Improving Fast-GCN for classifying Research Papers

Sizhe Ma
Department of Civil and Environmental Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
`sizhem@andrew.cmu.edu`

Keshav Narayan
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
`ksnaraya@andrew.cmu.edu`

Abstract

In this paper we attempt to modify Fast-GCN in order to improve both the speed and accuracy of research paper classification. We propose two separate methods for generating a sampling distribution at each layer in the GCN that we hypothesize could outperform the sampling method in Fast-GCN. Our results, however, show that in the case of the Pubmed and Cora dataset, our methods do not in fact outperform Fast-GCN. We explore and discuss potential reasons for this below in the paper. Additionally, we show that in the case of the Pubmed dataset, simpler methods like a Random Forest significantly outperform all Graphical Convolutional Networks, while in the Cora dataset, Graphical Convolutional Networks perform better. We explore reasons behind the discrepancy between the performances of both classes of models on these two similar dataset.

1 Introduction

In this project, we attempt to create a model that can efficiently and accurately determine the class of research papers in the Cora and Pubmed dataset. More efficient algorithms to help classify papers in the future could dramatically improve the efficiency of researchers, by enabling them to quickly find relevant work and by allowing conferences to fully automate the process of sorting papers into categories.

The Cora dataset consists of 2708 different papers. Each paper is represented as a binary vector of length 1433, in which the value of the vector at each index corresponds to the presence or absence of a specific word in the paper. Additionally, there are 5729 links between the different papers, each representing a citation between two papers. Each publication in the Cora dataset belongs to one of 7 categories. The Pubmed dataset is larger and consists of 19717 biomedical publications and 44338 citations between the papers. Each paper is represented as a binary vector of length 500, in which each element of the vector corresponds to the TF/IDF score of some word (essentially a weight that takes into account both the frequency of the word in the paper and the rareness of the word across all the papers).

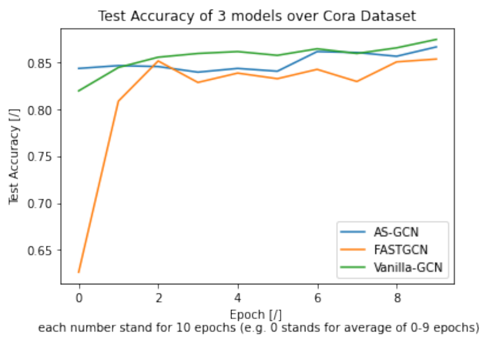
In this project, we attempt to treat this dataset as a graph, in which the papers are nodes, with word vectors of the papers as feature vectors of the nodes, and links between the papers as the edges. This graph is fairly sparse, since the number of edges is roughly on the same order of magnitude as the number of nodes, so we hope that an improvement of graphical convolutional neural networks will be able to perform well. Given such a citation graph along with features for each node, our goal is to correctly predict the category of each node as accurately as possible with the least amount of computation required. We measure test accuracy, train time, and average test f1 score as metrics to compare the models we propose against baselines. We specifically consider the setting in which not all nodes are available at both testing and training time (e.g for example when the set of nodes and edges might be expanding), since this is a more realistic setting for our use case. The algorithm presented in the paper Fast-GCN: Fast Learning with Graph Convolutional Networks via importance sampling (Chen et al., 2018)is able to perform in this setting, and we aim to improve upon these results in our project.

2 Background/Midway Summary

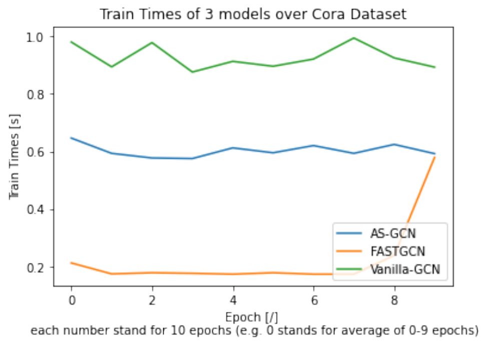
Below is a brief summary of the two dataset we conducted experiments on.

Dataset	**Nodes**	**Edges**	**Classes**	**Features**
Cora	2,708	5,429	7	1,433
Pubmed	19,717	44,338	3	500

For our baselines, we ran 3 different state of art Graph Convolutional Network models (Vanilla GCN, Fast-GCN, and AS-GCN). We found that Fast-GCN was significantly faster than the other methods on both Cora and Pubmed. However, the test accuracy of Fast-GCN was slightly worse than the other two methods.

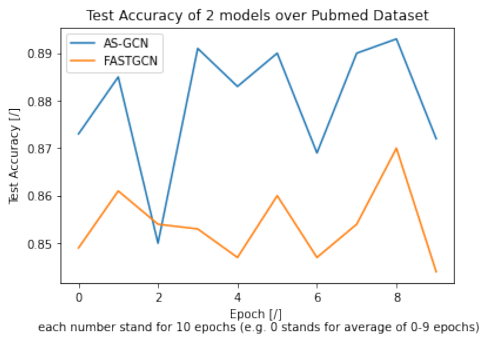


(a) Test Accuracy of 3 models over Cora Dataset

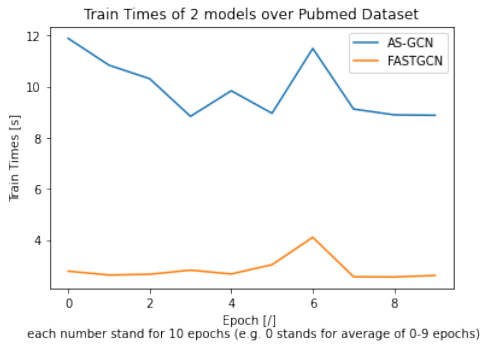


(b) Train Time of 3 models over Cora Dataset

Figure 1: Train/Test Outcome on Cora Dataset



(a) Test Accuracy of 2 models over Pubmed Dataset



(b) Train Time of 2 models over Pubmed Dataset

Figure 2: Train/Test Outcome on Pubmed Dataset

3 Related Works

3.1 Graph Convolutional Networks

Each layer of a GCN can be summarized with the following equation:

$$H^{l+1} = \sigma(\hat{A}H^{l+1}W^{l})$$

Essentially H^l is a matrix where each row contains a feature embedding of a node after l Graph Convolutional Layers. The ith row of H^l is generated by taking a weighted average of the feature embedding from the previous layer (with weights equal to $\hat{A}_i$), then transforming this embedding into a different dimensional space by multiplying on the right by W^l, and finally passing the resulting new embedding through a non-linearity. At the lth layer, this allows each node to aggregate information from nodes in the graph that are within l connections away.

3.2 Fast-GCN

Graph convolutions can be expensive operations, especially for densely connected graphs. One method for reducing the total amount of computation is to simply sample nodes at each layer in the GCN, and for each node to only aggregate information from its neighbors that were included in this sample. The paper FastGCN(Chen et al., 2018) performs this form of sampling, with two different weighting schemes: uniform sampling and importance sampling. In uniform sampling, each node in the layer is given an equal chance of being sampled. In importance sampling sampling, each node is a given a probability of being sampled proportional to the norm of its column in the adjacency matrix divided by the total sum of all the norms of columns in the adjacency matrix. This effectively leads to sampling with higher frequency nodes that are weighted heavily in the aggregation function for other nodes in the graph. A major advantage of Fast-GCN's sampling technique over other techniques is that the same sample is used for all nodes, which saves time and space, as sampling only needs to be performed once at each layer.

3.3 Graph-Sage and other Neighborhood-wise Sampling Methods

The Graph-Sage algorithm (Hamilton et al., 2017) also performs sampling to speed up the aggregation process but uses a very different sampling method from both Fast-GCN and AS-GCN. Instead of constructing a single sample for the whole layer, Graph-Sage creates a separate sample of neighbors for each node at each layer. This can allow for a more representative sample for each node, at the cost of more time in constructing the samples. For larger networks this will perform slower than Fast-GCN or other models that use a single sample for the entire layer. Additionally there are many schemes that are used for determining the probabilities for sampling the neighbors of each node at each layer.

4 Methods

4.1 Simple ML Models

We tested several methods during our project. First, since one of our main goals was to create a fast algorithm, we decided it would be necessary try out Logistic Regression, Random Forest Classifiers, as well as shallow Multi-layer Perceptrons on the data sets to see if these far simpler and faster models were as effective as the more complicated GCN's. Each paper in the Cora dataset is represented as a 1433 dimensional vector, each representing the presence or absence of a single word in the paper. Similarly each paper in the Pubmed dataset is represented as a 500 dimensional vector, where each feature is a TF/ID score of a word in the paper. We thought it was very plausible that these features alone were sufficient to effectively classify papers in these dataset into academic categories, and that features from other papers might not be very necessary. A graphical convolutional model should only be useful when there is relevant information for classifying a node that is not easily available from the node's own feature vector, but found in the feature vectors of other nodes close-by to it in the network.

4.2 Hybrid Methods

Additionally, we were also interested in whether some sort of hybrid between the GCN models and simpler models could work. We constructed a model that essentially uses the Fast GCN model to generate an embedding of each node in the graph. Then our model would concatenate this embedding with the original feature vector and pass this combined feature+embedding vector through a few fully connected layers before outputting the final probability distribution for each node. The assumption here was that some combination of the original features as well as information from other nearby nearby nodes might be more effective than either using only original feature vectors or neighborhood embedding aggregations alone.

4.3 Alternative Sampling Procedure 1

We tried out some different sampling methods for selecting vertices in Fast-GCN. Both of our methods normalize out the effect of the scale of the weights in each row of the adjacency matrix, while the original sampling method in Fast-GCN does not. We illustrate the intuition behind our first method with the following extreme example below:

Consider a simple graph with 4 vertices and the following adjacency matrix:

$$A = \begin{bmatrix} 0 & 1000 & 1000 & 1000 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

We know $H^{l+1} = \sigma(\hat{A}H^{l+1}W^l)$, which in this case gives us,

$$\begin{bmatrix} {h_1}^{l+1} \\ {h_2}^{l+1} \\ {h_3}^{l+1} \\ {h_4}^{l+1} \end{bmatrix} = \begin{bmatrix} 0 & 1000 & 1000 & 1000 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} {h_1}^{l} \\ {h_2}^{l} \\ {h_3}^{l} \\ {h_4}^{l} \end{bmatrix} W^l$$

$$\implies \begin{bmatrix} {h_1}^{l+1} \\ {h_2}^{l+1} \\ {h_3}^{l+1} \\ {h_4}^{l+1} \end{bmatrix} = \begin{bmatrix} 1000({h_2}^l + {h_3}^l + {h_4}^l) \\ {h_1}^l + {h_3}^l + {h_4}^l \\ {h_1}^l + {h_2}^l + {h_4}^l \\ {h_1}^l + {h_2}^l + {h_3}^l \end{bmatrix} W^l$$

Intuitively, given this we might want each node to have the same chance of being sampled here, since for each node the embedding of all of the other nodes from the previous layer are being aggregated equally. However the sampling method used in Fast-GCN, which weights each node proportionally to its column norm, would roughly given a $\frac{1}{3}$ chance nodes 2, 3, and 4 and close to a 0% chance for node 1 to be chosen in the sample.

In our sampling method, we simply normalize the values across each row and then take an average along column i to determine the probability of node i being chosen in the sample. In this case, that would result in a probability of 0.25 of each node being chosen. Lastly we include a hyper-parameter $c \in [0, 1]$, which adjusts our probabilities closer to the uniform to ensure that every node still has some minimum probability of being chosen. Our final probability for node i being chosen in the sample is determined by,

$$p_i = \frac{c}{n} + (1-c)p_i^{calculated}$$

4.4 Alternative Sampling Procedure 2

Another idea we considered was that when determining probabilities for the sampling distribution, we should not only take into account how closely connected a node is to other nodes, but also take into account how connected those other nodes are. Put simply, a node strongly connected to other nodes that are also well connected should have a higher chance of being sampled than a node strongly connected to otherwise relatively disconnected nodes. We can take this a step further and hypothesize that the ideal probability of sampling a node should be the probability we reach that node starting from a random other node and taking a few steps based on transition probabilities derived from the adjacency matrix. If the graph is irreducible, aperiodic, and we take infinite such steps, then the probability of reaching a particular node from any of the other nodes converges to its limiting distribution. We considered adding a small positive ϵ to every entry in the adjacency matrix and normalizing each row, in order to transform the adjacency matrix into a valid probability transition matrix with a limiting distribution. Then by finding the normalized eigenvector of this matrix with eigenvalue closest to 1 (i.e the limiting distribution), we can hopefully use this as an effective sampling distribution with which to select nodes.

5 Results

For our first alternative sampling method, we tried different values of the hyper-parameter c in order to find the optimal weighting between uniform sampling and our calculated probability distribution. We found that $c = 0.2$ worked the best on both Pubmed and Cora dataset, and plotted the results below in figure 3.

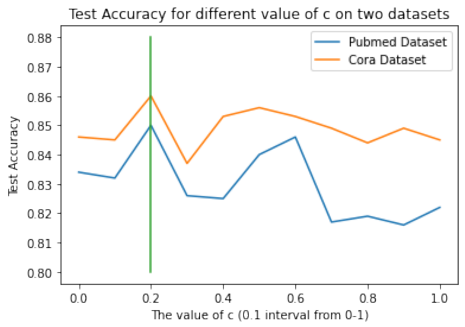


Figure 3: Test Accuracy with different value of c

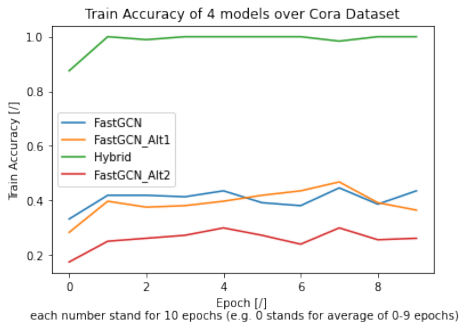
After determining the optimal value of c (0.2), we began comparing the performance of different methods on the two dataset. The following table is the final performance for each of the methods (test accuracy):

Dataset	**FastGCN**	**FastGCN with Alternative Sampling**	**Hybrid Methods**
Cora	0.854	0.860	0.680
Pubmed	0.844	0.847	0.886

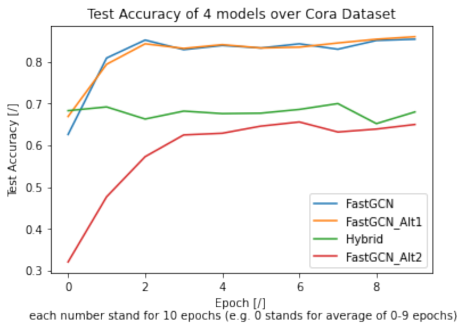
Additionally we ran several other very fast machine learning techniques on both dataset as part of our baseline:

Dataset	**Logistic Regression**	**Multi-layer Perceptron**	**Random Forest**
Cora	0.363	0.737	0.722
Pubmed	0.854	0.880	0.901

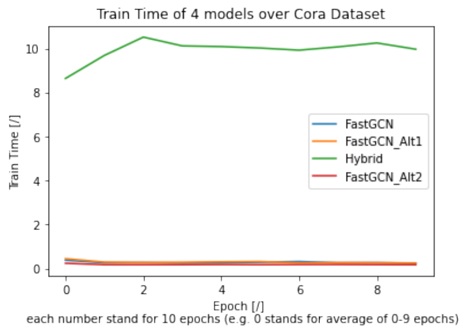
Below we present a comparison between GCN-Related Techniques in terms of train and test accuracy over time. Each figure contains 4 lines; one for each of the following: Vanilla FastGCN Methods, FastGCN with Alternative Sampling Methods, and the Hybrid Method.



(a) Test Accuracy of 3 models over Cora Dataset

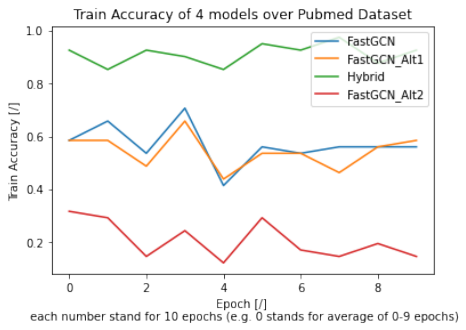


(b) Test Accuracy of 3 models over Cora Dataset

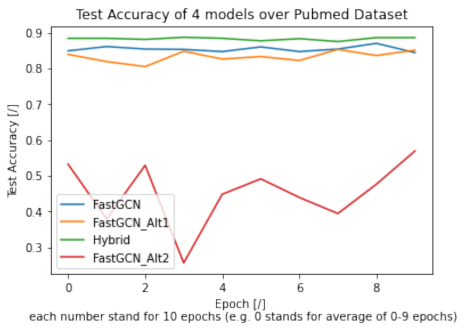


(c) Train Time of 3 models over Cora Dataset

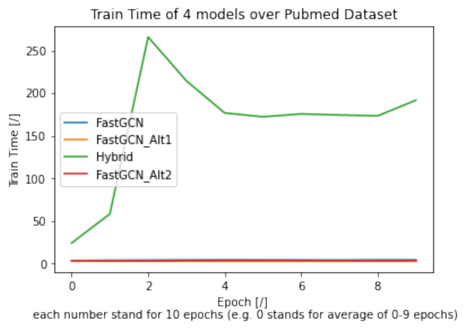
Figure 4: Train/Test Outcome on Cora Dataset



(a) Test Accuracy of 3 models over Pubmed Dataset



(b) Test Accuracy of 3 models over Pubmed Dataset



(c) Train Time of 3 models over Pubmed Dataset

Figure 5: Train/Test Outcome on Pubmed Dataset

6 Discussion and Analysis

One very interesting result that we discovered was that the Logistic Regression, MLP Classifier, and Random Forest all performed extremely well on Pubmed and poorly on Cora. In fact, the Random Forest far exceeded any of our results with Graphical Convolutional Networks, and both our MLP and Hybrid methods were also better. This suggests that Pubmed is not a very good dataset for comparing the performance of Graphical Convolutional Networks, and that a graphical convolutional network is not required for accurate predictions on Pubmed. Our hypothesis for why these simpler models were able to perform well on Pubmed but not on Cora is that the feature set in Pubmed was more descriptive than the feature set in Cora, and that there were fewer categories to choose between for Pubmed (3 categories) than for Cora (7 categories). In Cora, each feature simply represents the presence or absence of a single word in the paper. Information about the frequency with which a word appears in the paper and the rareness of these words is not included. It is likely that many different papers from different academic contexts have the same word appearing at least once, but vary significantly in the frequency with which such words appear. Given that the feature vectors themselves are less informative and that there are more classes to differentiate between, getting information from neighbors and other close by nodes is relatively more important, since neighboring nodes are likely to have the same label. However, in the case of Pubmed, each feature is a TF/IDF of a word in the paper, which provides a lot more information as to the context of the paper and thus its category. Additionally there are only 3 categories to differentiate between. Given this, a Random Forest or MLP Classifier is likely to perform relatively better since most of the information needed to classify the paper is already contained in its feature set, and so aggregating information from the node's neighbors is not very useful.

There are many future steps to consider in this direction. For one, we would want to try our models on more dataset that can be viewed as graphs to get a better understanding of exactly when Random Forests and MLP's outperform GCNs. We would also like to use larger and denser graphs for training and test.

Another interesting result we discovered was that both of our sampling schemes performed worse than the one included in the paper. This suggests that our intuition that we should normalize each row in the adjacency matrix when computing the probabilities for our sampling distribution is wrong. Nodes that have stronger connections in the adjacency matrix (e.g the first node in the first example shown earlier) should have a significantly larger influence in determining the probabilities that their neighbors are selected for the sample. Additionally trying to convert the adjacency matrix into an aperiodic, irreducible transition probability matrix by adding ϵ and normalizing rows might distort and lose information from the adjacency matrix and lead to a poor sampling distribution. It might also be better to deal with each connected component separately than to try and connect the graph by adding ϵ to each entry of the adjacency matrix. Each connected component might have its own stationary stationary distribution and we could potentially try sampling from some weighted combination of these stationary distributions. Furthermore, adding ϵ to each entry makes the adjacency matrix no longer sparse, which will reduce efficiently greatly when we attempt our methods on larger dataset.

One final idea we considered at the end of the project, but did not have time to implement, was perhaps generating samples from a conditional probability distribution that already takes into account samples generated so far. This would lead to the samples not being generated i.i.d. (Independent and Identically Distributed) but may lead to a more representative sample of the entire graph at each layer.

7 Teammates and work division

Before Midway Report:

Sizhe & Keshav: (1) Work with literature review around Graph Convolutional Network. (2) used their code to work on multiple dataset like Cora and Pubmed to see benchmarks.

After Midway Report:

Sizhe & Keshav: (1) More literature review on the topic of Graph Convolutional Network. (2) Brainstormed approaches for modifying sampling distribution in Graph Convolutional paper. Implemented some of the ideas we came up with and analysed results (3) Produce graphs for the report. (4) Write Report.

8 Code

Github Repo

9 References and citations

References

[1] Jie Chen, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling". In: *arXiv preprint arXiv:1801.10247* (2018).

[2] Lei Chen et al. "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 01. 2020, pp. 27–34.

[3] Wei-Lin Chiang et al. "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 257–266.

[4] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[5] Maosen Li et al. "Actional-structural graph convolutional networks for skeleton-based action recognition". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3595–3603.

[6] Xin Liu et al. "Sampling methods for efficient training of graph convolutional networks: A survey". In: *IEEE/CAA Journal of Automatica Sinica* 9.2 (2021), pp. 205–234.

[7] Cheng Wan et al. "BNS-GCN: Efficient Full-Graph Training of Graph Convolutional Networks with Partition-Parallelism and Random Boundary Node Sampling Sampling". In: *Proceedings of Machine Learning and Systems* 4 (2022).

[8] Felix Wu et al. "Simplifying graph convolutional networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6861–6871.

[9] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7370–7377.

[10] Si Zhang et al. "Graph convolutional networks: a comprehensive review". In: *Computational Social Networks* 6.1 (2019), pp. 1–23.