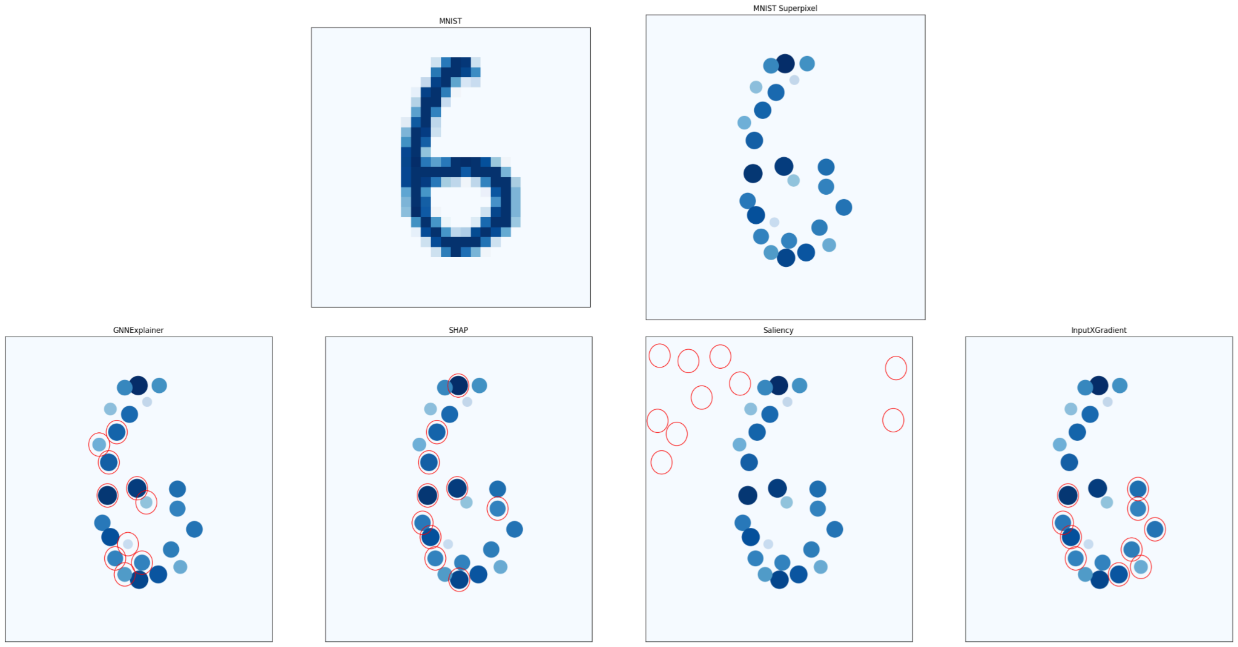

Fig 10: 10 most influential nodes, class: 6, model prediction: 6.

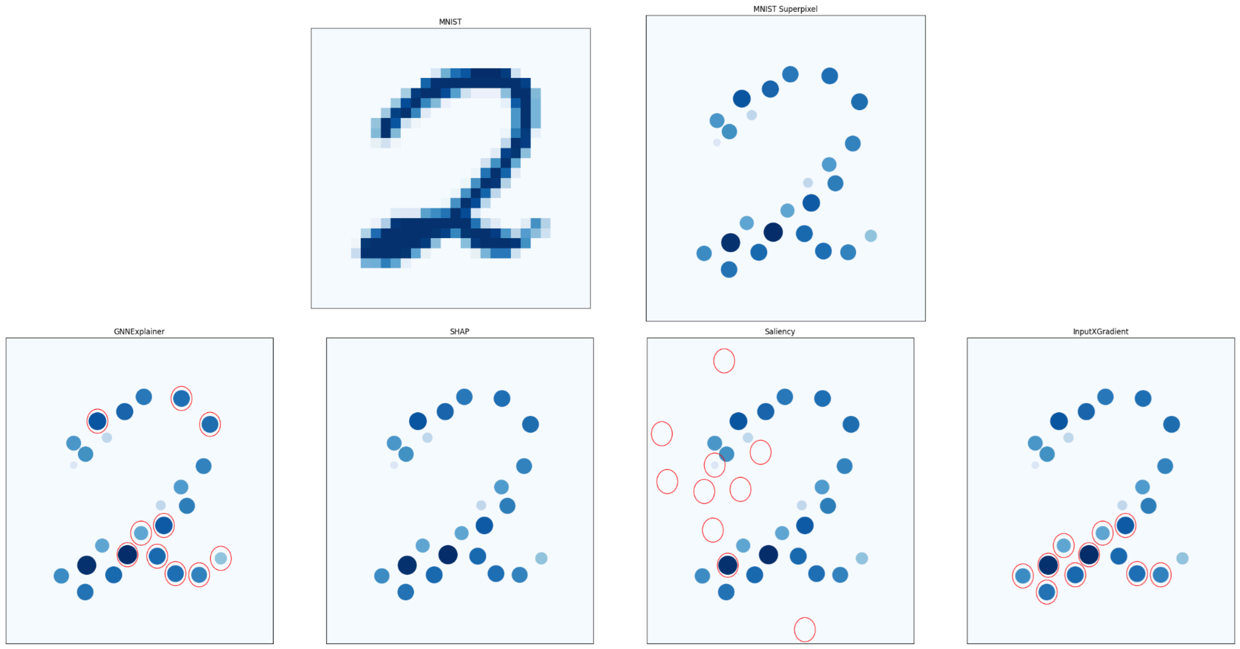

Fig 11: 10 most influential nodes, class: 2, model prediction: 2.

8. Conclusions

In conclusion, our project contributes valuable insights into the application of XAI methods for GNNs, offering a comprehensive comparison and evaluation across multiple datasets, most popular explainers, and meaningful metrics. The results obtained clearly visualize the performance difference between models and methods of evaluation. What is more, we also showed how classical explainers can also be used with graph neural networks and compared their insights into the models' decision making processes.

9. References

[1] https://pytorch.org/
[2] https://pytorch-geometric.readthedocs.io
[3] https://captum.ai/
[4] https://matplotlib.org/
[5] https://networkx.org/
[6] *GNNExplainer: Generating Explanations for Graph Neural Networks*,
[7] *Parametrized Explainer for Graph Neural Network*
[8] *PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks*
[9] *Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking*
[10] *Not Just a Black Box: Learning Important Features Through Propagating Activation Differences*
[11] *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*
[12] *A Unified Approach to Interpreting Model Predictions*
[13] *GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks*
[14] *Evaluating Explainability for Graph Neural Networks*
[15] *Automating the Construction of Internet Portals with Machine Learning*
[16] *A critical look at the evaluation of GNNs under heterophily: are we really making progress?*
[17] *Geometric deep learning on graphs and manifolds using mixture model CNNs*
[18] *GroupNorm? Then BatchNorm, InstanceNorm, LayerNorm, … | by Lujia | Medium*
[19] *https://paperswithcode.com/method/gcn*
[20] *https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html*
[21] *https://towardsdatascience.com/graph-attention-networks-under-the-hood-3bd70dc7a87*
[22] *https://paperswithcode.com/method/gcn*