Extention of Continuos Graph Neural Networks

Anonymous Authors[1]

1. Abstract

The implementation of Continuously Evolving Graph Neural Networks (CGNNs) serves as a strategic response to the over-smoothing predicament prevalent in conventional graph neural networks. By embracing a continuous dynamics modeling approach, the primary goal is to enhance the capture of long-term dependencies within graph-structured data. The landscape of graph-structured data modeling is enriched by key approaches such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), while the adaptation of Neural Ordinary Differential Equation (Neural ODE) by Chen et al. (2018) to graph-structured data marks a novel extension in the evolution of graph neural networks.

The practical implementation involves leveraging PyTorch and the provided codebase to replicate the paper's results, evaluate the performance of CGNN and WCGNN under various modifications, and experiment with hyperparameters. Notably, the results unveil CGNN's sensitivity to changes, indicated by a 79.6percent decrease, while WCGNN demonstrates greater resilience with an 81.5percent maintained performance. The implementation introduces key modifications, including Xavier uniform weight initialization, L1 and L2 regularization, and architectural changes in WCGNN to improve graph data capture. While providing valuable insights, the study underscores the importance of careful model modifications and opens avenues for further exploration and refinement.

2. Introduction

The motivation for this project stems from the pursuit of a more adaptive and expressive framework that can seamlessly accommodate the continuous evolution of graphs over time. Inspired by the research paper on continuous graph networks.The papers reviewed in the course of this project shed light on the shortcomings of existing graph convolutional models and propose solutions that leverage ordinary differential equations (ODEs) to capture the temporal dynamics of graph-structured data and long range dependencies. Our project overview encompasses the extension of the proposed continuous graph network model. This involves adapting the architecture to handle dynamic graphs, experimenting with different activation functions, exploring the impact of additional regularization techniques, and investigating the efficacy of incorporating deeper GCN layers.

3. Review of "Continuous Graph Neural Networks"

3.1. Storyline

The paper "Continuous Graph Neural Networks,"[2] unfolds by first emphasizing the critical role that graph structures play in various data-driven applications. It begins by highlighting the significance of modeling continuous dynamics in graph data, underscoring the limitations of traditional discrete graph neural networks (GNNs). The paper then progresses logically to elucidate the shortcomings of GNNs, particularly their inability to capture long-term dependencies and vulnerability to the over-smoothing problem, which can degrade performance.

High-level motivation/problem The larger goal and motivation for the research presented in the paper [2] is to advance the field of graph-based machine learning and data analysis. The paper addresses the fundamental challenge of learning informative representations from graph-structured data. Graphs are a versatile representation for a wide range of real-world phenomena, from social networks to biological interactions, and unlocking their potential is crucial for various applications. By introducing Continuous Graph Neural Networks (CGNNs), the paper aims to provide a novel framework for modeling the continuous dynamics of node representations in graph-structured data. The larger vision is to enable more effective and robust data-driven decision-making in fields where graphs are ubiquitous, such as social network analysis, recommendation systems, biology, and more. CGNNs offer a way to capture long-term dependencies and global relationships within graphs, which can improve the accuracy of various machine learning tasks in these domains.

Prior work on this problem Prior research has made significant efforts to address the challenge of effectively modeling graph-structured data and capturing long-term dependencies within such data. Some of the key approaches and methods used in prior work include:

1. Graph Convolutional Networks (GCNs): GCNs, proposed by Kipf and Welling in 2017, introduced a framework for learning node representations by aggregating information from neighboring nodes in a graph. These models have been widely adopted and serve as the basis for various graph-based tasks.

2. Graph Attention Networks (GATs): Graph Attention Networks, introduced by Velicković et al. in 2018, enhanced the idea of GCNs by incorporating attention mechanisms. This allowed nodes to attend to different neighbors with varying weights, improving their ability to capture complex relationships.

3. Neural ODE Neural ODE (Chen et al., 2018) : It is an approach for modelling a continuous dynamics on hidden representation, where the dynamic is characterised through an ODE parameterised by a neural network. However, these methods can only deal with unstructured data, where different inputs are independent. Our approach extends this in a novel way to graph structured data.

Research gap Prior research in the field has explored various techniques to address the problem of learning informative representations from graph-structured data, which forms the foundation of this paper's investigation. Notable approaches and methods include:

1. Graph Neural Networks (GNNs): Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) are two representative GNNs that have gained significant attention. These methods aim to model the discrete dynamics of node representations through the aggregation of information from neighboring nodes in multiple layers. However, they often suffer from over-smoothing issues and are limited in capturing long-range dependencies.

2. Graph Neural ODEs (GODEs): Recent research introduced the concept of GODEs, which extend the idea of Neural ODEs to graph data. GODEs aim to model continuous dynamics on graph-structured data, similarly to CGNNs, but they rely on existing GNNs as building blocks to parameterize their ODEs.

These prior approaches offer valuable insights and techniques for addressing the challenges of modeling graph-structured data. However, they often face limitations related to over-smoothing, difficulties in capturing long-term dependencies, and scalability issues.

Contributions The paper "Continuous Graph Neural Networks" makes noteworthy contributions to the field of graph representation learning through the introduction of Continuous Graph Neural Networks (CGNNs). These novel models extend the concept of Neural Ordinary Differential Equations (Neural ODEs) to graph-structured data, providing a continuous framework for evolving node representations. CGNNs excel in capturing long-term dependencies and dynamic information flow within graphs, effectively addressing the over-smoothing challenge often encountered in discrete graph neural networks. The models exhibit flexibility by offering two variants—one allowing independent changes in different feature channels and another modeling interactions between channels. The paper contributes theoretical insights, justifying the proposed Ordinary Differential Equations (ODEs) and offering analyses of the impact of eigenvalues on learned representations, thereby enhancing the understanding of the model's properties. Additionally, CGNNs demonstrate superior memory efficiency, maintaining constant memory usage over time, making them particularly suitable for modeling long-term node dependencies in large graphs. These contributions collectively advance the state-of-the-art in graph representation learning and pave the way for more effective and versatile applications in various domains.

3.2. Proposed solution

The paper introduces Continuous Graph Neural Networks (CGNNs) as a novel solution to address the challenge of capturing long-term dependencies and continuous dynamics within graph-structured data. The key idea behind CGNNs is to extend the concept of Neural Ordinary Differential Equations (Neural ODEs) to the realm of graph neural networks. This is achieved by defining continuous dynamics on node representations using Ordinary Differential Equations. The central ODE used in CGNNs is formulated as:

$$dH(t)dt = (A - I)H(t) + E$$

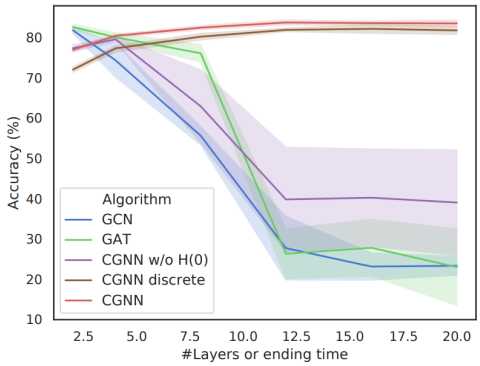
Here, (H(t)) represents the node representations at time (t), and (E) is the initial value computed by the encoder. The matrix (A) encodes the graph structure, and (I) is the identity matrix. This ODE allows for continuous propagation of information among nodes, akin to an epidemic model. The continuous nature of CGNNs helps mitigate the over-smoothing problem and enables the modeling of global dependencies by considering the entire range of time (t). This is in contrast to traditional discrete graph neural networks, which are sensitive to the number of layers and struggle to capture long-term dependencies. CGNNs offer flexibility by allowing different feature channels to evolve independently or interact with each other, depending on the variant used.

The introduction of CGNNs bridges the gap between continuous dynamics and graph representation learning, providing

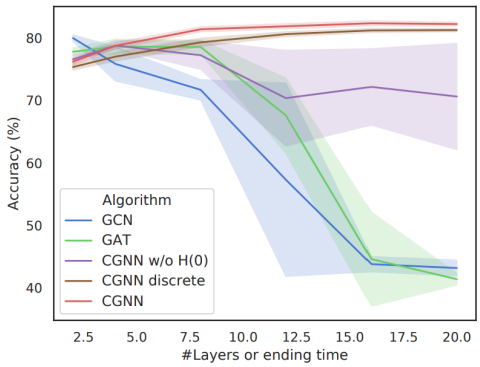
a promising solution for various data-driven applications where preserving long-term information flow within graphs is crucial. It offers a fresh perspective by utilizing ODEs to describe the dynamic behavior of nodes in a graph and opens up new possibilities for modeling complex systems in continuous time.

It is worth noting that CGNNs also exhibit robustness to over-smoothing and perform effectively even with a large number of layers, overcoming the overfitting issues that plague traditional GNNs. This demonstrates the practical advantages of CGNNs in capturing long-term dependencies while maintaining generalization capabilities.

3.3. Claims-Evidence



(a) Cora

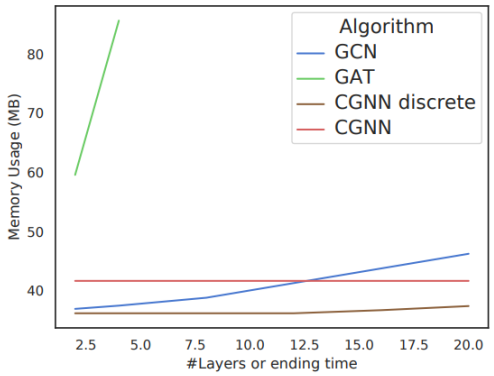


(b) Pubmed

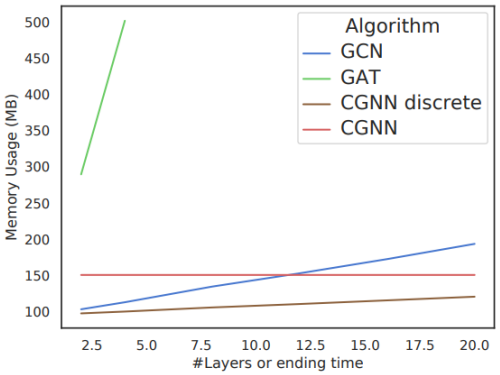
Figure 1. : Performance w.r.t. layers or ending time. Note that the red line also has error bars, but they are very small.

Claim 1: CGNNs effectively model long-term dependencies and overcome over-smoothing.

Evidence 1: Figure 1 which has been taken from [2] demonstrates the performance of CGNN and its variants concerning the number of layers or ending time. While traditional methods like GCN and GAT show diminishing performance with increased layers due to over-smoothing, CGNN remains stable and even benefits from longer time steps, indicating its robustness to over-smoothing and its ability to capture long-term dependencies.



(a) Cora



(b) Pubmed

Figure 2. Memory usage w.r.t. layers or ending time.

Claim 2: CGNNs exhibit constant memory usage regardless of the number of layers.

Evidence 2: Figure 2 which has been taken from [2] provides evidence of the memory efficiency of CGNN in comparison to traditional methods like GCN and GAT. While the memory usage of GCN and GAT linearly increases with the number of layers, CGNN maintains a constant and lower memory cost. This feature makes CGNN suitable for modeling long-term node dependencies in large graphs without excessive memory requirements.

Claim 3: The experimental results consistently demonstrate that CGNN in its continuous form, without discretization, consistently achieves the highest accuracy across all datasets when compared to other variants .

Model	Cora	Citeseer	Pubmed	NELL**
GAT-GODE*	83.3 ± 0.3	72.1 ± 0.6	79.1 ± 0.5	-
GCN-GODE*	81.8 ± 0.3	72.4 ± 0.8	80.1 ± 0.3	-
GCN	81.8 ± 0.8	70.8 ± 0.8	80.0 ± 0.5	57.4 ± 0.7
GAT***	82.6 ± 0.7	71.5 ± 0.8	77.8 ± 0.6	-
CGNN discrete	81.8 ± 0.6	70.0 ± 0.5	81.0 ± 0.4	50.9 ± 3.9
CGNN	$\mathbf{84.2 \pm 1.0}$	72.6 ± 0.6	$\mathbf{82.5 \pm 0.4}$	65.4 ± 1.0
CGNN with weight	83.9 ± 0.7	$\mathbf{72.9 \pm 0.6}$	82.1 ± 0.5	$\mathbf{65.6 \pm 0.9}$

Figure 3. Node classification results on citation networks. The values are taken from the original paper.

Evidence 3: By examining the results presented in Figure 3 taken from [2], it is evident that the CGNN variant with continuous dynamics (referred to as "CGNN" in the chart) consistently outperforms other variants, such as "GCN", "GAT" which uses discrete propagation steps. This consistent trend suggests that the continuous nature of CGNN plays a crucial role in its effectiveness.

3.4. Critique and Discussion

I think the paper is easy to understand and has provided all their claims supported by evidences. They have showed how with many layers their accuracy did not go down and was able to capture graph data. Also their constant memory usage with time for datasets have shown what they claimed. However they have said that the proposed model worked even better for complex graphs but they have not evaluated on it.Despite the innovative approaches presented in the paper, notable gaps exist in addressing certain aspects, One glaring omission is the absence of a comprehensive discussion on potential limitations inherent in their proposed methodologies.

4. Review of "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning"

4.1. Storyline

In this paper, the authors delve into the realm of semi-supervised classification on graphs, focusing on the Graph Convolutional Network (GCN). They commence by introducing the problem's significance and noting the prominence of GCNs while highlighting the need for a deeper understanding and optimization. The paper proceeds by elucidating the inner workings of GCNs, emphasizing the role of Laplacian smoothing in graph-based classification. It then dissects the limitations of GCNs, particularly the pitfalls of over-smoothing, which homogenizes vertex features. To mitigate these challenges, the authors propose innovative solutions: Co-Training via Random Walk Models, GCN Self-Training, and the amalgamation of both strategies. They offer theoretical insights on the number of labeled samples necessary for effective GCN training. Extensive experiments validate these approaches across diverse datasets, demonstrating their substantial superiority over standard GCNs and other state-of-the-art methods, especially in scenarios with limited labeled data. In conclusion, the paper contributes to unraveling the mysteries of GCNs and advances their applicability in semi-supervised graph-based classification while hinting at future research avenues.

High-level motivation/problem The high-level motivation behind this research is to enhance the understanding and application of Graph Convolutional Networks (GCNs) in the domain of semi-supervised classification on graphs. Beyond the scope of this paper, the larger goal is to harness the power of GCNs for various real-world applications involving structured data, such as social networks, recommendation systems, and biological networks. By addressing the challenges and limitations of GCNs, this research contributes to the broader vision of developing more robust and efficient graph-based machine learning methods. The ultimate aim is to create intelligent systems capable of making accurate predictions and classifications in scenarios with limited labeled data, thereby advancing the capabilities of AI in analyzing complex relational data and making informed decisions.

Prior work on this problem The paper briefly discusses prior work on the problem of semi-supervised classification using Graph Convolutional Networks (GCNs). It mentions that the introduction of GCNs, as presented in the paper by Kipf and Welling in 2017, significantly improved the performance of semi-supervised learning on graph-structured data. GCNs leverage graph convolution and spectral filters for feature propagation. However, it's highlighted that GCNs have limitations when dealing with small amounts of labeled data and require a validation set for optimal performance, which raises concerns about their practicality. The prior work, as outlined in the paper, laid the foundation for the research in this paper by showcasing the potential of GCNs while revealing their constraints, which this research aims to address.

Research gap The text introduces graph-based semi-supervised learning and Graph Convolutional Networks (GCNs). Research gaps can be identified in several areas. First, there's a need for more scalable spectral GCNNs to handle large graphs efficiently. Second, optimizing the integration of graph structures and feature vectors for enhanced accuracy is an ongoing challenge. Third, extending and adapting GCNs to different graph types and applications presents opportunities for advancement. Finally, there's a need for further exploration of practical applications of graph-based learning in real-world scenarios. Addressing

these gaps holds the potential to advance the field of graph-based machine learning.

Contributions The paper's main contributions encompass a comprehensive understanding of Graph Convolutional Networks (GCNs), unveiling their mechanisms and limitations, and presenting novel solutions to address these limitations. These novel solutions involve co-training with a random walk model, self-training GCNs, and a combination of both (Union and Intersection) to improve classification performance, particularly in scenarios with limited labeled data. Empirical validation is provided through extensive experiments on real-world datasets, highlighting the effectiveness of the proposed methods. The paper also offers theoretical insights into the conditions under which over-smoothing occurs in GCNs, shedding light on GCN behavior. Furthermore, it suggests future research directions for enhancing GCNs and expanding their applications in graph-based domains.

4.2. Proposed solution

The paper proposes innovative solutions to address the limitations of Graph Convolutional Networks (GCNs) in semi-supervised classification tasks. To tackle this challenge, the paper presents two novel strategies: co-training with a partially absorbing random walk model (ParWalks) and self-training for GCNs.

In co-training, the ParWalks algorithm is employed to exploit the global graph structure by identifying the most confident vertices closest to labeled data points. These confident instances are then added to the labeled set, enhancing the quality of training data and enabling GCNs to capture global graph information more effectively.On the other hand, the self-training approach utilizes the predictions generated by the GCN itself. The algorithm identifies the most confident predictions based on softmax scores and appends them to the labeled set. The key idea here is to create a self-reinforcing mechanism that incrementally improves the quality of the labeled set, leading to a more robust and accurate classifier.

Additionally, the paper introduces two strategies, namely "Union" and "Intersection," which combine co-training and self-training. The "Union" method aims to add more diverse labels to the training set, while the "Intersection" method selects the most confident predictions shared by both random walk and GCN-based approaches.

4.3. Claims-Evidence

Claim 1: The proposed co-training methods, specifically "Union" and "Intersection," significantly outperform traditional GCNs (GCN-V and GCN+V) in semi-supervised classification tasks.

Table 3: Classification Accuracy On Cora

	Cora					
Label Rate	0.5%	1%	2%	3%	4%	5%
LP	56.4	62.3	65.4	67.5	69.0	70.2
Cheby	38.0	52.0	62.4	70.8	74.1	77.6
GCN-V	42.6	56.9	67.8	74.9	77.6	79.3
GCN+V	50.9	62.3	72.2	76.5	78.4	79.7
Co-training	56.6	66.4	73.5	75.9	78.9	80.8
Self-training	53.7	66.1	73.8	77.2	79.4	80.0
Union	**58.5**	**69.9**	**75.9**	**78.5**	**80.4**	**81.7**
Intersection	49.7	65.0	72.9	77.1	79.4	80.2

Figure 4. : Classification Accuracy On Cora

Evidence 1: In Figure 4, Experimental results on the Cora dataset demonstrate that the "Union" method achieves an accuracy of 75.9 Percent with a labeling rate of 2 Percent, while GCN-V and GCN+V achieve 67.8Percent and 72.2 Percent, respectively. On CiteSeer, "Intersection" achieves an accuracy of 70.1 Percent with a 3 Percent labeling rate, surpassing GCN-V (66.9 Percent) and GCN+V (67.5Percent).

Table 5: Classification Accuracy On PubMed

	PubMed			
Label Rate	0.03%	0.05%	0.1%	0.3%
LP	61.4	66.4	65.4	66.8
Cheby	40.4	47.3	51.2	72.8
GCN-V	46.4	49.7	56.3	76.6
GCN+V	60.5	57.5	65.9	77.8
Co-training	**62.2**	**68.3**	**72.7**	78.2
Self-training	51.9	58.7	66.8	77.0
Union	58.4	64.0	70.7	**79.2**
Intersection	52.0	59.3	69.4	77.6

Figure 5. Classification Accuracy On PubMed

Claim 2: The proposed co-training strategies are particularly effective when labeled data is scarce, offering a considerable advantage over traditional GCNs.

Evidence 2: In Figure 5, on the PubMed dataset with a minimal labeling rate of 0.05 percent, the "Union" approach improves accuracy by 37 percent compared to GCN-V and 18 percent compared to GCN+V.

Claim 3: The co-training methods, especially "Union" and "Intersection," exhibit substantial superiority over various state-of-the-art baselines.

Table 6: Accuracy under 20 Labels per Class

Method	CiteSeer	Cora	Pubmed
ManiReg	60.1	59.5	70.7
SemiEmb	59.6	59.0	71.7
LP	45.3	68.0	63.0
DeepWalk	43.2	67.2	65.3
ICA	69.1	75.1	73.9
Planetoid	64.7	75.7	77.2
GCN-V	68.1	80.0	78.2
GCN+V	68.9	80.3	**79.1**
Co-training	64.0	79.6	77.1
Self-training	67.8	80.2	76.9
Union	65.7	**80.5**	78.3
Intersection	**69.9**	79.8	77.0

Figure 6. Accuracy under 20 Labels per Class

Evidence 3: In Figure 6, When compared to other methods such as LP (Label Propagation), Cheby (GCN with Chebyshev filter), and several other state-of-the-art techniques, our proposed methods consistently achieve higher accuracy rates, demonstrating their effectiveness in semi-supervised classification tasks.

4.4. Critique and Discussion

The paper maintains a high level of clarity and structure throughout, effectively communicating its ideas in a well-organized manner. It outlines the limitations of Graph Convolutional Networks (GCNs) and how the proposed co-training strategies address these issues. The narrative flows logically, making it easy to follow the paper's progression.

The paper's contributions are substantial. It provides a comprehensive analysis of GCNs, emphasizing their limitations when dealing with limited labeled data. The proposed co-training methods, namely "Union" and "Intersection," stand out as innovative approaches that significantly improve classification accuracy. In many cases, these methods outperform various state-of-the-art techniques, making them valuable additions to the machine learning toolbox. It also provides evidence along with claims to support it.

The paper makes a fundamental assumption that expanding the labeled dataset through co-training methods is an effective approach to improving classification accuracy in semi-supervised learning. Although this assumption is supported by strong empirical evidence, it would be valuable to explore scenarios or conditions in which these methods might not be as effective or even potentially counterproductive. Investigating the limitations or edge cases where co-training strategies may not provide significant benefits would contribute to a more comprehensive understanding of their applicability. This analysis would help researchers and practitioners better assess when and how to leverage co-training methods in real-world applications.

5. Review of "Graph Convolutional Networks for Text Classification"

5.1. Storyline

The paper introduces Text Graph Convolutional Networks (Text GCN), a novel approach for text classification. It begins by addressing the challenges of text classification on large datasets and provides background information on traditional and deep learning-based methods. The core concept of Graph Convolutional Networks (GCN) is explained, with a focus on its application to text data. The authors describe the construction of a heterogeneous text graph, detailing the methods for establishing relationships between words and documents. Text GCN is then presented, a two-layer model that learns representations for words and documents from the text graph. Extensive experiments compare Text GCN to various baseline models, revealing its superior performance. The paper concludes by emphasizing the effectiveness of Text GCN and suggesting potential future research directions in text classification and representation learning.

High-level motivation/problem The high-level motivation for this research is to address the challenges in text classification, especially on large datasets. The paper seeks to enable accurate and efficient text classification, even when labeled data is limited. The larger goal is to advance the field of natural language processing (NLP) by developing a model that can effectively learn predictive word and document embeddings. Text classification is a fundamental task in NLP with numerous real-world applications, and the paper aims to improve the state-of-the-art in this area. By proposing Text Graph Convolutional Networks (Text GCN), the authors strive to harness the power of graph-based representation learning to enhance the performance of text classification models, ultimately contributing to advancements in information retrieval, recommendation systems, and other NLP applications.

Prior work on this problem The paper presents a comprehensive examination of prior research related to text classification, encompassing a spectrum of methodologies and techniques. It surveys traditional approaches like TF-IDF combined with logistic regression and deep learning models such as Convolutional Neural Networks (CNN) in both randomly initialized and pre-trained word embedding variations. Long Short-Term Memory (LSTM) models for text classification, with and without pre-trained word embeddings, are also explored. The study further delves into para-

graph vector models (PV-DBOW and PV-DM) and presents Predictive Text Embedding (PTE), which learns word embeddings based on heterogeneous text networks. Additionally, the paper discusses fastText, Simple Word Embedding Models (SWEM), Label-Embedding Attentive Models (LEAM), and Graph Convolutional Neural Network (Graph-CNN) models. This extensive review of prior work provides the context and foundation for assessing the proposed Text Graph Convolutional Network (Text GCN) against established methods, ultimately showcasing its advancements in text classification.

Research gap The research paper identifies a critical research gap in the field of text classification related to the limited incorporation of graph-based models. While prior work has predominantly focused on utilizing traditional deep learning and embedding methods, such as CNN, LSTM, and paragraph vector models, the paper introduces Text Graph Convolutional Networks (Text GCN) as a novel approach. This gap is rooted in the failure of existing methods to adequately capture both document-word relations and global word-word relations in text data, a deficiency that Text GCN aims to address by leveraging graph-based convolution techniques. The research gap pertains to the need for methods that can effectively exploit structural information in large-scale text corpora, combining both local and global context, to enhance text classification performance.

Contributions This paper presents Text Graph Convolutional Networks (Text GCN), a novel framework for text classification that harnesses the power of graph-based convolution techniques. Text GCN effectively addresses the challenge of capturing both local document-word relationships and global word-word associations in large text corpora. The key contributions include remarkable improvements in text classification accuracy, particularly on datasets featuring longer text documents, a novel integration of graph-based models into text classification, offering a fresh perspective to enhance classification performance, and an extensive experimental analysis that demonstrates the effectiveness of Text GCN when compared to various state-of-the-art text classification and embedding methods on multiple benchmark datasets.

5.2. Proposed solution

The proposed solution in this paper addresses the research gap in text classification by introducing a novel approach using Graph Convolutional Networks (GCN) specifically designed for text data. The key idea is to model the entire corpus as a heterogeneous graph, where word nodes and document nodes represent the nodes in the graph. Unlike previous methods that primarily focus on local consecutive word sequences or rely on predefined document citation relations, the proposed Text Graph Convolutional Network (Text GCN) takes advantage of both word co-occurrence within documents and global word co-occurrence in the entire corpus. This is achieved by constructing document-word and word-word edges based on term frequency-inverse document frequency (TF-IDF) for document-word edges and point-wise mutual information (PMI) for word-word edges. The text graph is then fed into a two-layer GCN, allowing information exchange between pairs of documents and capturing high-order neighborhood information. The model is trained using a softmax classifier for document classification, with the objective of minimizing cross-entropy error over labeled documents. This innovative approach not only outperforms state-of-the-art methods for text classification but also automatically learns interpretable word and document embeddings, showcasing the effectiveness of the proposed Text GCN in capturing both local and global semantic information in textual data.

5.3. Claims-Evidence

Model	20NG	R8	R52	Ohsumed	MR
TF-IDF + LR	0.8319 ± 0.0000	0.9374 ± 0.0000	0.8695 ± 0.0000	0.5466 ± 0.0000	0.7459 ± 0.0000
CNN-rand	0.7693 ± 0.0061	0.9402 ± 0.0057	0.8537 ± 0.0047	0.4387 ± 0.0100	0.7498 ± 0.0070
CNN-non-static	0.8215 ± 0.0052	0.9571 ± 0.0052	0.8759 ± 0.0048	0.5844 ± 0.0106	**0.7775 ± 0.0072**
LSTM	0.6571 ± 0.0152	0.9368 ± 0.0082	0.8554 ± 0.0113	0.4113 ± 0.0117	0.7506 ± 0.0044
LSTM (pretrain)	0.7543 ± 0.0172	0.9609 ± 0.0019	0.9048 ± 0.0086	0.5110 ± 0.0150	0.7733 ± 0.0089
Bi-LSTM	0.7318 ± 0.0185	0.9631 ± 0.0033	0.9054 ± 0.0091	0.4927 ± 0.0107	0.7768 ± 0.0086
PV-DBOW	0.7436 ± 0.0018	0.8587 ± 0.0010	0.7829 ± 0.0011	0.4665 ± 0.0019	0.6109 ± 0.0010
PV-DM	0.5114 ± 0.0022	0.5207 ± 0.0004	0.4492 ± 0.0005	0.2950 ± 0.0007	0.5947 ± 0.0038
PTE	0.7674 ± 0.0029	0.9669 ± 0.0013	0.9071 ± 0.0014	0.5358 ± 0.0029	0.7023 ± 0.0036
fastText	0.7938 ± 0.0030	0.9613 ± 0.0021	0.9281 ± 0.0009	0.5770 ± 0.0049	0.7514 ± 0.0020
fastText (bigrams)	0.7967 ± 0.0029	0.9474 ± 0.0011	0.9099 ± 0.0005	0.5569 ± 0.0039	0.7624 ± 0.0012
SWEM	0.8516 ± 0.0029	0.9532 ± 0.0026	0.9294 ± 0.0024	0.6312 ± 0.0055	0.7665 ± 0.0063
LEAM	0.8191 ± 0.0024	0.9331 ± 0.0024	0.9184 ± 0.0023	0.5858 ± 0.0079	0.7695 ± 0.0045
Graph-CNN-C	0.8142 ± 0.0032	0.9699 ± 0.0012	0.9275 ± 0.0022	0.6386 ± 0.0053	0.7722 ± 0.0027
Graph-CNN-S	–	0.9680 ± 0.0020	0.9274 ± 0.0024	0.6282 ± 0.0037	0.7699 ± 0.0014
Graph-CNN-F	–	0.9689 ± 0.0006	0.9320 ± 0.0004	0.6304 ± 0.0077	0.7674 ± 0.0021
Text GCN	**0.8634 ± 0.0009**	**0.9707 ± 0.0010**	**0.9356 ± 0.0018**	**0.6836 ± 0.0056**	0.7674 ± 0.0020

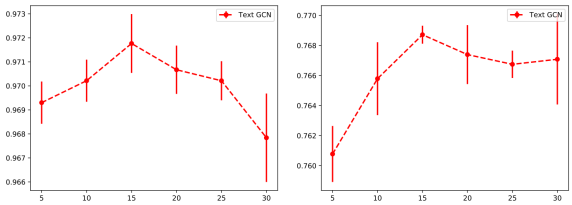
Figure 7. : Test Accuracy on document classification task. We run all models 10 times and report mean ± standard deviation. Text GCN significantly outperforms baselines on 20NG, R8, R52 and Ohsumed based on student t-test (p ¡ 0.05)

Claim 1: Text GCN outperforms baseline models on text classification tasks.

Evidence 1: Figure 7 represents test accuracy results, where Text GCN significantly outperforms all baseline models on datasets 20NG, R8, R52, and Ohsumed, based on a student t-test (p ¡ 0.05).

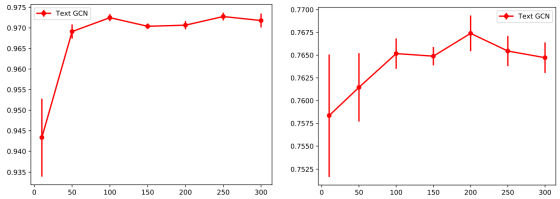
Claim 2: The performance of Text GCN is influenced by the choice of window size in the text graph.

Evidence 2: Figure 8 illustrates how the test accuracy varies with different sliding window sizes in experiments conducted on datasets R8 and MR. The results show that test accuracy improves with larger window sizes, but the average accuracy plateaus when the window size exceeds 15, indicating an optimal range for window size choice.



(a) R8 (b) MR

Figure 8. Test accuracy with different sliding window sizes.



(a) R8 (b) MR

Figure 9. : Test accuracy by varying embedding dimensions

Claim 3: The dimension of the first-layer embeddings in Text GCN plays a crucial role in classification performance.

Evidence 3: Figure 9 demonstrates the impact of different embedding dimensions on test accuracy in the R8 and MR datasets. The results suggest that choosing an appropriate dimension for embeddings is essential, as overly low or high dimensions do not lead to improved classification performance.

5.4. Critique and Discussion

The paper's strengths lie in its comprehensive literature review, clearly identifying a research gap, and outlining the contributions of Text GCN effectively. The claim-evidence structure used to present findings is a strong point, especially the evidence showing that Text GCN outperforms baseline models. However, to enhance the paper, it could provide more in-depth explanations of why Text GCN outperforms other models, offer insights into the underlying mechanisms, and discuss the broader practical implications of the findings.

6. Implementation

6.1. Implementation Motivation

By implementing CGNNs and experimenting with them, I hope to gain insights into how continuous dynamics can help overcome the over-smoothing problem often encountered in traditional graph neural networks. CGNNs use a continuous approach to model relationships in graph-structured data, and I'm curious to see how this affects their ability to capture long-term dependencies in graphs. Overall, I expect that implementing CGNNs will provide me with a deeper understanding of their capabilities and limitations, and how they can be useful in various data-driven applications.

6.2. Implementation Setup and Plan

In the implementation process, Python will serve as the primary programming language, with reliance on well-established deep learning frameworks such as PyTorch. These frameworks provide the essential tools for constructing and training graph neural networks. The primary goal of this phase is to ensure a comprehensive grasp of the paper's approach and to validate the results as reported in the paper.

In terms of evaluation metrics, the primary focus will center around standard graph-based assessment criteria. This includes accuracy and specific loss functions tailored to reduce oversmoothing or overfitting. The core objective of this implementation is to effectively demonstrate CGNN's superiority over existing methods. This superiority is demonstrated through its capability to capture long-term dependencies, mitigate over-smoothing issues, and maintain memory efficiency, in alignment with the emphases in the paper.

As for the codebase, the starting point will be the provided CGNN codebase available at `https://github.com/DeepGraphLearning/ContinuousGNN`. The implementation sequence is structured with a primary focus on initially replicating the paper's results. Following this, the scope of the experiments will be extended to evaluate CGNN and WCGNN that is tailerd to different convolution/linear networks, changes in hyperparameters based on trial and error, adding more loss functions to reduce loss and weight initialization and also changes in activation fucntion along with a few custom functions.

6.3. Implementation Details

Model	Reported (%)	Reproduced (%)	Experimented (%)
CGNN	82.7	82.9	79.6
WCGNN	82.1	83.2	81.5

Table 1. Comparison of Reported, Reproduced, and Experimented Results

Table 1 provides a comprehensive comparison of reported, reproduced, and experimentally modified results for two models, CGNN and WCGNN. The 'Reported' values denote the performance percentages initially presented by the

authors in their respective papers. On the other hand, 'Reproduced' values represent the model's performance in its original state, without any modifications or additional experiments. The 'Modification' values illustrate the model's performance after incorporating changes or experiments that I have done for the final checkpoint.

I have made changes to CGNN and WCGNN where CGNN stands for continuos graph neural network whereas WCGNN stands for weight continuos graph neural network.

In my reimplementation of the Continuously Evolving Graph Networks (CE-GNN) paper, I introduced several modifications to the original architecture to explore its flexibility and potentially improve performance. Firstly, I expanded the GNN depth by adding an extra Graph Convolutional Network (GCN) layer ('conv2') within the Ordinary Differential Equation (ODE) . GCN was used insted of the author's linear layer as I wanted to capture more graph data which the linear layers could not. This alteration aims to enhance the model's capacity to capture intricate relationships in the data. Additionally, I incorporated a linear layer in WGNN as the final layer of the model to further refine the node embeddings before classification. The activation function and dropout strategy were retained, with ReLU serving as the activation after solving the ODE and dropout applied subsequently. This decision was made to maintain the original GNN's non-linearity while introducing a regularization mechanism for better generalization. CGNN has two GCN whereas WCGNN has two GCN and one fully connected Linear layer.

In my adaptation of the ODE function module, I made deliberate changes to explore the impact of alternative activation functions and coefficient adjustments. The ODE function, responsible for capturing the system dynamics in the Continuous Graph Networks (CE-GNN), underwent several modifications to assess their implications on the model's learning behavior.Firstly, I replaced the original sigmoid activation function with a leaky ReLU activation in the forward pass . This choice introduces a controlled amount of non-linearity to the model, potentially allowing it to capture more complex patterns in the evolving graph.Additionally, I adjusted the coefficient in the differential equation ('alph * 0.6 * (ax-x)') to 0.6, deviating from the original factor of 0.5.

In the pursuit of improving the training dynamics and generalization capabilities of the continuous graph neural network, I introduced several modifications and strategies to the original code. These changes aim to explore alternative weight initialization and regularization techniques to enhance the model's performance.

1. **Weight Initialization:** Xavier Uniform Initialization: I implemented Xavier uniform weight initialization specifically for linear layers in the model. This initialization method is known for mitigating issues related to vanishing or exploding gradients, contributing to a more stable and efficient training process. The weight initialization was done for linear layers only. As I have only one linear fully connected layer it is performed once in the WGNN. Since CGNN has no weight there was no linear layer added to it and hence the weight initialization does not work in CGNN.
2. **L1 Regularization:** To encourage sparsity in the model's weights and prevent overfitting, I incorporated L1 regularization into the loss function. The regularization term (`self.opt['decay_l1'] × l1_reg`) penalizes the absolute values of the weights.
3. **L2 Regularization:** The term (`self.opt['decay'] × l2_reg`) is integrated into the loss function to control the magnitude of weights and prevent overfitting. This term penalizes the squared values of the weights, discouraging overly large weight values that might lead to model overfitting.

The loss function now includes both L1 and L2 regularization terms, providing a comprehensive regularization approach. The coefficients (`self.opt['decay']` and `self.opt['decay_l1']`) control the strength of each regularization technique. By experimenting with different regularization techniques and weight initialization strategies, the aim is to strike a balance between model complexity and generalization, ultimately leading to improved performance in scenarios with limited labeled data.

A lot more hyperparameters where changed in WCGNN like the alpha was reduced based on trial and error and also the hidden layers were increases from 16 to 40 to capture more data.

6.4. Results and interpretation

The 'Modification' results shed light on the models' adaptability to changes or experimental interventions. CGNN experiences a notable decrease in performance (79.6 percent), indicating a potential sensitivity to modifications. On the other hand, WCGNN maintains a commendable performance level even after modifications (81.5 percent), suggesting a greater resilience to alterations. Figure 10 and Figure 11 are the evidence of the output accuarcy after experiments. These outcomes align with our expectations to some extent, as we anticipated some variability in model performance under modifications. However, the extent of CGNN's sensitivity underscores the importance of cautious modifications in certain model architectures. My assumption that using GCN instead of linear layers in both CGNN and WCGNN to capture data in graph, using weight initialization to reduce vanishing gradients and also using L1 and L2 regularization

```
Epoch: 386 | Loss: 0.123 | Dev acc: 0.734 | Test acc: 0.749 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 387 | Loss: 0.121 | Dev acc: 0.740 | Test acc: 0.758 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 388 | Loss: 0.111 | Dev acc: 0.744 | Test acc: 0.762 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 389 | Loss: 0.113 | Dev acc: 0.746 | Test acc: 0.767 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 390 | Loss: 0.117 | Dev acc: 0.744 | Test acc: 0.754 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 391 | Loss: 0.095 | Dev acc: 0.744 | Test acc: 0.755 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 392 | Loss: 0.122 | Dev acc: 0.746 | Test acc: 0.762 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 393 | Loss: 0.138 | Dev acc: 0.742 | Test acc: 0.775 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 394 | Loss: 0.107 | Dev acc: 0.752 | Test acc: 0.769 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 395 | Loss: 0.123 | Dev acc: 0.750 | Test acc: 0.768 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 396 | Loss: 0.143 | Dev acc: 0.740 | Test acc: 0.763 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 397 | Loss: 0.123 | Dev acc: 0.742 | Test acc: 0.765 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 398 | Loss: 0.117 | Dev acc: 0.730 | Test acc: 0.760 | Forward: 3 2.995 | Backward: 0 0.000
Epoch: 399 | Loss: 0.152 | Dev acc: 0.718 | Test acc: 0.752 | Forward: 3 2.995 | Backward: 0 0.000
79.600
```

Figure 10. Code snippet for cgnn with enhanced code

```
Epoch: 389 | Loss: 0.207 | Dev acc: 0.790 | Test acc: 0.809 | Forward: 294 301.913 | Backward: 0 0.000
Epoch: 390 | Loss: 0.170 | Dev acc: 0.794 | Test acc: 0.820 | Forward: 294 301.893 | Backward: 0 0.000
Epoch: 391 | Loss: 0.211 | Dev acc: 0.764 | Test acc: 0.780 | Forward: 294 301.872 | Backward: 0 0.000
Epoch: 392 | Loss: 0.216 | Dev acc: 0.790 | Test acc: 0.806 | Forward: 294 301.852 | Backward: 0 0.000
Epoch: 393 | Loss: 0.204 | Dev acc: 0.794 | Test acc: 0.815 | Forward: 294 301.832 | Backward: 0 0.000
Epoch: 394 | Loss: 0.199 | Dev acc: 0.776 | Test acc: 0.801 | Forward: 294 301.813 | Backward: 0 0.000
Epoch: 395 | Loss: 0.219 | Dev acc: 0.798 | Test acc: 0.819 | Forward: 294 301.793 | Backward: 0 0.000
Epoch: 396 | Loss: 0.186 | Dev acc: 0.786 | Test acc: 0.809 | Forward: 294 301.773 | Backward: 0 0.000
Epoch: 397 | Loss: 0.178 | Dev acc: 0.788 | Test acc: 0.811 | Forward: 294 301.754 | Backward: 0 0.000
Epoch: 398 | Loss: 0.202 | Dev acc: 0.784 | Test acc: 0.810 | Forward: 294 301.734 | Backward: 0 0.000
Epoch: 399 | Loss: 0.195 | Dev acc: 0.798 | Test acc: 0.813 | Forward: 294 301.715 | Backward: 0 0.000
81.500
```

Figure 11. Code snippet for wcgnn with enhanced code

for reduction of overfit data and trial and error hyperparameter changes have not made major accuracy changes.

While our results provide valuable insights into the behaviors of CGNN and WCGNN, it's crucial to acknowledge the limitations of our study. The negative results, such as the decrease in CGNN performance after modification, emphasize the need for a nuanced understanding of model adaptability. Our findings neither validate nor invalidate specific hypotheses outlined in the motivation section but rather open avenues for further investigation and refinement.

7. Conclusion and Discussion

This journey has revealed how subtle adjustments in architecture, such as weight initialization and regularization, fine tuning various layers can significantly impact model performance. As I had tested my code on CORA dataset, I was unable to test it on various other datasets that are linked to graphs. Also the assumption that replacing linear with GCN would help with the capturing of long range dependancy did not give any positive nor negative results.Future work could focus on refining model adaptability, exploring diverse datasets, and addressing computational complexities, contributing to advancements in continuous graph network research.

References

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xhonneux, L.-P., Qu, M., and Tang, J. Continuous graph neural networks. In *International Conference on Machine Learning*, pp. 10432–10441. PMLR, 2020.

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.

(Xhonneux et al., 2020) (Li et al., 2018) (Yao et al., 2019)