# Graph Neural Networks-based Multilabel Classification of Citation Network

Guillaume Lachaud[1,2], Patricia Conde-Cespedes[1,3], and Maria Trocan[1,4]

[1] ISEP - Institut Supérieur d'Électronique de Paris.
10 rue de Vanves, Issy les Moulineaux, 92130-France
[2] glachaud@isep.fr
[3] pconde@isep.fr
[4] maria.trocan@isep.fr

**Abstract.** There is an increasing number of applications where data can be represented as graphs. Besides, it is well-known that artificial intelligence approaches have become a very active and promising research field, mostly due to deep learning technologies. However popular deep learning architectures were designed to treat mostly image and text data. Graph Neural Network is the branch of machine learning which builds neural networks for graph data. In this context, many authors have recently proposed to adapt existing approaches to graphs and networks. In this paper we train three models of Graph Neural Networks on an academic citation network of Computer Science papers, and we explore the advantages of turning the problem into a multilabel classification problem.

**Keywords:** Graph Neural Networks · Citation Network · Multilabel Classification

## 1 Introduction

Graphs are a kind of structured data that have become one of the pillars of our society. They are ubiquitous, appearing for example in biology with protein interaction graphs, in chemistry with molecules, in epidemiology, telecommunications, finance, sociology with social networks. Graphs are expressive enough that they can model complex networks such as interactions between billions of users. As technology develop and the collected data becomes more structured, graph's role in shaping our world will increase.

While traditional machine learning was focused on providing good statistical models that had a strong theoretical background, deep learning employs a data driven approach. The first neural network models can be trace back as far 70 years ago, however they lacked the computational resources and data to be efficient. With the rise of faster computers and the use of Graphical Processing Units (GPUs), alongside the accumulation of data on the Internet, deep learning has seen tremendous success in computer vision [12] and natural language

processing [3].

Although Convolutional Neural Networks (CNNs) perform well on image tasks and Recurrent Neural Networks (RNNs) are the standard for manipulating text sequences, neither of these architectures are designed to exploit the relations between nodes in graphs. These led researchers to build new architectures of neural networks dedicated to handling graph data: Graph Neural Networks (GNNs). In just a few years the field of GNN has vastly grown and incorporates knowledge from machine learning and graph theory [24].

In this paper, we perform a multiclass classification on the citation network of all the Computer Science papers (CS) published in the arXiv web, extracted from the the Open Graph Benchmark repository [9], and referred to as *ogbn-arxiv*. We train three graph neural networks and we select the best performing one to examine the misclassification errors. We attribute the errors to two main causes: in many cases, the second most likely class predicted by the model is the real class. Secondly, the model has difficulty classifying some of the less represented classes. We propose turning the task into a multilabel classification task.

In many cases, classification task require neural networks to produce multiple labels: images have several elements, text can be related to different topics, and nodes in a graph may be related to several classes via different neighbors. Formally speaking, given an input feature vector $x$, multilabel classification aims at finding a prediction of a label vector $y$, where each element takes the value 0 or 1, indicating whether the input belongs to a class or not. Multilabel classification for each type of data led to the development of specialized network architectures [16,21,13].

The paper is divided as follows: Section 2 presents the history of GNNs and some of the existing graph datasets that are used as benchmarks. We detail the architecture of the GNNs we are using in the paper. Section 3 describes ogbn-arxiv in detail; Section 4 introduces the results of our experiments Section 5 considers a multilabel classification approach to mitigate the errors of the model. Section 6 concludes the work and offers future areas of improvement.

## 2 Related works

Graph neural networks first emerged in the context of extending recurrent neural networks to handle structured data [18]. Most of the leading approaches now follow a structure similar to the one introduced in [6]: the hidden representation of a node is updated using the hidden representation of its neighbors. The main differences in architectures are usually in how the representations are used, e.g. aggregating the features in GraphSAGE [7], and in the weighing of the neighbors, e.g. using fixed weights in Graph Convolutional Networks (GCN) [11], or assigning attention score in Graph Attention neTwork (GAT) [20]. For complete

surveys of graphs neural networks, see [24,23]. One of our motivations for selecting GraphSAGE, GCN and GAT, is that since they are all variations on a more general architecture framework, experiments done with these models will to some extent generalize to other GNNs.

We use the following notations to describe the models: a graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges. The adjacency matrix of $\mathcal{G}$ is $A$. $N$ represents the number of nodes in the graph, i.e. the cardinality of $\mathcal{V}$. The set of neighbors of a node $u$ is written as $\mathcal{N}_u$ [5]. The activation matrix of layer $l$ is $H^{(l)} \in \mathbb{R}^{N \times D}$, where $D$ is the dimensionality of the layer. We use $h_u^{(l)}$ to specify the activation vector of node $u$ at layer $l$. The weight matrix at layer $l$ is $W^{(l)}$. By convention, $H^{(0)} = X$, the initial feature matrix of the nodes. $\sigma$ represents an activation function, e.g. a ReLU (Rectified Linear Unit), defined as $x \mapsto \max(0, x)$.

### Graph Convolutional Networks

GCNs (**G**raph **C**onvolutional **N**etworks) were introduced in [11] in 2017 for semi-supervised learning on graphs. The main idea is based on using spectral graph wavelet transforms (also called spectral convolutions on graphs) and their approximation using Chebyshev polynomials, both introduced in [8] and refined in [2]. The authors in [11] used a first-order approximation of the spectral convolutions and approximated the largest eigenvalue of the graph Laplacian to be equal to 2, arguing that neural network would adapt during training. With these assumptions, they defined a graph convolutional layer to be defined by the following propagation rule

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

$\tilde{A}$ is the adjacency matrix of the graph with added self-connections, i.e. $\tilde{A} = A + I_N$. Finally, $\tilde{D}_{ii} = \sum_j \tilde{A_{ij}}$.

A GCN is usually composed of several of these layers, with the activation function of the last layer being a softmax to output probabilities.

### Graph Attention Networks

Inspired by attention mechanisms in deep learning which were first developed in natural language processing for dealing with sequences [1], GATs (**G**raph **A**ttention Ne**T**works) were introduced in 2018 in [20] and use the following layer-wise propagation rule

$$h_u^{(l+1)} = \sigma\left(\sum_{v \in \mathcal{N}_u} \alpha_{uv}^{(l)} W^{(l)} h_v^{(l)}\right) \tag{2}$$

[5] The neighborhood can include the node itself.

$\alpha_{uv}^{(l)}$ is the *normalized attention coefficient* at layer $l$ of node $v$ with respect to $u$, that is, it indicates how important the features of node $v$ are to node $u$. The coefficients are obtained by computing the attention coefficients then performing a softmax for normalization. More formally, with $e_{uv}^{(l)}$ the attention coefficients at layer $l$ and $a$ the attention mechanism, we have

$$e_{uv}^{(l)} = a\left(W^{(l)}h_u^{(l)}, W^{(l)}h_v^{(l)}\right) \tag{3}$$

$$\alpha_{uv}^{(l)} = \frac{\exp(e_{uv}^{(l)})}{\sum_{w\in\mathcal{N}_u}\exp(e_{uw}^{(l)})} \tag{4}$$

The attention mechanism can be any function that takes as input two vectors, with the same dimension as the product $W^{(l)}h_u^{(l)}$ and outputs a real value. For example, $a$ can be a feed-forward neural network.

### GraphSAGE

GCN and GAT can perform both transductive and inductive learning, that is, use the observed data to predict the behaviour of new data (transductive), or use the observed data to infer general rules as to the behaviour of the data (inductive). Graphs are particularly challenging for inductive learning because it requires learning the structures of subgraphs. Contrary to GCN and GAT, GraphSAGE (Graph **SA**mple and aggre**G**at**E**) was specifically designed to tackle the challenge of inductive learning [7] and was introduced in 2017.

The network uses the following forward propagation rules for layer $l+1$ and for each node $v$ in $\mathcal{V}$

$$h_{\mathcal{N}_v}^{(l+1)} = \mathbf{aggregate}_l\left(\{h_u^{(l)}, \forall u \in \mathcal{N}_v\}\right) \tag{5}$$

$$h_v^{(l+1)} = \sigma\left(W^{(l)}\mathbf{concatenate}(h_v^{(l)}, h_{\mathcal{N}_v}^{(l)})\right) \tag{6}$$

$$h_v^{(l+1)} = \frac{h_v^{(l+1)}}{\|h_v^{(l+1)}\|^2} \tag{7}$$

$h_{\mathcal{N}_v}^{(l+1)}$ is an activation vector for the neihborhood of node $v$. The **concatenate** function concatenates the two vectors for the matrix multiplication with $W^{(l)}$.

The authors in [7] argue that the aggregator function should be symmetric so as to be independent of the ordering of the neighbors, trainable and have high representational capacity. An aggregator can be a simple mean aggregator, which, when injected in Eq. 5, gives

$$h_{\mathcal{N}_v}^{(l+1)} = \sigma\left(W^{(l)}\cdot\left(\frac{1}{|\mathcal{N}_v|+1}(h_v^{(l)} + \sum_{u\in\mathcal{N}_v} h_u^{(l)})\right)\right) \tag{8}$$

More aggregators are described, such as an LSTM one, in [7].

While the neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$ can be defined arbitrarily, in practice we draw a uniform sample of fixed size from the neighbors of the node. This draw is performed for every layer.

In order to compare GNN architectures, it is essential to have high quality datasets and well designed benchmarks.

## 3 Dataset description

The first graph benchmark datasets were quite small, having only a few thousand nodes [17]. This posed a problem when trying to compare GNNs, because the graphs were too small to create meaningful differences between the methods. In the last few years, there has been a trend in trying to standardize benchmarking of GNNs [4]. In this vein, [9] introduced an ensemble of graph datasets of varying size and type: directed, undirected, homogeneous, heterogeneous, etc. Furthermore, they provided a benchmark framework to ease to process of comparing methods. These datasets tackle the three types of graph tasks, which are node, edge and graph-level property predictions. The dataset we use in this paper, ogbn-arxiv, comes from their paper.

A leaderboard is available at `https://ogb.stanford.edu/docs/leader_nodeprop/` to see the best-performing models on obgn-arxiv. Many of the leading submissions are based on some variations of GATs. Two notable approaches in the leaderboard are presented in [19] and [10]. The first one proposes to use a GNN that spreads information further than one node to its first neighbors using multi-hop neighborhood aggregation. The second is an approach that does not use GNNs, but rather rely on using a multi-layer perceptron followed by two post-processing steps that propagate errors from the training data.

ogbn-arxiv is a directed homogeneous graph. It has $169,343$ nodes which represent papers from the arXiV Computer Science repository, and $1,166,243$ directed edges, which correspond to citations between papers. Each node has a date attribute, corresponding to the year of publication, which ranges between 1991 and 2020[6], and a 128-dimensional feature vector. The feature vector represents the average of the word embeddings of all the words in the title and the abstract of the paper, which the authors of [9] obtained by applying a WORD2VEC [15] model that they trained on Microsoft Academic Graph (MAG) [22].

Each paper is labelled by the authors and the arXiv moderators; these labels are assigned a number between 0 and 39, representing the 40 categories of the arXiv CS (Computer Science) repository. Each category is denoted by two letters;

[6] There are 10 papers whose publication date is before 1991, which is the year arXiv was publicly released.

their full name can be found at `https://arxiv.org/corr/subjectclasses`.

In [9], the authors proposed splitting the dataset with respect to the year of publication of the papers, on the basis that this reflects one of the real world applications of GNNs, which is to predict the category of new papers using only already published papers; furthermore, they argue it is a more challenging task than just randomly splitting between train, validation and test. Thus, the split is the following: the train set consists of all papers published before 2018; the validation set has all the papers published in 2018; and the test comprises all the papers from 2019 (inclusive) onwards.

## 4 Experiments

In this section, we use a single class classification approach. A deep analysis of the misclassifications will lead us to explore a multilabel approach in Section 5.

For all the experiments, we use a 24GB NVidia RTX GPU. The code is written in Python, Pytorch and PyTorch Geometric [5]. We use the OGB (Open Graph Benchmark) [9] package to get the *ogbn-arxiv* dataset.

Our choice of GCN, GraphSAGE and GAT is based on the following observations: they are special cases of Message Passing Neural Networks [6] which are one of the dominant forms of graph neural networks. Furthermore, they form the basis of most of the leading architectures in the leaderboard on the OGB datasets. The leaderboard is available at `https://ogb.stanford.edu/docs/leader_nodeprop/`.

We train 10 instances of GCN, GraphSAGE and GAT for 500 epochs each. Because GNNs are prone to *over-smoothing* when using too many layers [14], each of our models has 3 layers with a hidden layer size of 256 units. We use dropout to mitigate overfitting. Because GraphSAGE and GCN only handle undirected graphs, we consider the citation network as an undirected network. Average results of training are presented in Table 1. The GAT outperforms the other models. This is consistent with the results of the leaderboard for ogbn-arxiv available at `https://ogb.stanford.edu/docs/leader_nodeprop/`, where GAT-based models occupy the first places.

**Table 1.** Training results for the arXiv dataset

| method | validation accuracy | testing accuracy |
|---|---|---|
| GCN | $73.33 \pm 0.09$ | $72.06 \pm 0.2$ |
| Graphsage | $71.94 \pm 0.1$ | $70.77 \pm 0.26$ |
| GAT | $\mathbf{73.66 \pm 0.11}$ | $\mathbf{72.41 \pm 0.19}$ |

In the rest of the paper, in keeping with the results obtained in our own experiments and by other researchers, we focus on the results given by the GAT model.

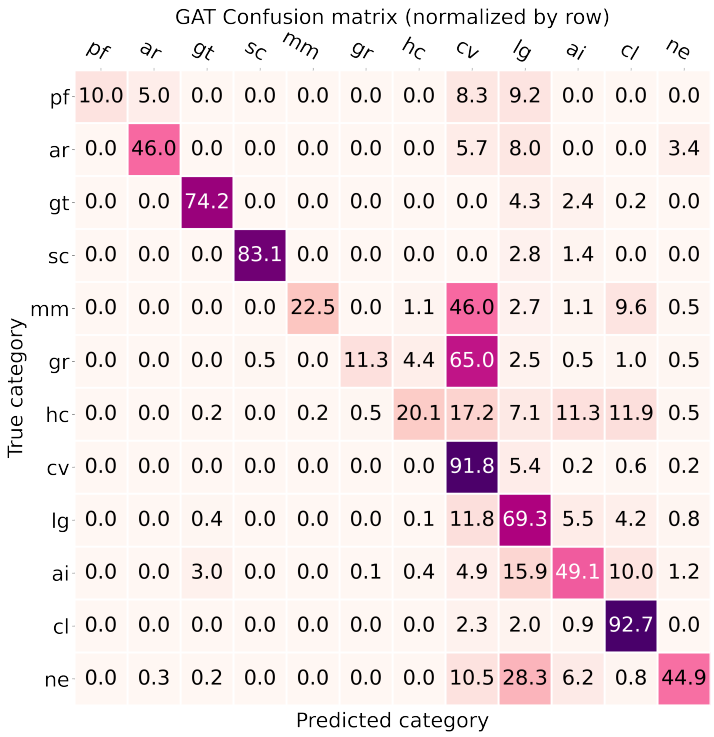


**Fig. 1.** Susbset of the confusion matrix

To analyze the misclassification errors made by the model, we need to go beyond the accuracy score and look at the confusion matrix on the test set, which will help us see where it fails to generalize. The value at row $c_i$ and column $c_j$ indicates the number of times the model has assigned the label $c_j$ to a node from $c_i$, divided by the total number of nodes of category $c_i$ in the test set. The rows have been normalized and each one adds up to 100. The higher the values outside the diagonal are, the more the model made mistakes. The categories were ordered in such a way that the small classes occupy the first columns while the middle of the matrix is for the most populated classes and the rest of the columns represent mostly middle-sized classes. A subset of the full confusion matrix is displayed in 1 with the categories that are discussed in the rest of the paper.

We first observe that the size of the category in the training set, as represented by the "train size" column in Table 3 is not a sufficient indicator of poor performance. The model achieves low accuracy on such categories as **ar** (Hardware Architecture) and **pf** (Performance) but successfully classifies nodes from **gt** (Computer Science and Game Theory) and **sc** (Symbolic Computing), despite the fact that these categories have approximately the same number of nodes in the training set. This suggests that some categories display more cohesiveness than others, and that the network is able to detect this pattern.

Still on the topic of categories with little representation, we see systematic misattribution for nodes in the **mm** (Multimedia) and **gr** (Graphics) categories, which are classified as **cv**. Considering that the three subjects likely share a similar terminology, and that the initial features of the nodes were based on the words in the title and the abstract, there is little hope, without changing the features, to correctly predict these classes.

Next, we are faced with subject areas that are intrinsically interdisciplinary, which means they exploit ideas from other areas of research. The most eminent representative of these categories is **hc** (Human Computer Interaction). By design, HCI tends to capitalize on the advances in various fields, e.g. computer vision, natural language processing, and study the impact, positive or negative, they can have on users. In ogbn-arxiv, this will be reflected in two manners: **hc** nodes have neighbors that can belong to other classes, and two **hc** nodes can have vastly different features.

Finally, the error which is the key factor in driving down the accuracy is the confusion of categories within a group of similar categories. This is exemplified with the categories **cv** (Computer Vision), **lg** (Machine Learning), **ai** (Artificial Intelligence), **cl** (Computation and Language, mostly natural language processing) and **ne** (Neural and Evolutionary Computation). About 30% of **ai** nodes in the test set are incorrectly attributed to the one of the above classes, while 20% of **lg** nodes and 35% of **ne** nodes are similarly misclassified. All these categories mutually fuel the research of the others. The two biggest reasons for the misclassification are a combination of two causes mentioned earlier: many nodes from these categories share a similar terminology, e.g. papers on neural networks have similar characteristics; and the nodes cite papers from all the areas in the group.

Considering the overlapping themes of some categories, as well as the interdisciplinary content of some papers, a multilabel classification approach is preferable to the single label classification task. Firstly, it allows a finer grained categorization of papers, distinguishing between papers in the robotics field that have a computer vision component with those that have a natural language processing component. Secondly, it helps concentrate on the bigger errors made by the neural networks: those in which the category is not in the top predictions.

## 5 Multilabel classification approach

Instead of focusing on only the top prediction of the model, we retrieve the three most likely predicted classes of our GAT model for each node in the test set. The set of estimated probabilities is usually obtained by applying a softmax activation function to the last layer of the neural network; in the case of multi-class classification, to make a prediction, we simply output the category which is associated with the highest probability. We compute the number of times the correct category is the prediction (accuracy, or top 1), as well as the number of times it appears in the two (top 2) or three (top 3) categories with the highest estimated probabilities. Overall, while the model achieves 72.4% accuracy, the right category is in the two highest predictions 87.3% of the time, a 15% increase. In the top 3, this number rises to 92.4%. Results for each category are presented in Table 3, alongside the relative size of the category in the training set (given in percentage) and its population in the test set. The arXiv categories in bold are the ones discussed in the text. Additionally, a representation of the top 3 predictions for some nodes is presented in Figure 2.

**Table 2.** Top 3 score on training, validation and test

| dataset | top 1 | top 2 | top 3 |
|---|---|---|---|
| train | 79.31 | 90.36 | 94.14 |
| valid | 73.62 | 87.76 | 92.73 |
| test | 72.27 | 87.25 | 92.36 |

We see that, within a group of non-mutually exclusive categories, there are some classes that attract most of the predictions, such as the **cv** and **cl** which are in the group of artificial intelligence related categories. These leads to poor accuracy scores for the **lg** and **ai** classes. However, when we look at the three highest estimated probabilities, the network gets most of the **lg** and **ai** samples right. For example, node 1 in Figure 2 belongs to the **lg** category, which is the second prediction of the model. Similarly, nodes 2, 3, 5 and 7 all belong to the **ai** category, which is the second or the third prediction from the model.

Additionally, the top three predictions are either related to the true category, or to the category of the neighbors. For example, node 1 has neighbors that belong to the **ai**, **lg**, **cv**, **cl** categories. This means that the model is properly learning from the information contained in the neighbors. Nodes with neighbors from different categories than themselves will rarely be classified in the correct category, but the top predictions of the model will most often be related to the content of the paper. This suggests that focusing on a single category is not sufficient to properly classify a paper, and that a better way is to look at the first two or three predictions to get a meaningful categorization of the paper. Node 1 is paper from the **ai** category, but it cites papers from the **cv** category;

**Table 3.** Top 3 category predicted by the GAT model. The train size represents the percentage of nodes in the training set that are from each category. The test column indicates the number of nodes from the test set that are in each category.

| subject | top 1 | top 2 | top 3 | train size (%) | test | subject | top 1 | top 2 | top 3 | train size (%) | test |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **cv** | 91.83 | 98.10 | 98.99 | 10.99 | 10477 | cy | 17.13 | 43.12 | 59.33 | 1.12 | 654 |
| **lg** | 69.27 | 91.30 | 96.38 | 7.69 | 10740 | cg | 75.72 | 85.62 | 90.42 | 1.64 | 313 |
| it | 90.56 | 96.28 | 97.54 | 17.91 | 2849 | dm | 26.02 | 54.65 | 78.81 | 1.71 | 269 |
| **cl** | 92.74 | 97.06 | 98.27 | 4.77 | 4631 | pl | 47.41 | 74.09 | 81.87 | 1.39 | 386 |
| **ai** | 49.14 | 71.13 | 82.68 | 5.70 | 1455 | **hc** | 20.10 | 36.82 | 52.89 | 0.77 | 622 |
| ds | 69.87 | 86.85 | 92.43 | 5.97 | 1414 | dl | 76.17 | 81.78 | 85.05 | 1.21 | 214 |
| ni | 55.12 | 84.32 | 91.12 | 4.46 | 1250 | fl | 55.45 | 73.18 | 80.00 | 1.02 | 220 |
| cr | 67.04 | 82.18 | 87.91 | 3.15 | 1869 | sd | 77.47 | 89.89 | 94.95 | 0.50 | 475 |
| dc | 52.73 | 75.12 | 83.07 | 3.23 | 1246 | ma | 6.28 | 27.62 | 62.76 | 0.43 | 239 |
| lo | 67.94 | 90.18 | 94.13 | 3.96 | 733 | et | 57.89 | 74.64 | 81.82 | 0.44 | 209 |
| ro | 70.91 | 88.77 | 94.63 | 1.83 | 2066 | **mm** | 22.46 | 52.94 | 66.31 | 0.42 | 187 |
| si | 68.40 | 82.61 | 88.18 | 3.14 | 1041 | **sc** | 83.10 | 88.73 | 88.73 | 0.52 | 71 |
| **gt** | 74.16 | 87.40 | 91.55 | 2.76 | 627 | ce | 28.36 | 44.78 | 52.99 | 0.42 | 134 |
| sy | 63.72 | 79.47 | 84.96 | 2.06 | 419 | na | 33.33 | 55.56 | 70.37 | 0.48 | 54 |
| se | 62.00 | 76.24 | 81.81 | 1.69 | 808 | **gr** | 11.33 | 56.16 | 77.34 | 0.22 | 203 |
| ir | 46.52 | 77.47 | 90.13 | 1.48 | 892 | **pf** | 10.00 | 34.17 | 52.50 | 0.26 | 120 |
| cc | 51.59 | 71.88 | 87.25 | 2.47 | 345 | ms | 57.83 | 79.52 | 84.34 | 0.30 | 83 |
| db | 63.41 | 74.43 | 83.37 | 1.78 | 481 | **ar** | 45.98 | 65.52 | 74.71 | 0.27 | 87 |
| **ne** | 44.90 | 64.97 | 82.96 | 1.42 | 628 | oh | 0.00 | 1.96 | 3.92 | 0.33 | 51 |
| os | 8.33 | 25.00 | 50.00 | 0.08 | 36 | gl | 0.00 | 0.00 | 0.00 | 0.02 | 5 |

thus it is likely to contain a sizable amount of information related to **computer vision**, even if it is not the main theme of the paper.

We also observe that the challenges faced by interdisciplinary categories remain when we observe the top three predictions: the model correctly has **hc** in its top three predictions in only 53% of the cases. Node 4 and node 7 in Figure 2 illustrate the situation. Node 1 only have **lg** neighbors, while node 2 only has **cv** neighbors. Furthermore, **hc** is not in the first three predictions of the model.

## 6 Conclusion and future works

In this paper we trained three common graph neural networks (GCN, GraphSAGE and GAT) on the ogbn-arxiv graph for node classification. While typical classification tasks allow the objects to belong to a unique category, we found that many misclassification errors come from the fact that some papers share common features with several other categories that are related. This led us to reframe the problem as a multilabel classification problem where a node might belong to more than one category with a given probability. For instance, a paper in the robotics category might tackle a computer vision problem, while another one might deal with a natural language processing task. We found that considering the top three predicted classes, the real class was present in more than 92%

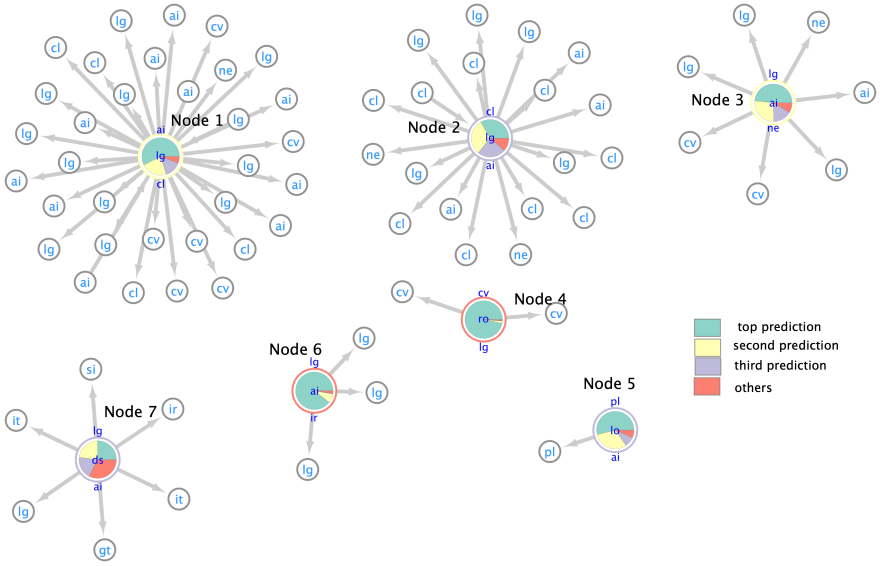


**Fig. 2.** Top 3 predictions for a few nodes in the graph. The pie chart represents the probability assigned by the model to the the first three categories. For each node with a piechart, the label of the first prediction is the one on top, the second prediction the one in the middle and the third prediction the one at the bottom. The nodes without piecharts are the neighbors of the nodes on which we do the predictions, and have their true label written inside them.

of the cases. In addition, we observed that the categories in the top predictions are usually related to the true category, or to the category of the neighbors of the paper. These facts validate the multilabel approach.

Some perspectives for future works include performing a similar analysis on bigger datasets to generalize our findings. The multilabel approach is likely to extend to other domains, as objects in social networks or other real world data do not usually belong exclusively to one class. Furthermore, different set of features can be explored to improve discrimination between classes.

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
2. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 3844–3852. NIPS'16, Curran Associates Inc., Red Hook, NY, USA (Dec 2016)

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-Training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423
4. Dwivedi, V.P., Joshi, C.K., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking Graph Neural Networks. arXiv:2003.00982 [cs, stat] (Jul 2020)
5. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric. arXiv:1903.02428 [cs, stat] (Apr 2019)
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
7. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (Dec 2017)
8. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis **30**(2), 129–150 (Mar 2011). https://doi.org/10.1016/j.acha.2010.04.005
9. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H.T. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual (2020)
10. Huang, Q., He, H., Singh, A., Lim, S.N., Benson, A.R.: Combining label propagation and simple models out-performs graph neural networks. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (May 2017). https://doi.org/10.1145/3065386
13. Lanchantin, J., Sekhon, A., Qi, Y.: Neural message passing for multi-label classification. In: Brefeld, U., Fromont, É., Hotho, A., Knobbe, A.J., Maathuis, M.H., Robardet, C. (eds.) Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11907, pp. 138–163. Springer (2019). https://doi.org/10.1007/978-3-030-46147-8_9
14. Li, G., Muller, M., Thabet, A., Ghanem, B.: DeepGCNs: Can GCNs Go As Deep As CNNs? In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9266–9275. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00936

15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013)
16. Nam, J., Kim, J., Loza Mencía, E., Gurevych, I., Fürnkranz, J.: Large-Scale Multi-label Text Classification — Revisiting Neural Networks. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) Machine Learning and Knowledge Discovery in Databases, vol. 8725, pp. 437–452. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44851-$9_2$8
17. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective Classification in Network Data. AI Magazine **29**(3), 93 (Sep 2008). https://doi.org/10.1609/aimag.v29i3.2157
18. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. IEEE Transactions on Neural Networks **8**(3), 714–735 (May 1997). https://doi.org/10.1109/72.572108
19. Sun, C., Wu, G.: Adaptive Graph Diffusion Networks with Hop-Wise Attention. arXiv:2012.15024 [cs] (Dec 2020)
20. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net (2018)
21. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: CNN-RNN: A Unified Framework for Multi-label Image Classification. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2285–2294. IEEE, Las Vegas, NV, USA (Jun 2016). https://doi.org/10.1109/CVPR.2016.251
22. Wang, K., Shen, Z., Huang, C., Wu, C.H., Dong, Y., Kanakia, A.: Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies **1**(1), 396–413 (Feb 2020). https://doi.org/10.1162/qss$_{a0}$0021
23. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. **32**(1), 4–24 (2021). https://doi.org/10.1109/TNNLS.2020.2978386
24. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. IEEE Trans. Knowl. Data Eng. **34**(1), 249–270 (2022). https://doi.org/10.1109/TKDE.2020.2981333