Evaluation of Simple and Deep Graph Convolutional Network Variants

Taylor Warner [1]

Abstract

Graph Convolutional Networks (GCNs) are a type of neural network intended to operate on graph structured data. There has been extensive research in the last several years on GCNs. This paper explores simplified and deep variations of semi-supervised classification GCNs, evaluates claims accompanying each approach, and explores the re-implementation of a simplified GCN with tests extended across citation and page-page graph networks. The simplified GCN achieves similar node-classification accuracy and requires only a fraction of the training time.

1. Introduction

Graph data structures represent one of the foundational abstract data types of computer science with a wide variety of implementations and applications. Because real world datasets are typically stored in graph structures, there has been perpetual interest in machine learning applications for graphs over the past two decades. Graph neural networks are interesting because there is current research investigating alternative designs and layer architecture.

This paper provides the background of one message passing neural network (MPNN) architecture, the Graph Convolutional Network (GCN), and the review of three simple and deep variants: Simple Graph Convolution (SGC), Graph Convolutional Network via Initial residential and Identity mapping (GCNII), and a Geometric graph structure focused GCN (Geom-GCN). Following the reviews, this paper explores a re-implementation of SGC and GCN with extended testing across citation network datasets and page-page datasets to compare against metrics in the reviewed papers.

[1]Purdue University, West Lafayette, Indiana, United State. Correspondence to: <
>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Review of SGC

2.1. Storyline

Motivation Following the motivations outlined in the Introduction, graph data structures have many real world applications. Because of the history of GCN development with deep neural networks, there are potential performance and efficiency improvements that would translate across to the GCN applications.

Prior Work Initial theoretical attempts towards semi-supervised learning of graph data had a wide range of methods. In 2005, Ph.D. candidate Xiaojin Zhu published his Ph.D. thesis "Semi-Supervised Learning with Graphs," which laid the groundwork for learning graph structures and included mathematical conversions to kernel matrices from graph Laplacian matrices (Zhu, 2005). Zhu proved one can construct graph kernels from the spectral decomposition of graph Laplacians, which changes the data classification problem into a convex optimization problem and does not have the potential convergence to a local minima faced with Laplacians. Zhu tested these mathematical formulas using document categorization tasks on seven popular 20-newsgroups.

Research later published in NeurIPS 2009 demonstrated the Laplacian Regularization method and Laplacian Eigenvector methods could not handle increasing data sizes (Nadler et al., 2009). This research confirmed the need for some kind of spectral decomposition of graph Laplacians in order to learn graph data structures. For reader context, spectral decomposition of graph Laplacians can be compared to performing dimensionality reduction of multidimensional data for easier label grouping and classification.

Between 2009 and 2014, key research into graph structured data pushed the kernel model forward as the graph neural network model (Scarselli et al., 2009) with model evaluation using spectral networks (Bruna et al., 2013). In 2016, Thomas Kipf and Max Welling introduced a semi-supervised classification model of Graph Convolutional Networks (GCNs) as a scalable approach to Convolutional Neural Networks (CNNs) (Kipf & Welling, 2016). Because GCNs can be used to learn representations of graphs similar

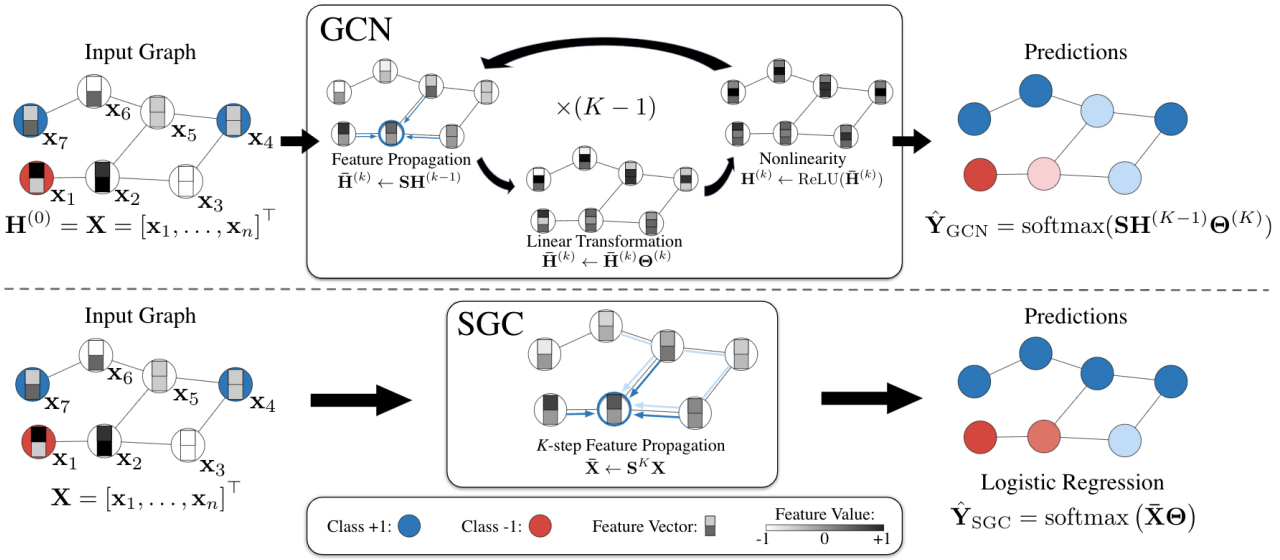

Figure 1. Schematic layout of GCN and SGC from *Figure 1* of SGC. GCN is shown to have multiple transforms and a final representation with the linear classifier. SGC is show with a feature propagation step and a final representation with logistic regression.

to how CNNs learn representations of images, this model popularized following the ICML publication.

As GCNs provide an efficient variant of CNNs on graphs, the applications include citation networks (Kipf & Welling, 2016), social networks (Chen et al., 2018), natural language processing (Yao et al., 2019), and computer vision (Wang et al., 2018).

Research Gap Because the previously listed research groups furthered the development of GCNs during the push for deep learning with non-linear models, GCNs inherited a non-linear structure. There was little to no published research of GCNs developed and tested using only linear models by 2018.

Contributions Wu et al. presented the Simple Graph Convolution (SGC) capable of achieving comparable node classification performance to GCN and other graph neural networks and up to two orders of magnitude faster on the largest dataset in their evaluation (2019).

2.2. Proposed Solution

The SGC authors argue most real world classifiers operate linearly and the additional non-linear complexity should only be used when simpler methods are inadequate (Wu et al., 2019). GCNs are built using multi-layer neural networks and can be reduced into a single linear transformation, as shown in Figure 1. By removing non-linear activation layers, the SGC is shrinks to a linear model with a graph spectral domain similar to a low-pass filter, reflecting previous research done by Bruna, et al. (2013).

GCN Solution Background With K denoting a number of hidden layers, K-layer GCNs focuses on three main steps prior to the prediction for each layer K: feature propagation, linear transformation and non-linearity. The initial node representations, $\mathbf{H}^{(0)}$, are the input features given as the input matrix $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n feature vectors and d-dimensions. For each layer K, the input node is represented by the matrix $\mathbf{H}^{(k-1)}$ and the output node by $\mathbf{H}^{(k)}$. The feature propagation step is where each node v_i is averaged with the local feature vectors, which can be expressed in vector form over the entire graph as sparse matrix multiplication as shown in equation (1). Let $\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ given $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. The linear transformation step is where each layer is transformed linearly with a learned matrix $\mathbf{\Theta}^{(k)}$ below in equation (2). The non-linear step is applied with the ReLU non-linear activation function in equation (3). These equations can be combined for each K-layer prediction with the $softmax$ classifier below in equation (4).

$$\bar{\mathbf{H}}^{(k)} \leftarrow \mathbf{S}\mathbf{H}^{(k-1)} \tag{1}$$

$$\bar{\mathbf{H}}^{(k)} \leftarrow \bar{\mathbf{H}}^{(k)}\mathbf{\Theta}^{(k)} \tag{2}$$

$$\mathbf{H}^{(k)} \leftarrow \mathrm{ReLU}(\bar{\mathbf{H}}^{(k)}) \tag{3}$$

$$\hat{\mathbf{Y}}_{\mathrm{GCN}} = softmax(\mathbf{S}\mathbf{H}^{K-1}\mathbf{\Theta}^{K}) \tag{4}$$

Table 1. Data from Table 2 of the SGC paper, specifically the bottom portion representing **Their experiments**. *Table 2* caption: *Test accuracy (%) averaged over 10 runs on citation networks.*

	CORA	CITESEER	PUBMED
GCN	81.4 ± 0.4	70.9 ± 0.5	79.0 ± 0.4
GAT	83.3 ± 0.7	72.6 ± 0.6	78.5 ± 0.3
FASTGCN	79.8 ± 0.3	68.8 ± 0.6	77.4 ± 0.3
GIN	77.6 ± 0.3	66.1 ± 0.9	77.0 ± 1.2
LNET	80.2 ± 3.0	67.3 ± 0.5	78.3 ± 0.6
ADALNET	81.9 ± 1.9	70.6 ± 0.8	77.8 ± 0.7
DGI	82.5 ± 0.7	71.6 ± 0.7	78.4 ± 0.7
SGC	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0

Simple Graph Convolution By removing the ReLU nonlinear activation function between each layer, the resulting model is linear, see equation (5). This expression can be simplified by collapsing the repeated multiplication of the normalized adjacency matrices $\mathbf{S}$ into a single matrix $\mathbf{S}^K$ and the weight matrices $\mathbf{\Theta}^i$ into a single matrix $\mathbf{\Theta}$ as denoted in equation (6). The authors split the feature extraction and smoothing, denoted as $\bar{\mathbf{X}} = \mathbf{S}^K\mathbf{X}$, from the linear logistic regression classifier $\hat{\mathbf{Y}} = softmax(\bar{\mathbf{X}}\mathbf{\Theta})$ to optimize their implementation. This separation allows for logistic regression on the pre-processed features using stochastic gradient descent.

$$\hat{\mathbf{Y}} = softmax(\mathbf{S}...\mathbf{SSX}\mathbf{\Theta}^1\mathbf{\Theta}^2...\mathbf{\Theta}^K) \quad (5)$$

$$\hat{\mathbf{Y}}_{\text{SGC}} = softmax(\mathbf{S}^K\mathbf{X}\mathbf{\Theta}) \quad (6)$$

The SGC authors discussed the spectral analysis comparisons of GCN and SGC; however, because it does not change the proposed solution, it will not be discussed in this review.

2.3. Claims-Evidence

SGC was tested across different types of datasets, including citation networks, social networks, text classification, semi-supervised user geolocation, relation extraction, zero-shot image classification, and graph classification. Due to the extensive nature of this research and the amount of training and testing time required to collect the pertinent data, this paper will focus solely on the first type of dataset: citation networks. The authors of SGC claim the SGC can match the performance of GCN and state-of-the-art graph networks on commonly tested citation networks while reducing the training time. Cora, Citeseer, and Pubmed were the three main citation network datasets tested (Sen et al., 2008). They extended citation network tests with a custom Reddit dataset, but GCN ran into Out of Memory (OOM) issues during training.

Claim 1 The authors of SGC claim the SGC can match the test accuracy of GCN on the Cora citation network.

Evidence 1 Given Table 1, which represents the data in Table 2 from the SGC paper, the SGC authors demonstrate test accuracy of the Cora citation network differs by $0.4 \pm 0.4\%$ between SGC and GCN. SGC provides comparable performance to GCN given the SGC author's implementations of both SGC and GCN.

Claim 2 The authors of SGC claim the SGC can match the test accuracy of GCN on the Pubmed citation network.

Evidence 2 Similarly to Evidence 1 and given Table 1, the SGC authors demonstrate test accuracy of the Pubmed citation network differs by $0.1 \pm 0.1\%$ between SGC and GCN. SGC provides comparable performance to GCN given the SGC author's implementation of both SGC and GCN.

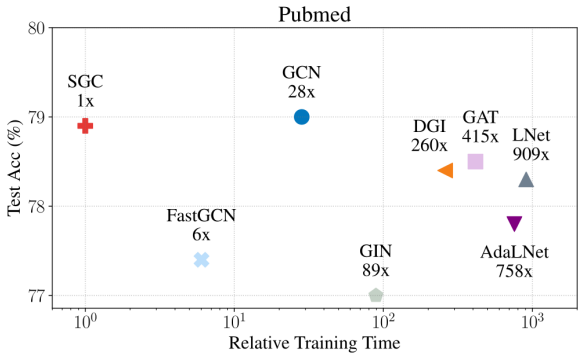

Figure 2. Performance over training time on the Pubmed dataset taken from part of *Figure 3* of SGC. SGC is the fastest graph network to train while achieving competitive performance.

Claim 3 The authors of SGC claim the SGC outperforms the GCN training time of the Pubmed dataset with comparable test accuracy.

Evidence 3 Figure 2 shows the SGC author's Pubmed measured training time for SGC and other graph networks and supports Claim 3. Figure 2 shows the SGC has a relative training time gain of 28x over GCN while producing comparable test accuracy.

Critique and Discussion The SGC provided a novel approach at the time of publication. On initial review of Table 1, the SGC test accuracy variation is very low compared to the other graph networks and may indicate the dataset was modified for better performance. Regarding Figure 2, the Pubmed dataset may have been selected over other datasets due to a larger training time gap between SGC and other graph networks. These decisions indicate selection bias to highlight the best scenarios instead of nominal or expected performance.

3. Review of GCNII

3.1. Storyline

Motivation Following a similar motivation to SGC, there has been continued research interest in applying learning networks to graph data structures due to the array of applications.

Prior Work The prior work is similar to SGC with references to additional models, such as GAT (Veličković et al., 2018), to discuss the problem of over-smoothing. Previous research done by Li et al. indicates stacking non-linear layers degrades accuracy as the graph nodes converge during training (2018). Residual connections can be effective for training deep neural nets, but deeper GCN and GAT models with 4 to 32 layers performed worse than the 2 layer model.

Research Gap While extending the two layer GCN and GAT models does not improve accuracy, additional modifications used in other deeper neural nets had not yet been explored.

Contributions Chen et al. propose the Graph Convolutional Network via Initial residual and Identity mapping (GCNII) model to solve over-smoothing problem (2020). The authors also provide the theoretical analysis behind multi-layer GCN models compared to multi-layer GCNII models.

3.2. Proposed Solution

By combining initial residual, which constructs a skip connection from the input layer, with identity mapping, which adds an identity matrix to the weight matrix, the authors' hypothesis indicates this would prevent over-smoothing and provide higher accuracy with more layers. From the GCNII paper, the formal definition of the l-th layer of GCNII in equation (7), where α allows for each layered node to contain value from the initial input (initial residual) and β allows adding a scaled identity matrix to the weight matrix (identity mapping). $\mathbf{P}$ is the same as $\mathbf{S}$ from Section 1.2, such that $\mathbf{P} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$.

$$\mathbf{H}^{(l+1)} = \sigma(((1-\alpha_l)\tilde{\mathbf{P}}\mathbf{H}^{(l)} + \alpha_l\mathbf{H}^{(0)})((1-\beta_l)\mathbf{I}_n + \beta_l\mathbf{W}^{(l)})) \tag{7}$$

Equation (7) can be directly compared to equations (1) through (4) of the original GCN solution. The inclusion of α and β allow for scaling the impact of the previous layer and direct manipulation of the weight matrix impact per layer.

3.3. Claims-Evidence

The GCNII authors test GCNII on three previously mentioned standard citation networks: Cora, Citeseer, and Pubmed, and extend the classification tests to four new graph networks: Chameleon, Cornell, Texas, and Wisconsin. Chameleon is a page-page network on specific topics on Wikipedia (Rozemberczki et al., 2021). Cornell, Texas, and Wisconsin are webpage datasets called WebKB[1] and collected from computer science departments of three universities by Carnegie Mellon University (Pei et al., 2020).

Claim 1 The authors claim the GCNII model, with 16, 32, or 64 layers, outperforms both GCN and GAT on classification of the three previously mentioned standard citation networks: Cora, Citeseer, and Pubmed.

Evidence 1 Table 2 in the GCNII paper contains a summary of classification accuracy with GCN and GAT performing similarly to Table 1 results from the SGC authors. However, the table shows GCNII and a GCNII variant outperforming both GCN and GAT on all citation networks. GCNII (non-variant) results show 85.5% on Cora using 64 layers, 73.4% on Citeseer using 32 layers, and 80.2% on Pubmed using 16 layers, compared to GCN with 81.5% on Cora, 71.1% on Citeseer, and 79.0% on Pubmed.

Claim 2 The authors claim the GCNII model outperforms both GCN and GAT on the page-page graph network Chameleon.

Evidence 2 Table 5 in the GCNII paper contains a mean classification accuracy of full-supervised node classification. GCN has a 28.18% mean classification accuracy, GAT has a 42.93% mean classification accuracy, GCNII with 8 layers has a 60.61% mean classification accuracy, and a variant labelled GCNII* with 8 layers has a 62.48% mean classification accuracy.

Claim 3 The authors claim the GCNII model outperforms both GCN and GAT on the three WebKB webpage graph networks: Cornell, Texas, and Wisconsin.

Evidence 3 From Table 5 in the GCNII paper, GCN and GAT mean classification accuracy ranges between 45% to 59% across the three webpage graph networks. GCNII and the variant GCNII* range between 69% and 81%.

Critique and Discussion The results shown in Table 3 and Table 5 of the GCNII paper show a significant increase in classification accuracy across multiple layers compared to the GCN and GAT, but the paper is missing any discussion on training and testing time. Some neural nets can significantly increase training time required when increasing the number of layers, so moving from a 2 layer model to

[1] https://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

a 64 layer model introduces nontrivial timing requirements changes. As for the mean classification accuracy in Table 5, some of the measurement results for different models, including GCN, do not match the original publication or other research papers rerunning the results. The authors discuss creating 10 random splits for training, validation, and testing, but do not mention how the tests are setup, such as the number of runs, the number of epochs, etc. The results in this paper are promising; however, not discussing the training time on a 64 layer variant compared to a 2 layer baseline indicates a gross oversight.

4. Review of Geom-GCN

4.1. Storyline

Motivation Following a similar motivation as both SGC and GCNII, there has been continued interest in variants of graph convolutional networks. GCN is a type of message-passing neural network (MPNN) because each node sends a feature representation (or "message") to neighborhood nodes, such that individual nodes are defined by the adjacent nodes of the neighborhood. The Geom-GCN authors, Pei et al., explore the weaknesses of the MPNN applications to graph structured data, including the model losing node neighborhood structural data and long-range node dependencies during training, and potential improvements using geometric structure method embedding (2020).

Prior Work The Geom-GCN authors cite similar research to the previous two papers, SGC and GCNII, regarding the GCN development, but also additional work done on graph geometry for disassortative graphs (Newman, 2002).

Research Gap Going beyond model depth or minor adjustments, the authors discuss a lack of research regarding graph geometry in different MPNN applications to graph data, such as GCN.

Contributions The main contributions include the background and implementation of multiple geometric aggregation schemes for GCN with measured benchmark comparisons to other graph networks.

4.2. Proposed Solution

In order to overcome the weaknesses of MPNNs, the Geom-GCN authors propose a geometric aggregation scheme as part of the graph convolutional network. There are three main components to the geometric aggregation scheme: 1. node embedding; 2. structural neighborhood; and 3. bi-level aggregation. Node embedding involves using a mapping function from a node $v \in V$ in the graph $G = (V, E)$ to a representation vector z and can be represented as $f : v \rightarrow z_v$ where $z_v \in \mathbb{R}^d$. Structural neighborhood involves using a relational operator τ defined in the latent space, such that $\tau : (z_v, z_u) \rightarrow r \in R$ where R is a set of geometric relationships. Bi-level aggregation involves using both high level and low level aggregation functions to update hidden features of the nodes in the neural network layer. The overall aggregation is shown in equation (8) where ReLU is the non-linear activation function σ, $\mathbf{W}_l$ is the weight matrix, and $\mathbf{e}$ are virtual nodes.

$$\mathbf{h}_v^{l+1} = \sigma(\mathbf{W}_l * \underset{i \in g, sr \in R}{||} \mathbf{e}_{(i,r)}^{v,l+1}) \tag{8}$$

4.3. Claims-Evidence

Claim 1 The authors claim Geom-GCN produces competitive results for node-label classification tasks of the three previously mentioned standard citation networks: Cora, Citeseer, and Pubmed.

Evidence 1 The Geom-GCN authors constructed three Geom-GCN variants with different embedding methods for experimentation. Table 3 from the Geom-GCN paper shows the mean classification accuracy from GCN, GAT, and the three Geom-GCN variants. Cora results are similar across the five models, but the Geom-GCN-I variant uses the Isomap embedding method and outperforms GCN and GAT on Citeseer with a 5% improvement and Pubmed datasets with 2% improvement.

Claim 2 The authors claim Geom-GCN outperforms GCN and GAT on node-label classification tasks for page-page datasets, including two page-page networks on specific Wikipedia topics (Chameleon and squirrel) and the WebKB[2] with Cornell, Texas, and Wisconsin datasets mentioned in the review of GCNII.

Evidence 2 Table 3 from the Geom-GCN paper shows selected improvements by Geom-GCN variants compared to GCN and GAT. Geom-GCN-P, which uses the Poincare embedding method, shows the highest performance across the page-page datasets. GCN and GAT have low accuracy on the Chameleon and squirrel datasets; for squirrel, GCN accuracy measures only 23.96% and GAT accuracy measures only 30.03%. Geom-GCN-P demonstrates a significant improvement with a measured accuracy of 60.90% on this same dataset.

Claim 3 The authors claim Geom-GCN is a flexible network that can be combined using arbitrary embedding spaces.

Evidence 3 There are three embedding methods mentioned in the initial experimentation setup: Isomap (Geo-

[2] https://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

GCN-I), Poincare (Geom-GCN-P), and struc2vec (Geom-GCN-S). In Section 4.3.2 of the Geom-GCN paper, the authors demonstrate the combination of arbitrary embeddings with a selection of tests on double embeddings. However, the measured mean classification accuracy is lower on almost all of the datasets in Table 5 compared to the single embedding method results from Table 3, both tables from the Geom-GCN paper. Geom-GCN can be extended to multiple embedding spaces, but there may be a performance cost depending on dataset underlying graph structure compared to the embedding methods used.

Critique and Discussion Section 4.3.3 of the Geom-GCN paper discusses time complexity analysis mostly in terms of the input, hidden units, and virtual nodes, and a running time comparison between GCN, GAT, and Geom-GCN. While there are improvements to node-label classification task accuracy for certain datasets using Geom-GCN over the original GCN, the running time comparison provides further insight into the comparison. Almost all of the datasets show GAT and Geom-GCN an order of magnitude slower than GCN. The running time cost can be as important or more important than minor accuracy improvements in most situations, so this was a great addition to see in this paper. If an example graph network called for a graph structure similar to the Cora dataset, which has $\sim$2 edges per node and $\sim$2 nodes per feature, it would be very important to include the running time into graph network selection. With the Cora dataset, the GCN and Geom-GCN produce similar accuracy levels ($\sim 85\%$), but Geom-GCN takes $10\times$ longer to train.

5. Implementation

5.1. Implementation Motivation

Graph convolutions are interesting due to the prevalence of graph data structures across different industries. The original GCN paper presented by Kipf and Welling has been cited thousands of times by other researchers in the years following their report. There have been several novel approaches to simplifying and powering graph convolutional networks for classification or recommendations in recent machine learning conferences, and SGC has received a large amount of attention due to the performance claims and efficiency gains over traditional GCN.

Because SGC code[3] is available on GitHub by one of the authors, this implementation aimed to rerun the citation network experiments from the paper for both SGC and GCN using a PyTorch implementation and extend node-classification tests across several different page-page datasets. The main goal behind rerunning these experiments was to prove if the claims in the SGC paper extend beyond standard citation networks to other types of graph datasets, specifically page-page datasets.

Table 2. Test configuration for SGC and GCN.

ITEM	SGC	GCN
NUMBER OF EPOCHS	100	100
OPTIMIZER	ADAM	ADAM
LOSS FUNCTION	CROSSENTROPY	NLL
INITIAL LEARNING RATE	0.2	0.01
WEIGHT DECAY	5E-6	5E-4
HIDDEN UNITS	0	50
DROPOUT RATE	0.0	0.5
DEGREE OF APPROX.	2	0

5.2. Implementation Setup and Plans

The initial plan involved pulling the SGC code available on GitHub and locally rerunning the citation network tests on SGC and GCN using the three common citation networks: Cora, Citeseer, and Pubmed. This plan proved unsuccessful due to Python package dependency issues. SGC code relied on specific, older versions of PyTorch, NetworkX, and SciPy that gave incompatibility issues with pip and other Python package managers attempted. Upon further review of the SGC code, there are several discrepancies in the GCN implementation compared to the the PyTorch implementation[4] referenced in the SGC paper. The `forward` function in the Graph Convolution layer is missing a return parameter with the correct output, representing a major flaw in the implementation. The Graph Convolution layer is the fundamental layer of the GCN model and may indicate errors in the experiment results from Table 1 (see Table 2 from the SGC paper).

An updated plan included a SGC and GCN re-implementation and reusing the following from the SGC code: citation network datasets, functions to load the datasets into PyTorch tensors, extract the features, extract the labels, determine the train and test indexes, and normalize the sparse adjacency matrices. The following were re-implemented using SGC code as a reference: the SGC model, the Graph Convolutional Layer (GCL), the GCN model; the train and test functions for SGC and GCN; the model configuration, setup, and analysis functions. Instead of dealing with a Python package manager and potential issues of non-repeatable tests due to a custom local environment, the experiments were run in a Google Colab Jupyter Notebook on a T4 hosted runtime with each test repeated and averaged over 20 runs. The SGC and GCN test configurations outlined in Table 2 match descriptions from each

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

[4] https://github.com/tkipf/pygcn

original paper, respectively. Tests on the Citeseer dataset produced a large number of missing and zero based node errors, so the citation network dataset tests were downselected to only run SGC and GCN train and test functions on the Cora and Pubmed citation networks. The main implementation motivation for these citation network tests was to prove the validity of the SGC paper claims with an expectation of lower average performance than reported, so training and testing on two of the three citation networks met this goal.

Given the SGC authors split the Cora and Pubmed datasets into train and test groups to match the PyTorch GCN implementation and testing, this implementation reused the same train, validate, and test indexes for initial tests. The nodes were sorted and split into train, validate, and test ranges by selecting a portion of graph nodes using a (x, y) 2D index. The test index is selected first on a sorted dictionary of the nodes, then a validation set for use during training, and finally, the remainder are used for training. The Cora dataset nodes are split into groups of 1,000/500/140 for test, validation, and training; the Pubmed dataset nodes are split similarly with groups of 1,000/500/60. A short summary of the initial tests: train and test SGC and GCN for Cora and Pubmed citation network datasets to compare performance using training time and measured test accuracy against the published metrics from the SGC paper.

Training and testing SGC and GCN was extended to the Wikipedia page-page dataset Chameleon (2021) and the the WebKB[5] page-page datasets consisting of Cornell, Texas, and Wisconsin computer science webpage graphs. This allowed confirmation of the GCN results published in both GCNII and Geom-GCN, generation of new results comparing SGC against GCNII and Geom-GCN implementations, and generation of timing metrics to compare GCN and SGC on these new datasets. The Geom-GCN authors published the set of 10 randomized training/validation/testing splits across the page-page datasets[6], so these splits were incorporated as part of the function to load the new datasets. The extension tests followed a similar pattern as with the citation network datasets: use the training split to train the model, use the validation split for performance validation during training, and use the testing split for post-training analysis of the model. Each test configuration was repeated and averaged over 20 runs. A short summary of the extended tests: train and test SGC and GCN for Chameleon, Cornell, Texas, and Wisconsin datasets and compare performance using training time and measured test accuracy against the reported test accuracy metrics in literature.

[5] https://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

[6] https://github.com/bingzhewei/geom-gcn

Table 3. Test results for SGC and GCN, averaged over 20 runs with 100 epochs each on a T4. Training time was measured in seconds.

MODEL	DATASET	TRAIN TIME	ACCURACY
GCN	CORA	0.58	81.2 ± 0.4
GCN	PUBMED	0.71	79.3 ± 0.4
SGC	CORA	0.16	80.6 ± 0.1
SGC	PUBMED	0.14	77.9 ± 0.1

5.3. Implementation Details

Three code bases were used for reference during implementation: SGC (https://github.com/Tiiiger/SGC), PyGCN (https://github.com/tkipf/pygcn), and Geom-GCN (https://github.com/bingzhewei/geom-gcn). As mentioned in Section 5.2, the SGC, GCL, and GCN were re-implemented. New functions were written for configuration, train, test, and runtime analysis for citation network datasets and the extended page-page dataset tests. The Cora and Pubmed citation network datasets were reused from SGC, as well as functions to normalize the graphs, parse index files, precompute features (adjacency matrices), and load citation networks. PyGCN was used as a reference for re-implementing GCL and the GCN model. The page-page network datasets and a function to load and normalize these datasets were reused from GeomGCN.

5.4. Results and Interpretation

See Table 3 for preliminary results of SGC and GCN on the Cora and Pubmed datasets. Test accuracy across the model and dataset conditions does not deviate in a statistically significant way from the SGC results shown in Table 1. However, the Pubmed dataset relative training time shown in Figure 2 differs significantly in this rerun experiment. The claimed performance gain of relative training time of SGC over GCN for the Pubmed dataset was 28x; the measured performance gain of relative training time of SGC over GCN of the Pubmed dataset in this rerun is 5.1x. The initial interpretation of this result may be due to the incorrectly implemented Graph Convolutional layer (GCL) in the SGC authors' code causing slower convergence during training with the layer output mismatch while forward feeding the nodes. However, this hypothesis has not been tested due to the issues surrounding the attempts to run the SGC authors' code.

The test configuration for SGC and GCN remained the same, detailed in Table 2. Due to the new training splits, each model was run 20 times per split on each of the 4 new datasets. The measured time and accuracy was averaged for each dataset across the 10 splits, with the results shown in Table 4. There is a measurable difference in the GCN

Table 4. Test results for SGC and GCN on new datasets, averaged over 10 splits with 20 runs of 100 epochs each on a T4. Training time was measured in seconds. The T column with a "T" represents test results measured and collected with this implementation, "*" represents the reported value from Table 5 of the GCNII paper.

T	MODEL	DATASET	TRAIN TIME	ACCURACY
T	SGC	CHAM.	0.21	61.6 ± 0.0
T	GCN	CHAM.	0.58	55.9 ± 1.3
*	GCN	CHAM.	–	28.18
*	GCNII	CHAM.	–	60.61
T	SGC	CORN.	0.16	56.5 ± 0.0
T	GCN	CORN.	0.55	59.7 ± 2.0
*	GCN	CORN.	–	52.70
*	GCNII	CORN.	–	74.86
T	SGC	TEXA.	0.16	54.9 ± 0.0
T	GCN	TEXA.	0.53	52.7 ± 2.3
*	GCN	TEXA.	–	52.16
*	GCNII	TEXA.	–	69.46
T	SGC	WISC.	0.16	47.1 ± 0.0
T	GCN	WISC.	0.54	48.5 ± 2.4
*	GCN	WISC.	–	45.88
*	GCNII	WISC.	–	74.12

performance reported in the GCNII paper across these four datasets, especially regarding the Chameleon dataset. Testing using the re-implemented version of GCN demonstrates an accuracy of $55.9 \pm 1.3\%$, which is nearly double the 28.18% node classification accuracy of GCN reported in Table 5 in the GCNII paper.

With the inclusion of four new datasets, it is important to note the relative accuracy between SGC and GCN. The initial and extended measurements confirm the validity of the SGC authors' claims of competitive performance to state of the art Graph Convolutional Networks and decreased training time. However, the statement in the SGC paper indicating up to two orders of magnitude faster training time compared to GCN is not reflected by any measurements on any of the citation network datasets or webpage datasets. The re-implemented SGC maintains $\sim 3\times$ training speed increase over GCN across all tested datasets. SGC also requires pre-computing the adjacency matrices required during training; the pre-computation times are included in the measured training time of SGC.

My personal expectation prior to testing was for SGC to under-perform compared to GCN due to the removal of hidden layers from the model. However, the tests consistently showed SGC performance nearly on par with GCN and a decrease in required training time across all datasets and all splits. GCN accuracy was higher on a majority of the datasets, but SGC had higher accuracy on Chameleon and Texas page-page datasets. In the situation of implementing a graph structure node-classification model in a production environment, I would attempt to use and modify a simplified GCN model like SGC and only turn to deeper models if absolutely necessary.

6. Conclusion and Discussion

Deep variants of GCN, including GCNII and Geom-GCN, can achieve higher node-classification accuracy on citation network datasets and webpage network datasets. These deep variants may need unique modifications to achieve scenario specific accuracy improvements, see GCNII* compared to GCNII in Section 6.2 of the GCNII paper. These additional variations can increase training time due to hidden layer complexity. Simplified variants of GCN, including this re-implementation of SGC, can achieve similar node-classification accuracy across datasets in a fraction of the time compared to the deep variants and the base GCN. SGC is an impressive graph convolutional network and should be considered a strong candidate in any application where training time is an important factor regarding model performance.

The performance metrics of SGC hold across citation networks and page-page networks; this may indicate the metrics hold across other types of graph datasets but will require further testing to validate. As mentioned in Section 2.1, applications of graph networks include citation networks, social networks, natural language processing, and computer vision. These other types of applications will require further investigation and testing.

References

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013. URL https://api.semanticscholar.org/CorpusID:17682909.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Li, H., Yang, Y., Chen, D., and Lin, Z. Optimization algorithm inspired deep neural network structure design, 2018.

Ming Chen, Z. W., Zengfeng Huang, B. D., and Li, Y. Simple and deep graph convolutional networks. 2020.

Nadler, B., Srebro, N., and Zhou, X. Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta,

A. (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/68ce199ec2c5517597ce0a4d89620f55-Paper.pdf.

Newman, M. E. J. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, Oct 2002. doi: 10.1103/PhysRevLett.89.208701. URL https://link.aps.org/doi/10.1103/PhysRevLett.89.208701.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):1–22, May 2021. ISSN 2051-1310. doi: 10.1093/comnet/cnab014.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, X., Ye, Y., and Gupta, A. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6857–6866, 2018.

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

Yao, L., Mao, C., and Luo, Y. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7370–7377, 2019.

Zhu, X. *Semi-supervised learning with Graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.