Reproduction of the paper entitled ”Simplifying Graph Convolutional Networks”

Jason Liartis
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
jliartis@ails.ece.ntua.gr

Vassilis Lyberatos
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
vaslyb@ails.ece.ntua.gr

Paraskevi-Antonia Theofilou
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
paristh@ails.ece.ntua.gr

***Abstract*—In this work we studied the paper entitled Simplifying Graph Convolutional Networks which presents a barebones Graph Convolutional Neural Network with the aim of achieving high performance in less demanding applications by limiting the excess complexity and unnecessary calculations of other widespread architectures. Our motivation for studying this particular topic stems from the fact that Graph Convolutional Networks are at the cutting edge of technological development and are encountered in a multitude of applications. More specifically, the topic of Simplifying Graph Convolutional Networks piqued our interest since the complexity of networks, large data sets, their high computational requirements and the time required to complete the various processes require the establishment of simpler networks that guarantee generalized high performance.**

***Index Terms*—graph neural networks, node classification, graph classification**

I. Introduction

Graphs have been widely used to model real-world entities and the relationship among them. Objects can be denoted as nodes of a graph and edges can be used to represent relationship between them. Graphs are used almost in every field [1] and appear in numerous applications domains such as social networks [2], citation networks [3], applied chemistry [4], bioinformatics, neuroscience [5], natural language processing [6] and computer vision [7]. These prove the importance of working with graphs. For this reason, Graph Neural Networks (GNNs) [8] were developed and their extensions and improvements are being studied with the aim of offering the scientific community the appropriate tools for solving such problems.

Graph Convolutional Networks (GCNs) [9] appeared late in the Deep Learning revolution and were inspired by existing deep networks in computer vision such as ResNets [10]. As a consequence they carry significant complexity derived from those networks, which may be unnecessary for less demanding applications on graphs. To simplify these networks and limit their complexity, with the aim of their faster response, it is necessary to remove non-linearities and use a final linear transformation function. The result of these actions is the creation of a linear model corresponding to a fixed low-pass filter followed by a linear classifier. This model shows comparable and many times higher performance (accuracy) compared to state-of-the-art (SOTA) models for a variety of applications, requiring less computing power and fewer parameters to adjust. In addition, this model is intuitively interpretable because of its linear characteristics. We refer to this simplified linear model as Simple Graph Convolution (SGC) [11].

For the evaluation of SGC, the authors of the paper used node classification datasets for citation and social networks, to which the network responded by giving very good results compared to GCN and other SOTA networks. It was also evaluated in other downstream tasks such as text classification, user geolocation, relation extraction and zero-shot image classification, producing remarkable results in these as well. Therefore, SGC responds to both classical and non-classical node classification problems of small and large datasets. We, in turn, reproduced the experiments that were presented in order to become familiar with the subject and tried to apply and extend the proposed network to other tasks and datasets that the authors had not considered.

More specifically, we applied the experiments to datasets related to citation networks, social networks and text classification. Subsequently, we extended the study and evaluation of the algorithm to new datasets that come from the subject of citation and social networks. We also introduced two new problems to solve: image classification and graph classification on fake news data. Thus, we achieved both the reproduction and the extension of the experiments of the studied model (SGC).

II. Related Work

The first references to GNNs were made for learning a graph representation by finding stable states through fixed-point iterations [12], [13]. In 2014, GCNs as a spectral graph-based extension of convolutional networks to graphs were introduced [14]. Afterwards a simplification of GCNs came up, ChebyNets [15], where the authors used graph convolutions using Chebyshev polynomials to remove the computationally expensive Laplacian eigendecomposition. In 2017, a new more light-weighted GCN architecture was proposed by stacking layers of first-order Chebyshev polynomial filters with a redefined propagation matrix S [9]. Some other variants of GCNs came up based on sampling and aggregation [16], [17]. Also a new series of publications was prompted that was using multi-scale information by raising S to higher order [18]–[20]. In 2018, Graph Isomorphism Network [21] tested the ability to distinguish any two graphs and in 2019 more traditional algorithms like random walk were employed for improving the performance of GCNs [22].

Due to over-smoothing [23] and vanishing gradient problems [24], [25] a lot of SOTA models, published the last two years, were based on simple and light-weighted architectures [26], [27]. Many of them used residual connection as a method to tackle deep GNNs complexity [27]–[30] and many others used normalization and regularization techniques [31]–[35]. Enough of them, similar to SGC, used linearity in their architectures [36], [37] as a way to increase the simplicity of their models.

III. Data and Features

For our experiments we used some datasets for node classification in order to evaluate the proposed model. First, the datasets on which the basic evaluation of the models in the paper was performed were examined. Then, new datasets from the citation and social networks space were used. In addition, some of the downstream tasks of the paper were replicated and new ones were put forward for evaluation.

The datasets we examined and used in the paper are: Cora, Citeseer and Pubmed citation network datasets and the Reddit social network dataset.

Cora: The Cora dataset [38] consists of 2708 scientific publications classified into one of seven classes of Machine Learning areas. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words. According to Cora's label split, there are 140 training, 500 validation and 1,000 testing nodes.

CiteSeer: The CiteSeer dataset [38] consists of 3312 scientific publications classified into one of six classes. The citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words. According to CiteSeer's label split, there are 120 training, 500 validation and 1,000 testing nodes.

PubMed Diabetes: The PubMed Diabetes dataset [38] consists of 19717 scientific publications from PubMed database pertaining to diabetes classified into one of three classes. The citation network consists of 44338 links. Each publication in the dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words. According to PubMed's label split, there are 60 training, 500 validation and 1,000 testing nodes.

Reddit: The Reddit dataset is a graph dataset [39] from Reddit posts made in the month of September, 2014. The node label in this case is the community, or "subreddit", that a post belongs to. 50 large communities have been sampled to build a post-to-post graph, connecting posts if the same user comments on both. In total this dataset contains 232,965 posts and 11,600,000 links with an average degree of 492. According to Reddit's label split, there are 152,000 training, 24,000 validation and 55,000 testing nodes.

Text classification [40] is an important and classical problem in natural language processing. Recently, the new research direction of GNNs or graph embeddings has attracted wide attention [41] [42]. This type of networks is effective at tasks thought to have rich relational structure and can preserve global structure information of a graph in graph embeddings. A single large graph is constructed from an entire corpus, which contains words and documents as nodes. The edge between two word nodes is built by word co-occurrence information and the edge between a word node and document node is built using word frequency and word's document frequency. Then, the text classification problem is turned into a node classification problem. The method can achieve strong classification performances with a small proportion of labeled documents and learn interpretable word and document node embeddings. In this paper, SGC was evaluated on text classification datasets like: 20NG, R8, R52, Ohsumed and MR, which follow the refered structure.

The new datasets we studied and belong to the field of citation and social networks are: Ogbn-arxiv citation network dataset, Facebook Page-Page, Twitch, Deezer and GitHub social network dataset. The charteristics of all the citation and social network datasets are described in Tab.I.

Ogbn-arxiv: The ogbn-arxiv dataset [43] is a directed graph where each node is an arXiv paper and each directed edge indicates that one paper cites another one. It consists of 169,343 nodes and 1,166,243 edges. Each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. The embeddings of individual words are computed by running the skip-gram model over the MAG corpus. The task is to predict the 40 subject areas of arXiv Computer Science papers.

Facebook Page-Page: The Facebook Page-Page network dataset [44] contains 22,470 nodes, 342,004 edges, 128 node features and 4 classes. Nodes correspond verified Facebook pages (users) and edges show mutual likes between users. Node features are extracted from the site descriptions. The task is multi-class classification of the site category.

Twitch: The Twitch dataset [44] is a social network of Twitch users which was collected from the public API in Spring 2018. Nodes are Twitch users and edges are mutual follower relationships between them. There are 168,114 nodes, 6,797,557 edges, 128 node features and 2 classes. The associated task is binary classification of whether a streamer uses explicit language.

Deezer: The Deezer dataset [44] is a social network of Deezer users which was collected from the public API in March 2020. Nodes are users of music streamig service Deezer from European countries and edges are mutual follower relationships between them. The vertex features are extracted based on the artists liked by the users. There are 28,281 nodes, 92,752 edges, 128 node features and 2 classes. The task related to the graph is binary node classification - one has to predict the gender of users. This target feature was derived from the name field for each user.

GitHub: The Github dataset [38] is a social network dataset where nodes correspond to developers who have starred at least 10 repositories and edges to mutual follower relationships. There are 37,000 nodes, 578,000 edges, 128 node features and 2 classes. Node features are location, starred repositories, employer and e-mail address. The task is to classify nodes as web or machine learning developers.

In addition, as an extension of the present study, we tried to apply the proposed model to new datasets, which correspond to different tasks from the ones already applied. These are image and graph classification.

Flickr: The Flickr dataset [45] is built by forming links between images sharing common metadata from Flickr. Edges are formed between images from the same location, submitted to the same gallery, group, or set, images sharing common tags, images taken by friends, etc. The original images are collected from PASCAL, ImageCLEF, MIR, and NUS-wide. There are 89,250 nodes and 899,000 edges. As node features is used a 500-dimensional bag-of-word representation. Each image belongs to one of the 7 classes.

Graph classification is a problem with practical applications in many different domains. To solve this problem, certain graph statistics, like graph features are calculated in order to predict the class label of graph. Real world graphs are complex and noisy and these traditional approaches are computationally intensive. With the introduction of the deep learning framework [46] [47], there have been numerous attempts to create efficient classification approaches.

The **UPFD** dataset [48] includes two sets of tree-structured graphs curated for evaluating binary graph classification, graph anomaly detection, and fake/real news detection tasks. The dataset includes fake&real news propagation (retweet) networks on Twitter built according to fact-check information from Politifact and Gossipcop. The news retweet graphs were originally extracted by FakeNewsNet. Each graph is a hierarchical tree-structured graph where the root node represents the news: the leaf nodes are Twitter users who retweeted the root news. A user node has an edge to the news node if someone retweeted the news tweet. Two user nodes have an edge if one user retweeted the news tweet from the other user. Near 20 million historical tweets were crawled from users who participated in fake news propagation in FakeNewsNet to generate node features in the dataset. On the one hand, the Politifact dataset is consisted of 41,054 nodes and 40,740 edges. On the other hand, the Gossipcop dataset is consisted of 314,262 nodes and 308,798 edges respectively.

TABLE I: Dataset statistics of the citation and social networks

Dataset	**# Nodes**	**# Edges**	**# Features**	**# Classes**
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pumbed	19,717	44,338	500	3
Ogbn-arxiv	166K	1M	128	4
Twitch EN	7,126	77,774	128	2
Facebook	22,470	342K	128	4
Deezer	28,281	185K	128	2
GitHub	37,000	578K	128	2
Flickr	89,250	899K	500	7
Reddit	233K	114M	602	41

IV. Methodology

SGC follows the structure of GCNs [9] in the context of node classification with the difference that the whole process between input and output level is reduced to a simple feature propagation step followed by standard logistic regression instead of applying a linear classifier after the repeated transformation of the feature vectors by the K layers.

The models take a graph with some labeled nodes as input and generate label predictions for all graph nodes. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V}$ represents the vertex set consisting of nodes $\{v_1, \ldots, v_n\}$, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric (typically sparse) adjacency matrix where a_{ij} denotes the edge weight between nodes v_i and v_j. A missing edge is represented through $a_{ij} = 0$. The degree matrix $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n)$ is defined as a diagonal matrix where each entry on the diagonal is equal to the row-sum of the adjacency matrix $d_i = \sum_j a_{ij}$.

Each node v_i in the graph has a corresponding $d-$ dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^d$. The entire feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ stacks n feature vectors on top of one another, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$. Each node belongs to one out of C classes and can be labeled with a C-dimensional one-hot vector $\mathbf{y}_i \in \{0,1\}^C$. We only know the labels of a subset of the nodes and want to predict the unknown labels.

To understand the structural difference between GCNs and SGCs we present the steps followed in each case. The Figure 1 illustrate the architecture of each type of layers.

GCNs following CNNs and MLPs structure learn a new feature representation for the feature $\mathbf{x}_i$ of each node over multiple layers, which is subsequently used as input into a linear classifier. For the k-th graph convolution layer, we denote the input node representations of all nodes by the matrix $\mathbf{H}^{(k-1)}$ and the output node representations $\mathbf{H}^{(k)}$. Naturally, the initial node representations are just the original input features:

$$\mathbf{H}^{(0)} = \mathbf{X}, \tag{1}$$

which serve as input to the first GCN layer.

In each layer of GCNs, node representation follow three stages: feature propagation, linear transformation and nonlinear transition.

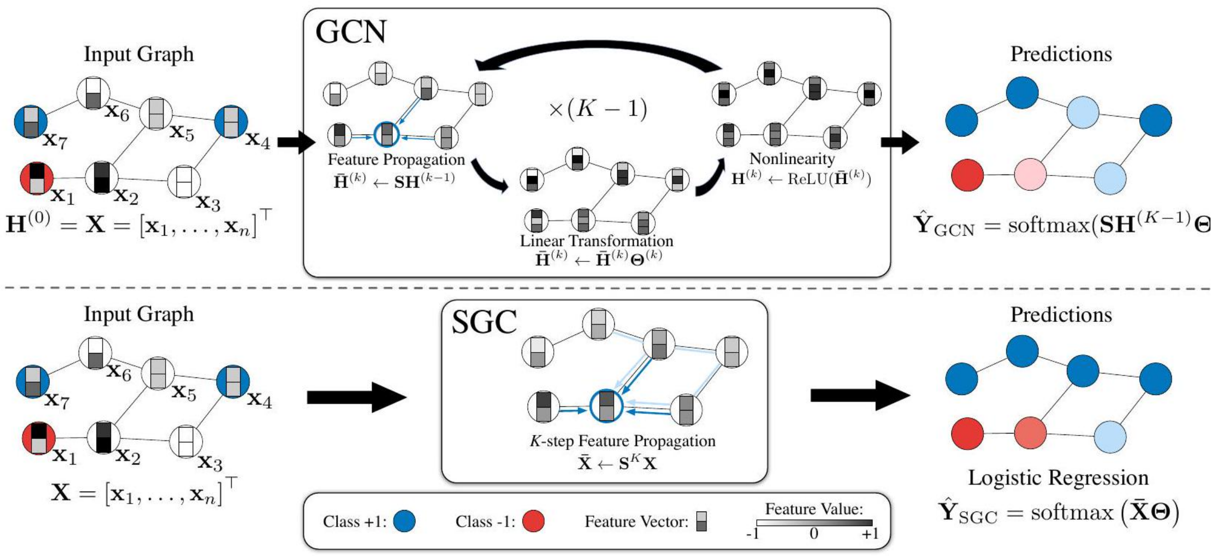
Input Graph
$\mathbf{x}_6$
$\mathbf{x}_5$
$\mathbf{x}_7$
$\mathbf{x}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{H}^{(0)} = \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$
GCN
$\times(K-1)$
Feature Propagation
$\bar{\mathbf{H}}^{(k)} \leftarrow \mathbf{S}\mathbf{H}^{(k-1)}$
Nonlinearity
$\mathbf{H}^{(k)} \leftarrow \mathrm{ReLU}(\bar{\mathbf{H}}^{(k)})$
Linear Transformation
$\bar{\mathbf{H}}^{(k)} \leftarrow \bar{\mathbf{H}}^{(k)}\mathbf{\Theta}^{(k)}$
Predictions
$\hat{\mathbf{Y}}_{\text{GCN}} = \mathrm{softmax}(\mathbf{S}\mathbf{H}^{(K-1)}\mathbf{\Theta}$
Input Graph
$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$
SGC
K-step Feature Propagation
$\bar{\mathbf{X}} \leftarrow \mathbf{S}^K\mathbf{X}$
Predictions
Logistic Regression
$\hat{\mathbf{Y}}_{\text{SGC}} = \mathrm{softmax}\left(\bar{\mathbf{X}}\mathbf{\Theta}\right)$
Class +1: ●
Class -1: ●
Feature Vector: ▯
Feature Value:
-1
0
+1

Fig. 1: Schematic layout of a GCN and a SGC

According to feature propagation at the beginning of each layer the features $\mathbf{h}_i$ of each node v_i are averaged with the feature vectors in its local neighborhood,

$$\overline{\mathbf{h}}_i^{(k)} \leftarrow \frac{1}{d_i+1}\mathbf{h}_i^{(k-1)} + \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)} \quad (2)$$

More compactly, we can express this update over the entire graph as a simple matrix operation. Let $\mathbf{S}$ denote the "normalized" adjacency matrix with added self-loops,

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \quad (3)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. The simultaneous update in Equation 2 for all nodes becomes a simple sparse matrix multiplication

$$\overline{\mathbf{H}}^{(k)} \leftarrow \mathbf{S}\mathbf{H}^{(k-1)}. \quad (4)$$

Intuitively, this step smoothes the hidden representations locally along the edges of the graph and ultimately encourages similar predictions among locally connected nodes.

Following the stages of feature transformation and nonlinear transition, the smoothed hidden feature representations are transformed linearly given the fact that each layer is associated with a learned weight matrix $\Theta^{(k)}$. Then, a nonlinear activation function such as ReLU is applied pointwise before outputting feature representation $\mathbf{H}^{(k)}$. Thus, the representation updating rule of the k-th layer is:

$$\mathbf{H}^{(k)} \leftarrow \mathrm{ReLU}\left(\overline{\mathbf{H}}^{(k)}\Theta^{(k)}\right). \quad (5)$$

The pointwise nonlinear transformation of the k-th layer is followed by the feature propagation of the $(k+1)$-th layer.

For node classification, the last layer of a GCN predicts the labels using a softmax classifier. Denote the class predictions for n nodes as $\hat{\mathbf{Y}} \in \mathbb{R}^{n\times C}$ where $\hat{y}_{ic}$ denotes the probability of node i belongs to class c. The class prediction $\hat{\mathbf{Y}}$ of a K-layer GCN can be written as:

$$\hat{\mathbf{Y}}_{\text{GCN}} = \mathrm{softmax}\left(\mathbf{S}\mathbf{H}^{(K-1)}\mathbf{\Theta}^{(K)}\right), \quad (6)$$

where softmax $(\mathbf{x}) = \exp(\mathbf{x})/\sum_{c=1}^{C}\exp(x_c)$ acts as a normalizer across all classes.

Similar to CNNs, GCNs' hidden layers increase the receptive field of internal features [49], given that in each layer the hidden representations are averaged among neighbors that are one hop away, which means that each node obtains feature information from all nodes that are k-hops away in the graph. Thus every node has more information about the whole network after every layer. This is the part which is extracting features from the neighbors. The deeper layers increase the expressivity of MLPs as feature hierarchies are created, however it is not all that beneficial [50]. Each layer has a non-linearising function but it does not add anything to the receptivity of the network.

In SGC, the goal is the removal of non-linearities and the application of a linear classifier at the end. The benefit arises from the local average, then the non-linearities introduce unnecessary complexity. For this reason, SGC follow two stages: linearization and classification with logistic regression.

The nonlinear transition functions between each layer are removed and only the final softmax is kept (in order to obtain probabilistic outputs). The resulting model is linear and more simple, but still has the same increased "receptive field" of a K-layer GCN,

$$\hat{\mathbf{Y}} = \mathrm{softmax}\left(\mathbf{S}\ldots\mathbf{S}\mathbf{S}\mathbf{X}\Theta^{(1)}\mathbf{\Theta}^{(2)}\ldots\Theta^{(K)}\right) \quad (7)$$

To simplify notation they collapse the repeated multiplication with the normalized adjacency matrix $\mathbf{S}$ into a single matrix by raising $\mathbf{S}$ to the K-th power, $\mathbf{S}^K$. Further, they reparameterize the weights into a single matrix $\Theta = \Theta^{(1)}\Theta^{(2)}\dots\Theta^{(K)}$. The resulting classifier becomes

$$\hat{\mathbf{Y}}_{\mathrm{SGC}} = \mathrm{softmax}\left(\mathbf{S}^K\mathbf{X}\boldsymbol{\Theta}\right), \tag{8}$$

which is refered as SGC.

According to the Equation 8, a natural and intuitive interpretation of SGC is presented: by distinguishing between feature extraction and classifier, SGC consists of a fixed (i.e., parameter-free) feature extraction/smoothing component $\overline{\mathbf{X}} = \mathbf{S}^K\mathbf{X}$ followed by a linear logistic regression classifier $\hat{\mathbf{Y}} = \mathrm{softmax}(\overline{\mathbf{X}}\Theta)$. Since the computation of $\overline{\mathbf{X}}$ requires no weight it is essentially equivalent to a feature pre-processing step and the entire training of the model reduces to straight-forward multi-class logistic regression on the pre-processed features $\overline{\mathbf{X}}$.

From a graph perspective, SGC corresponds to a fixed filter on the graph spectral domain. The self-loops which were added to the original graph as a renormalization trick [9], shrinks the underlying graph spectrum effectively.

V. Experiments/Results/Discussion

We reproduced the main experiments proposed from the paper. Firstly we conducted the experiments concerning social and citation networks and got familiarized with the provided GitHub repository[1]. Afterwards we carried out experiments about the downstream task, text classification. The majority of our experiments had the same results with the original work (see Tab.II), indicating the reproducibility of the paper.

TABLE II: Test accuracy (%) on citation networks

	Cora	Citeseer	Pubmed
Numbers from literature:			
GCN	81.4	70.9	79.0
GAT	83.3	72.6	78.5
SGC	81.0	71.9	78.9
Our experiments:			
GCN	80.5	67.6	79.2
GAT	83.2	67.6	78.5
SGC	81.0	71.9	78.9

In order to extend the work and test the proposed architectures we had experiments with different kind of datasets and tasks. In order to follow the proposed order of the experiments, firstly we experimented with a new citation dataset (Ogbn-arxiv) and with new social network datasets (Facebook, Twitch, Github and Deezer). For exploring new downstream tasks, we conducted experiments with a dataset for image classification (Flickr) and with two datasets for graph classification (Politifact and Gossipcop).

For our experiments we used four different GCNs. We chose a classical GCN architecture, the simplified SGC architecture, the attention-based GAT [51] architecture and the representation learning based SAGE [52] architecture.

We had a certain setting in our experiments. For the experiments conducted on Cora, Citeseer, Pumbed, Reddit, Facebook, Twitch, Deezer, GitHub and Flickr dataset we used three different models: GCN, GAT and SGC. We chose the following hyper-parameters for the the training setup: 0.2 *learning rate*; 0.005 *weight decay*; 100 *epochs*, 2 *layers* and 16 *hidden channels*. For the text classification task we had the following hyper-parameters: 3 *epochs*; 16 *batch size* and we used for each task the *weight decay* proposed from the literature. For the graph classification task (Politifact and Cossipcop) we used: 60 *epochs*, 2 *layers*, 0.01 *learning rate*, 0.01 *weight decay* and 128 *hidden channels*. For this task we tried all the above mentioned architectures (SGC, GCN, SAGE and GAT).

For implementing the GCNs we used the PYG (PyTorch Geometric) library [53]. It is a library built upon PyTorch to easily write and train GCNs for a wide range of applications related to structured data. It is worth mentioning that the experiments were conducted in a GeForce GTX 1080 GPU. We also provide the GitHub repository[2].

Observing the results of our experiments we confirm the claim of the paper that sometimes SGC achieves performance close to SOTA. This is demonstrated in both the experiments conducted in graph classification datasets (see Tab III) and social network datasets (see Tab IV).

TABLE III: Test accuracy (%) on fakenews dataset (graph classification)

	Politifact	Gossipcop
Numbers from literature:		
GCN	83.2	95.1
GAT	-	96.5
SAGE	84.6	97.5
Our experiments:		
GCN	84.1	96.0
GAT	82.8	96.5
SAGE	81.9	96.6
SGC	83.2	96.0

The fact that SGC performs better to simple problems is observed in our experimental results. Taking into account the results of our models in Flickr dataset (89.250 nodes, 899K edges and 7 classes) and in Twitch dataset (7.126 nodes, 77.774 edges and 2 classes), the SGC model compare to GCN performs better to the low-complexity Twitch task.

Besides trying diverse GCN models in different datasets we experimented with a different structure of the SGC architecture. The idea of the new architecture was to reduce the dimensions of the node features in order to optimize the performance and produce node embeddings. We added a linear

[1]https://github.com/Tiiiger/SGC

[2]https://github.com/vaslyb/simplifying-graph-convolutional-networks-reproduction

TABLE IV: Test accuracy (%) on social network datasets

	Flickr	Arxiv	Facebook	GitHub	Twitch	Deezer
Numbers from literature:						
GCN	49.2	71.4	93.2	86.5	69.5	63.5
Our experiments:						
GCN	42.3	49.1.9	90.8	87.1	65.3	63.4
SGC	49.8	45.9	89.8	86.4	63.7	62.5

regression layer for exporting node embeddings from node features and train the new altered SGC architecture. The results were not satisfactory enough, achieving lower accuracy than the original SGC architecture.

VI. Conclusions/Future Work

The methodology of the SGC neural network presents an intuitive, simple, and expressive formulation for learning the latent representations of the nodes labels, which builds upon the general theory of graph convolutional networks. What makes this methodology appealing is that the operations are linear between the adjacency matrix, the features, and the parameters prior to the use of the softmax function. This makes it an ideal candidate to work with in exploring different applications of its formulation as the feature projections are linear and that the adjacency matrix is clearly an operation aggregating feature information of the vicinity of nodes (number of 'hops').

SGC as a simplification of GCNs offer high efficiency by combining its low complexity with a short response time. This is evident from the results obtained from the experiments of both the paper to be studied and our own. This model seems to be a good starting point, like a baseline model, for later related studies and extensions to graph representation problems.

Future work could proceed by examining how representations can be made to handle known variable interactions in the feature matrix and how to perform dimensionality reduction on the feature set needed from the nodes.

Another exploration of the SGC could be to study how nodes' labels can be changed based on the size of the influence neighborhood. More specifically, could be introduced an approach which allows different nature of a simulation of node label states based on the number of walks taken between nodes which influence each other. This can be useful for answering questions regarding which users once included in a field of influence will alter a node's category label.

References

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2018. [Online]. Available: https://arxiv.org/abs/1812.08434

[2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[3] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[4] S. Ye, J. Liang, R. Liu, and X. Zhu, "Symmetrical graph neural network for quantum chemistry with dual real and momenta space," *The Journal of Physical Chemistry A*, vol. 124, no. 34, pp. 6945–6953, 2020.

[5] A. Bessadok, M. A. Mahjoub, and I. Rekik, "Graph neural networks in network neuroscience," 2021. [Online]. Available: https://arxiv.org/abs/2106.03535

[6] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," vol. 33, pp. 7370–7377, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4725

[7] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," 2018. [Online]. Available: https://arxiv.org/abs/1803.08035

[8] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A gentle introduction to graph neural networks," *Distill*, 2021, https://distill.pub/2021/gnn-intro.

[9] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[11] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[12] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, no. 2005, 2005, pp. 729–734.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[14] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[15] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[16] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

[17] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[18] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[19] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," in *uncertainty in artificial intelligence*. PMLR, 2020, pp. 841–851.

[20] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," *arXiv preprint arXiv:1901.01484*, 2019.

[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[22] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[23] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[24] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[25] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "Deepergcn: All you need to train deeper gcns," *arXiv preprint arXiv:2006.07739*, 2020.

[26] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[27] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725–1735.

[28] S. Gong, M. Bahri, M. M. Bronstein, and S. Zafeiriou, "Geometrically principled connections in graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 415–11 424.

[29] K. Xu, M. Zhang, S. Jegelka, and K. Kawaguchi, "Optimization of graph neural networks: Implicit acceleration by skip connections and more depth," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 592–11 602.

[30] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *International conference on machine learning*. PMLR, 2021, pp. 6437–6449.

[31] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[32] A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. Narayanan, and X. Qian, "Bayesian graph neural networks with adaptive connection sampling," in *International conference on machine learning*. PMLR, 2020, pp. 4094–4104.

[33] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," *arXiv preprint arXiv:1909.12223*, 2019.

[34] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," *arXiv preprint arXiv:1905.10947*, 2019.

[35] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," *Advances in neural information processing systems*, vol. 33, pp. 4917–4928, 2020.

[36] C. Cai and Y. Wang, "A simple yet effective baseline for non-attributed graph classification," *arXiv preprint arXiv:1811.03508*, 2018.

[37] E. Buchnik and E. Cohen, "Bootstrapped graph diffusions: Exposing the power of nonlinearity," in *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 2018, pp. 8–10.

[38] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[39] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[40] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7370–7377.

[41] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018. [Online]. Available: https://arxiv.org/abs/1806.01261

[42] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," 2017. [Online]. Available: https://arxiv.org/abs/1709.07604

[43] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[44] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, p. cnab014, 2021.

[45] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[46] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," 2017. [Online]. Available: https://arxiv.org/abs/1707.05005

[47] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2014–2023. [Online]. Available: https://proceedings.mlr.press/v48/niepert16.html

[48] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun, "User preference-aware fake news detection," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2051–2055.

[49] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 447–456.

[50] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 646–661.

[51] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[52] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[53] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.