Contents

Team members:

- Jędrzej Miczke 156068
- Mateusz Nowicki 156064
- Krzysztof Skrobała 156039
- Wojciech Bogacz 156034

1 Introduction

This project aims to test methods for predicting a paper category based on its abstract, and the graph structure of citations between papers. The main idea is to reproduce the results from the paper *Semi-Supervised Classification with Graph Convolutional Networks* by Thomas N. Kipf and Max Welling, presented at ICLR 2017, and to compare it with other methdods.

2 Related Work

Graph attention networks (Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.) Graph Attention Networks introduce a novel attention formulation of graph convolution that allows each node to learn how much importance to assign to each of its neighbors, rather than aggregating them with fixed weights normalized by degree as in prior GCNs. The key innovation is a masked self-attention mechanism applied locally on graph neighborhoods, which is fully differentiable, parameter efficient, and avoids costly spectral operations such as Laplacian eigendecomposition. This design naturally handles differently sized neighborhoods, supports directed graphs, and does not depend on knowing the full graph structure upfront, making it well suited for inductive learning on unseen graphs. Empirically, GATs consistently outperform or match newest graph convolutional and inductive methods across both transductive citation networks and large-scale inductive benchmarks, demonstrating that learned, data-driven neighbor weighting significantly improves representational power and generalization over uniform or heuristic aggregation schemes.

Transformers networks (Yun, S., Jeong, M., Kim, R., Kang, J., & Kim, H. J. (2019). Graph transformer networks. Advances in neural information processing systems, 32.) Graph Transformer Networks address a key limitation of existing graph neural networks by learning the graph structure itself, rather than assuming a fixed, previously specified graph, which is especially problematic in heterogeneous or misspecified graphs. The central novelty is the Graph Transformer layer, which learns a soft, differentiable selection and composition of edge types to automatically generate useful meta-path-based graphs of varying lengths, effectively discovering multi-hop and composite relations directly from data. These learned graph structures are then used for standard graph convolution, and multiple such representations are combined to form powerful node embeddings. Compared to prior methods that rely on manually crafted meta-paths or homogeneous approximations, GTNs achieve brilliant performance on heterogeneous node classification benchmarks, adaptively choose effective relation lengths, and provide interpretable insights through attention weights over learned meta-paths, demonstrating both superior accuracy and structural interpretability. # Dataset description

3 Methods Decription

3.1 Graph Convolutional Networks (GCN)

Graph Convolutional Networks (GCNs) extend convolution to graph-structured data by iteratively aggregating and transforming features from a node's neighbors using the graph topology.

A common layer-wise propagation rule is:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

where:

- $\tilde{A} = A + I$ is the adjacency matrix with self-loops.
- $\tilde{D}$ is the degree matrix of $\tilde{A}$.
- $H^{(l)}$ represents the node features at layer l.
- $W^{(l)}$ denotes the learnable weights.
- $\sigma(\cdot)$ is a non-linear activation function.

We can note that when there are no edges, the $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ term becomes the identity matrix, and the layer reduces to a standard feedforward layer, which will be useful for a baseline comparison.

3.2 Graph Attention Networks (GAT)

Graph Attention Networks (GATs) incorporate an attention mechanism to assign different importance to neighboring nodes when aggregating features.

For a node i, attention coefficients are computed as:

$$\alpha_{ij} = \frac{\text{exp}\left(\text{LeakyReLU}\left(\vec{a}^{\top}[W\vec{h}_i \,\|\, W\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}(i)} \text{exp}\left(\text{LeakyReLU}\left(\vec{a}^{\top}[W\vec{h}_i \,\|\, W\vec{h}_k]\right)\right)}$$

and the node update is:

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

where: - $\vec{h}_i$ represents the input node features. - W and $\vec{a}$ are learnable parameters. - $\mathcal{N}(i)$ denotes the neighbors of node i. - $\|$ is the concatenation operation. - $\sigma(\cdot)$ is a non-linear activation function.

3.3 Graph Sample and Aggregate (GraphSAGE)

GraphSAGE is an inductive graph representation learning method that learns node embeddings by sampling and aggregating features from a fixed-size set of neighboring nodes.

For node i at layer l, the neighborhood aggregation is:

$$h_{\mathcal{N}(i)}^{(l)} = \text{AGGREGATE}^{(l)} \left(\{ h_j^{(l)} : j \in \mathcal{N}(i) \} \right)$$

and the node update is:

$$h_i^{(l+1)} = \sigma \left(W^{(l)} \cdot [\, h_i^{(l)} \,\|\, h_{\mathcal{N}(i)}^{(l)}] \right)$$

followed by normalization.

where: - $\mathcal{N}(i)$ denotes the **sampled** neighbors of node i. - $W^{(l)}$ are the learnable weights. - $\|$ indicates concatenation. - AGGREGATE$(\cdot)$ is a differentiable function (e.g., mean, max-pooling, or LSTM).

3.4 Graph Transformer

Graph Transformers adapt the self-attention mechanism of Transformers to graph-structured data, enabling nodes to attend to other nodes based on learned attention scores while incorporating graph structure via positional or edge encodings.

For a node i, attention is computed as:

$$\text{Attn}(i, j) = \frac{(W_Q h_i)(W_K h_j)^\top}{\sqrt{d}}$$

and the node update is:

$$h_i' = \sum_{j \in \mathcal{V}} \text{softmax}_j \left(\text{Attn}(i, j) + b_{ij} \right) W_V h_j$$

where: * h_i are the node features. * W_Q, W_K, W_V are learnable projection matrices for Query, Key, and Value. * d is the attention dimension (used for scaling). * b_{ij} is a bias term encoding graph structure (e.g., shortest path distance or edge connectivity). * $\mathcal{V}$ is the set of all nodes in the graph.

4 Evaluation Strategy

We measure the performance of each model using accuracy on a held-out test set. Additionally, we monitor training and validation loss to assess convergence and potential overfitting.

Accuracy measures the proportion of correctly classified examples and is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

5 Results

Model	Val Accuracy	Test Accuracy
GCN - from the original paper	-	0.703
GCN - the replication	0.718	0.691
GCN - without edges	0.572	0.571
Graph Isomorphism Network	0.666	0.648
Graph Attention Network	0.748	**0.728**
Graph Sample and aggregate	0.710	0.707
GraphTransformer	0.724	0.694

5.0.1 Conclusions

Across all experiments on the Citeseer dataset, models that incorporate graph structure consistently outperform those that ignore it, confirming the importance of relational information in semi-supervised node classification. Our replication of the original GCN achieved slightly lower accuracy than reported in the paper but still demonstrated strong performance relative to baselines. Among all models evaluated, the Graph Attention Network achieved the highest accuracy, suggesting that adaptive, attention-based neighbor weighting is more effective than fixed aggregation. GraphSAGE and GraphTransformer also performed competitively, while GIN lagged somewhat, reflecting its sensitivity to hyperparameters in smaller citation networks. In contrast, models without edges—GCN without edges and an MLP—performed significantly worse, emphasizing that leveraging graph connectivity is crucial for achieving high performance in this task.