Implementation of Weight Adjusting GNN With Differentiable Pooling for User Preference-aware Fake News Detection

Jay Prakash Maurya (jpeemaurya@gmail.com)
VIT Bhopal University
Vivek Richhariya
Lakshmi Narain College of Technology, Bhopal
Bhupesh Gour
Lakshmi Narain College of Technology, Bhopal
Vinesh Kumar
VIT Bhopal, University

Research Article

Keywords: fake news detection, GNN, GCN, UPFD, training loss, training accuracy

Posted Date: February 9th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3942141/v1

Additional Declarations: The authors declare no competing interests.

Abstract

In the last few years, false news has hurt people and society, drawing attention to classify and identify news as fake or True. Major fake news detection algorithms either largely trust textual information via learning the internal knowledge of the extracted news material or writing style, or they focus on mining news content. To differentiate between fake and real news, the proposed experiment processes news information as a graph neural network with an attention-based differentiable pooling model. This sets the way for the user preference-aware fake detection (UPFD) in a graph-based structure. The attention-based differentiable pooling approach allows GNNs to adaptively extract information from the network by focusing on the most relevant nodes for a given task. One significant improvement is in the way the input data is formatted for the learning schema; in paired scenarios, tweet vectors are essential. Each pair includes a potential fake vector and a true vector; the latter's classification accuracy depends on how similar or different it is from the former. In particular, when it comes to historical events, the novel way that knowledge sets are handled in graph form and arranged in pairs of related terms provides a unique method for determining the veracity of news. To improve validation accuracy and learning, the proposed GNN-DP model also presents a comparison between the standard layer and the embedding layer. Moreover, comprehensive analyses and direct comparisons of the graph convolutional network (GCN) model's performance have been achieved by experimental evaluations.

I. INTRODUCTION

Due to its low cost, ease of use, and quick distribution of information, the Internet has completely changed the way we communicate and interact [1]. As a result, traditional newspapers are no longer as popular for news searches and reading as social media and online portals. Users can share and consume content about their daily lives with great ease thanks to online social network platforms [2]. But these platforms also make it possible for rumors and fake news to spread quickly and cheaply [3]. Especially during big events like national elections or pandemics, maliciously produced fake news can have a devastating effect on society [4]. Even though social media is an effective information tool, it can negatively impact society by influencing important events [5]. The aim of the proposed work in the context of news fraud with many contents and contextual information is to ascertain the news's authenticity and, ideally, categorize it as false or real automatically. Various methods have been used to examine news material and have even gone so far as to collect data from relevant sources [6]. These entities include implicit associations like other news pieces on related themes, as well as explicit ones like people who share, comment, or respond to the news [7].

FAKE NEWS DETECTION

There are several newly proposed methods for detecting false news that may be broadly classified into two major classes. Which first pipeline of the news is based on text pattern, for this model is just trained on input the text pattern found in the news article [8]. Various works are usually centered around distinct types of patterns. Using social media responses like likes, comments, and reposts, some studies try to

confirm the veracity [9]. Emotional pattern mining has received increased attention recently [10], with the underlying premise being that fake news likely contains blatant sentiment biases. To verify the accuracy of news reports, researchers suggest using the second pipeline, which is evidence-based, to examine semantic similarity or conflict in claim-evidence pairs [11]. Typically, searches with unverified assertions are used to retrieve evidence from fact-checking websites or the knowledge graph. The score to claim every word in the evidence of user awareness reflected with graph association, it then presents an attention-based interaction [12]. The subsequent techniques, which are similar to the groundbreaking work, use sequential techniques to issue the semantic embeddings that can applied to various coarseness [13]. The procedures for utilizing machine learning to detect fake news are given in Fig. 1.

II. NEURAL NETWORK ARCHITECTURES FOR FAKE NEWS DETECTION

We employed many deep neural network versions, which are covered in more detail in later sections, in light of the noteworthy advancement in neural network research. This section talks about the various neural network architecture variants that are used to identify bogus news.

Convolution Neural Networks (CNN)

Neurons along with learnable weights and biases make CNN comparable with regular neural structure. CNN consists input layer, several hidden layers, and an output layer [15]. As seen in Fig. 2, a CNN's hidden layers normally include convolutional, pooling, fully connected, and normalizing layers.

Graph Neural Network (GNN)

Designed to function directly on graph structures, Graph Neural Networks (GNNs) is a kind of neural network that acquires node representations by neighborhood propagation/aggregation [16]. Spectral techniques and spatial approaches are the two primary types of GNNs. GNNs have demonstrated efficacy in applications like text classification, sentiment analysis, recommender systems, and long-distance structural link capture in graphs. Node classification is a popular application where the network predicts node labels without using ground truth, handling different types of graphs without requiring pre-processing procedures. In terms of extracting structural features from UPFD self-learning graphs, GNNs perform well comparative to deep-learning techniques. Nevertheless, they are susceptible to noise in the dataset; even a tiny bit of noise added or removed from edges or nodes can hurt the output of a GNN.

Graph convolutional network (GCN)

Neural networks called graph-convolutional networks, or GCNs, are made for graph-structured data and are becoming more and more common in applications where the interactions between entities are naturally represented as graphs [17]. Nodes in GCNs represent entities, while edges represent relationships. By applying convolutional procedures to the graph, the fundamental principle of CNN is extended to irregular graph structures. By updating a node's representation with information from nearby

nodes, this action enables the model to capture local graph structure. Information is propagated through the graph by many layers of GCN, which also capture hierarchical aspects and improve node representations [18]. The model output is generated by the last layer, which is the Softmax layer that performs classification. Stochastic gradient descent and backpropagation are two methods used to train GCNs on labeled data [19]. Their exceptional ability to capture complex relationships in graph-structured data makes them useful for applications that highlight these dependencies, such the identification of false news. Graph neural networks, or GCNs, are a basic type of neural network with a detailed design that is shown in Fig. 3.

Contribution of Work

While there have been many previous attempts to address the issue of identifying false news, most published solutions rely on a limited range of publicly available, widely acknowledged, and verified real/fake news data. The current method sets the standard by separating genuine news from fake news using the UPFD social media dataset in a graph-based framework. Another novel feature of the proposed approach is the format of the input to the learning schema. Tweet vectors are primarily utilized in paired scenarios. There is a potential false vector and a real vector in every pair. Depending on how the later and the former differ or are similar, it must be properly categorized. To determine the news validity from previous occurrences, knowledge sets that are structured in graph form are arranged differently and in pairs of related words. Additionally, to increase the model's accuracy during validation and learning, the GNN-DP model that was suggested compared the embedding layer with the normal layer. Furthermore, model (GCN) experiments were carried out to evaluate and directly compare their performance.

III. LITERATURE REVIEW

Several creative frameworks that make use of graph-based models, semantic structures, and user preferences have been put forth for the detection of fake news.

The goal of Yingtong Dou et al.'s [1] UPFD framework is to jointly model graphs and content to capture user preferences. It encodes news material and user historical postings using several text representations learning techniques, creating a propagation graph for social media user-sharing cascades. According to Yuxiang Ren and Jiawei Zhang [2] node representation learning in networks, presented, a Hierarchical Graph with a attention mechanism. GET is a graph-based semantic structure mining tool that was presented by Weizhi Xu et al. [3]. It captures long-distance semantic interdependence through neighborhood propagation and models claims and evidences as graph structures. Adversarial contrastive learning is used in GETRAL by Junfei Wu et al. [4] to improve representation and investigate intricate semantic structures using claim and evidence graphs. KAN, which uses external knowledge to form a graph for fake news detection, was given by Yaqian Dun et al. [5]. Chenguang Song et al.'s study [6] used temporal-aware and structure-aware modules to investigate dynamic news transmission with DGNF. Node temporal interactions for dynamic evolution patterns in news propagation are modeled by Chenguang Song et al. [7] in their TGNF model. MVAN, a Multi-View Attention Network, was introduced by

Shiwen Ni et al. [10]to detect and produce explanations for bogus news. AENeT was introduced by Vidit Jain et al. [11]and uses attention mechanisms to increase accuracy on the LIAR dataset. In a thorough review, Shuzhi Gong et al. [14] divided graph-based techniques into three categories: knowledge-driven, propagation-based, and diverse social context-based methods. M. F. Mridha et al. [16] examined sophisticated methods for detecting fake news, highlighting the negative effects of false information and examining NLP approaches, DL structures, and evaluation metrics.

Table 1
Summary of Recent Works on Fake News Detection

Methodology used	Key Features	Results/Findings	Authors
UPFD	user preferences using content and graph modeling.	Utilizes GNNs for joint embeddings, effectively identifying fake news.	[1]
HGAT	Hierarchical attention in HIN for node representation learning.	Outperforms text-based and other network-based models in graph representation and node classification.	[2]
Graph-based semantic structure (GET)	Models claims and evidences as graph structures for long-distance semantic dependencies.	Enhances fake news detection through neighborhood propagation and graph structure learning.	[3]
GET with constructive learning (GETRAL)	Adversarial contrastive learning for representation enhancement.	Captures long-distance semantic dependencies, reduces redundant information, and performs well compared to other methods.	[4]
KAN	Leverages external knowledge from a knowledge graph.	Measures knowledge importance, integrating semantic and knowledge-level representations, showcasing superiority.	[5]
DGNF	graph-based fake news detection on dynamic news propagation graph	Effectively leverages temporal propagation along with network structure information for accurate predictions.	[6]
TGNF	graph neural network for fake news detection Temporally generated news	Incorporates temporal propagation information, faces challenges in information propagation per interaction.	[7]
GCN	attention mechanisms on rumor-type / spread news.	Superior performance compared to other rumor detection methods, captures dynamic nature of rumor propagation.	[8]
MMCN	Multi-level semantics from textual and visual content.	Utilizes BERT and ResNet models, exhibits superior performance on WEIBO and PHEME datasets.	[9]
MVAN	fake news detection and explanation generation.	Demonstrates strong performance and interpretative abilities, applicable to various social media text classification tasks.	[10]
AENeT	metadata attributes and news statements.	methods to improve accuracy on dataset LIAR, leveraging attention weights for improved performance.	[11]
DAGANN	cross-domain fake news detection.	Requires only partial domain sample data, with few samples to detect fake news.	[12]

IV. PROPOSED METHOD

During the pooling phase, attention weights are learned for every node rather than considering them all equally. After that, a weighted total of node representations is calculated using these weights, with nodes with higher attention weights contributing more to the pooled representation. DIFFPOOL, the proposed graph pooling technique, may be utilized end-to-end with various GNNs that potentially provide graphs with hierarchical representations. In each layer of a deep GCNN, DIFFPOOL establishes a distinguished soft cluster assignment for each node, transferring nodes to several clusters that serve as the coarsened input for the subsequent layer of the GNN. Figure 4 shows the architecture of the suggested methodology.

5.1 Proposed Method (After Weight adjustment over nodes)

Algorithm : Proposed Method

Let a graph G'= (V', E'), its adjacency matrix A ∈{0,1}|v' | |v'|.

1. **Input**:

• G' with features *X* and adjacency *A*.

• Graph-level target *Y* for graph classification.

2. **Initialization**:

• Initialize node embeddings$H^{(0)} = X$

• Set graph embedding $Z^{(0)}$as an initial node embedding$H^{(0)}$

3. **GNN Layer with DiffPool**:

• For l = 1 to L (GNN layers):

i. Standard GNN layer$H^{l} = \sigma\left(\widehat{A}^{(l)} H^{(l-1)} W^{(l)}\right)$

ii. Graph Pooling$S^{l} = Softmax\left(MLP(H^{l}\right)$

Where **MLP** is multilayer perceptron

• Compute the Coarsened adjacency matrix$A^{(l+1)} = S^{(l)}.\left(S^{(l)}\right)^{T}$

• Compute the Coarsened node embeddings$H^{(l+1)} = \left(S^{(l)}\right)^{T}.H^{(l)}$

• Update graph embedding$Z^{(l)} = Z^{(l-1)} + H^{(l)}$

4. **Graph Classification Head**:

• After the last GNN layer, apply a pooling operation to obtain the final graph embedding$Z^{(L)}$.

• Use $Z^{(L)}$ as input to a classifier to predict the graph-level output$\widehat{Y}$

5. **Loss Function**: Cross entropy loss for classification predicted graph-level output $\widehat{Y}$ and ground truth$Y$.

Algorithm : Proposed Method
End Algorithm

V. IMPLEMENTATION AND RESULTS

Dataset used

The UPFD dataset https://doi.org/10.1145/3404835.3462990 , which includes both authentic and fraudulent news networks on Twitter, was gathered using fact-checking websites like Politifact and Gossip Cop. This study makes use of the Politifact dataset, which has N=314 graphs, 157 of which are connected to false information. Social network tree structures are a unique type of graph in which news items are the root nodes and users who retweeted on news articles at the root are represented as leaf nodes. Each edge represents a user's retweet activity; they can retweet news directly or indirectly.

Configuration and implementation

The outcomes of the suggested endeavor are contrasted concerning the application of two models, GCN and GNN, utilizing differentiable pooling. The suggested GNN-DP technique builds a customized technique consisting of Graph Segregation and Batch normalization layers after the differential Pooling module is added. This calls the previously established GNNs for cluster assignment and node embedding processing. GCN performance was compared with parallel GNNs. First, we choose to do differential pooling twice before utilizing mean pooling and linear transformation to arrive at the final Softmax prediction. We then gradually reduce the size of the cluster from 500 to 100 and 100 to 20 (20% at a time). The combination that best suits the UPFD data must be chosen to prepare a design decision on the size reduction rate and number of layers. As seen in figures 5(a), (b), and (c), respectively, the distribution of the dataset new over sample categories in Train, Validation, and Test has been provided in colored graphs.

Fake news code implementation in the graph is derived from the already-implemented PyG package, and preparation is easily covered in the PyTorch dataset portion. Two primary preprocessing tasks were completed: first, the directed social network was cast to an undirected state, and second, the load in node feature was mapped (every news and every user mapping). With a maximum node of 500 for experimentation, the profile attributes include 10 dimensions encoded using a BERT model based on their previous Tweets, which contain 768 dimensions.

Table 2: Node x and y dimension set for data under Train, Validation, and Test dataset to create adjacency matrix 500 x 500

	train	val	test
x	(6072, 778)	(3778, 778)	(31204, 778)
y	(62,)	(31,)	(221,)
edge_idx	(2, 12020)	(2, 7494)	(2, 61966)

GCN Code

The trial was finished using the GPU-equipped Google Collab environment, which is hosted on Google Cloud. In Figure 6, the GCN Model Configuration Code is displayed. In order to work on graph implementation over UPFD using Python programming, the installation of the torch_geometry library over the torch package was completed, as illustrated in figure 7. There are one linear layer and two convolutional layers in a GCN design. In between the two convolutional layers, there is one layer of batch normalization as well. The functions shown in figure 8, figure 9 initiates number of layers like convolutional or batch normalization. The user input number of layers (num_layers) must be more than two.

Table 3 displays the training and testing outcomes of the GCN model for the parameter's accuracy, precision, F1-Score, and AUC across the three datasets used for the UPFD new, Validation Test, and Test dataset. Experiments on a test dataset yield 0.82 accuracy, 0.88 precision, and 0.82 F1-Score for the GCN model.

Table 3: GCN model Performance

Model	Dataset	Accuracy	AUC	F1- Score	Precision
GCN	UPFD New	0.9677	0.85	0.83	0.90
	Validation Test	0.8065	0.824	0.81	0.87
	Test	0.82	0.8233	0.82	0.88

GNN-DP

Differential pool construction on the third GNN pooling layer by comparing pool clusters for the first and second to the goal torch size of ([14, 20, 64]) using the pool's configuration as shown in [table 4]. The code snapshot depicting the GNN and GNN DP configuration is displayed in Figures 8 and 9, respectively. Table 4 displays the GNN DP cluster form arrangement over pooling layers. SoftMax pooling is used for the last adjacent node output in the GNN model following two-layer pooling.

Table 4: GNN DP model Performance

TorchSize	Cluster Assignment	embedding shape	embedding shape	adjacency matrix shape
First Pool	[16, 500, 100]	[16, 500, 64]	[16, 100, 64]	[16, 100, 100]
Second Pool	[16, 100, 20]	[16, 100, 64]	[16, 20, 64]	[16, 20, 20]

Comparison of Results

Plotting of the GCN and GNN-DP comparison graphs is displayed in Figures 11 and 12, respectively. Graphs make it abundantly evident and GNN technique has a great potential for quick and accurate detection of fake news found within a social network with minimum loss and high accuracy on training and validation sets of datasets. When it comes to the news dataset and graph adjacency, GCN validation accuracy and validation loss have not been stable at the same time, but GNN-DP exhibits steady results over the training and testing parameter epochs. After running the model, GNN-DP obtained the following results: loss_train: 0.3330, acc_train: 1.0000, loss_val: 0.6916, acc_val: 0.6129.

VI. CONCLUSIONS

This article focused on optimizing the hyperparameters and architecture of their graph neural networks, acknowledging the challenge of fact-checking news items. GNN-DP In comparison to GCN, training precision is improving, and training loss is lower. Result graphs shows comparison of implemented models for detecting fake news over UPFD on both techniques GCN and GNN with differential pooling and helps to choose the model. Additionally, it is suggested that differentiated pooling, a more expressive method, would produce superior results, however, GNN-DP might be too advanced for this study. GNN-DP may nevertheless be a beneficial choice in practical situations involving bigger news dispersion networks despite these drawbacks.

Declarations

Contribution from Proposed Work.

Although there have been numerous attempts to tackle the problem of fake news identification in the past, the majority of published solutions rely on a small set of real/fake news data that is already available, broadly acknowledged, and confirmed. The current approach paves the way with UPFD social media dataset in a graph-based structure that distinguishes between bogus and true news. The way the input to the learning schema is formatted is another innovative aspect of the suggested work. In a paired situation, tweet vectors are specifically used. In each pair, there is a true vector and a possible fake vector. The latter's proper classification depends on how similar or different it is from the former. To handle knowledge sets structured in graph form are different and organized in pairs of related words to identify news validity from past events. Moreover, the proposed model GNN -DP compared the embedding layer with the normal layer to improve the learning and validation accuracy of the model. Additionally, experiments with the model (GCN) conducted, to assess and compare directly their performance.

References

1. Dou, Yingtong, et al. "User preference-aware fake news detection." Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021, pp 2051-2055.
2. Ren, Yuxiang, and Jiawei Zhang. "Fake news detection on news-oriented heterogeneous information networks through hierarchical graph attention." 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021, pp 1-8.
3. Xu, Weizhi, et al. "Evidence-aware fake news detection with graph neural networks." Proceedings of the ACM Web Conference 2022. 2022, pp 2501-2510.
4. Wu, Junfei, et al. "Adversarial contrastive learning for evidence-aware fake news detection with graph neural networks." IEEE Transactions on Knowledge and Data Engineering (2023), pp 1-12.
5. Dun, Yaqian, et al. "Kan: Knowledge-aware attention network for fake news detection." Proceedings of the AAAI conference on artificial intelligence. Vol. 35. No. 1. 2021, pp 81-89.
6. Song, Chenguang, et al. "Dynamic graph neural network for fake news detection." Neurocomputing 505 (2022): 362-374.
7. Song, Chenguang, Kai Shu, and Bin Wu. "Temporally evolving graph neural network for fake news detection." Information Processing & Management 58.6 (2021): 102712, pp 1-18.
8. Choi, Jiho, et al. "Dynamic graph convolutional networks with attention mechanism for rumor detection on social media." Plos one 16.8 (2021): e0256039, pp 1-17.
9. Ying, Long, et al. "Multi-level multi-modal cross-attention network for fake news detection." IEEE Access 9 (2021), pp 132363-132373.
10. Ni, Shiwen, Jiawen Li, and Hung-Yu Kao. "MVAN: Multi-view attention networks for fake news detection on social media." IEEE Access 9 (2021): 106907-106917
11. Jain, Vidit, et al. "AENeT: an attention-enabled neural architecture for fake news detection using contextual features." Neural Computing and Applications 34.1 (2022): 771-782.
12. Yuan, Hua, et al. "Improving fake news detection with domain-adversarial and graph-attention neural network." Decision Support Systems 151 (2021): 113633, pp 1-11.
13. Li, Jia, and Minglong Lei. "A brief survey for fake news detection via deep learning models." Procedia Computer Science 214 (2022), pp 1339-1344.
14. Gong, Shuzhi, et al. "Fake news detection through graph-based neural networks: A survey." arXiv preprint arXiv:2307.12639 (2023), pp 1-18.
15. Mridha, Muhammad F., et al. "A comprehensive review on fake news detection with deep learning." IEEE Access 9 (2021), pp 156151-156170.
16. Varlamis, Iraklis, et al. "A survey on the use of graph convolutional networks for combating fake news." Future Internet 14.3 (2022): 70, pp 1-19.
17. Graham, Matthew H., and Ethan Porter. "Increasing demand for fact-checking." (2023), pp 1-62.

18. Baarir, Nihel Fatima, and Abdelhamid Djeffal. "Fake news detection using machine learning." 2020 2nd International workshop on human-centric smart environments for health and well-being (IHSH). IEEE, 2021, pp 125-130.
19. Shaikh, Jasmine, and Rupali Patil. "Fake news detection using machine learning." 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). IEEE, 2020, pp 1-5.

Figures

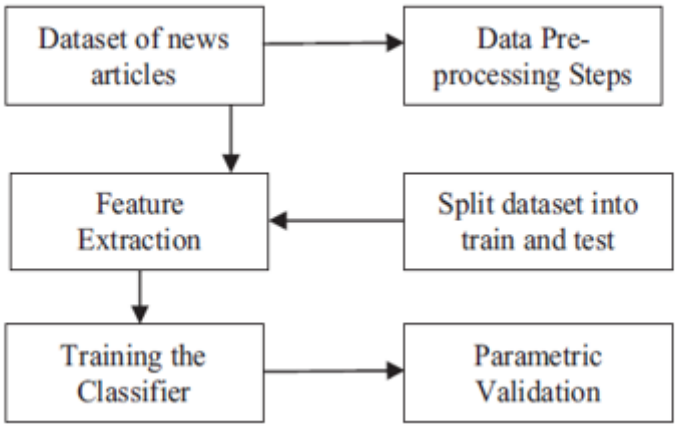


Figure 1

General Flowchart for fake news detection

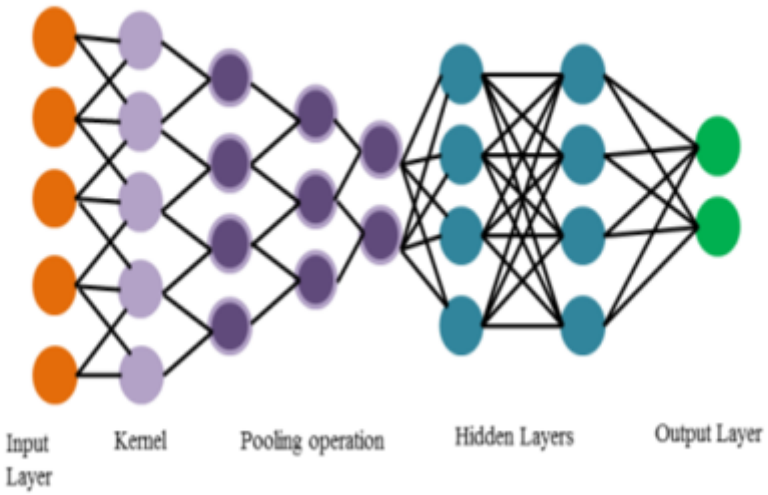


Figure 2

Architecture of CNN

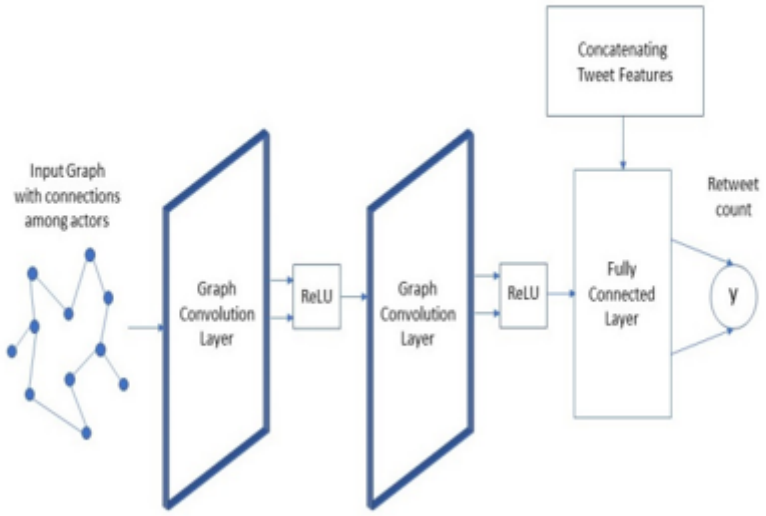


Figure 3

Architecture of the Graph Convolution Neural Network Model

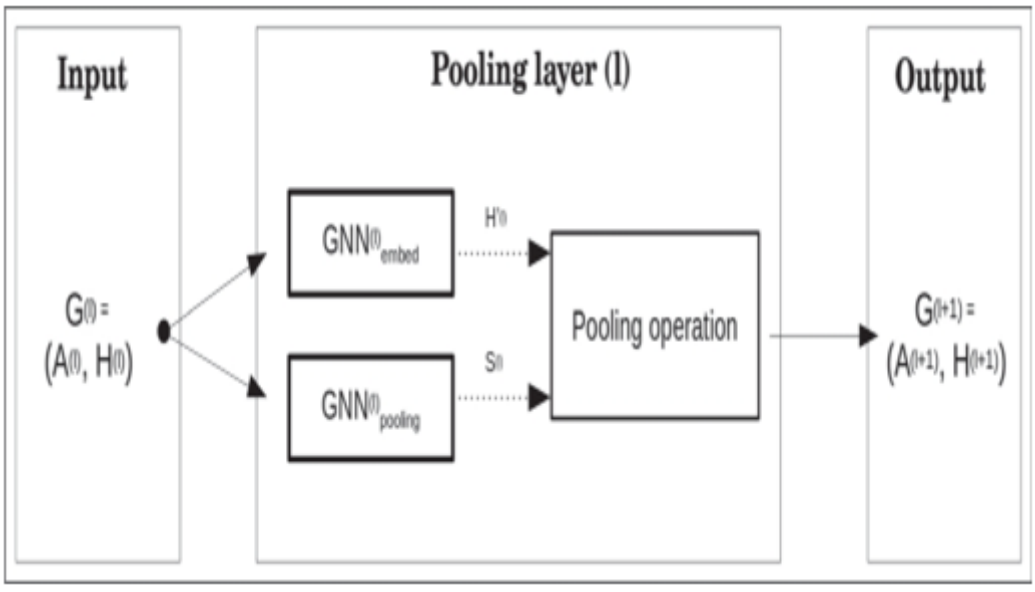
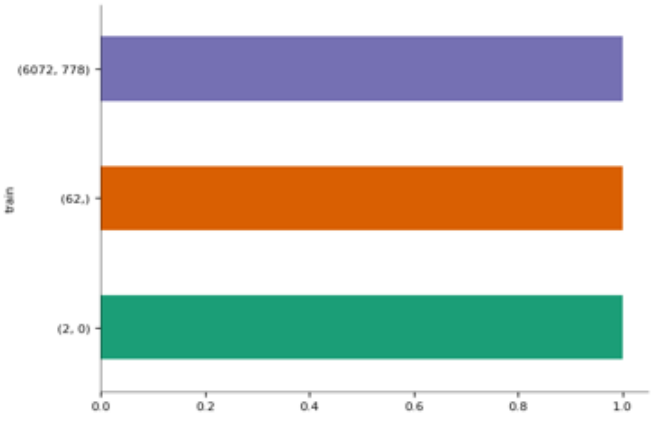
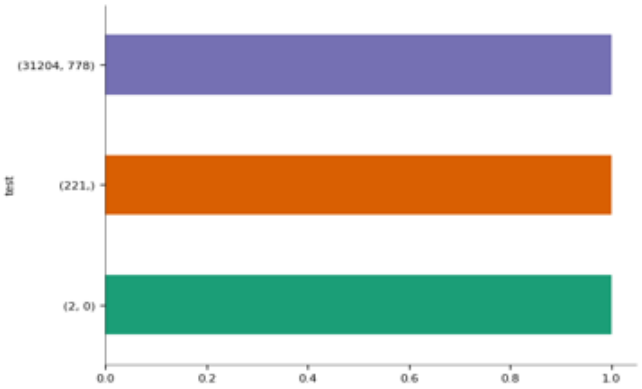


Figure 4

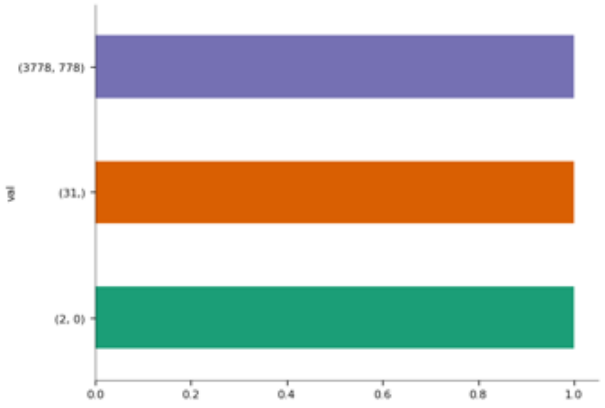
Proposed methodology



(a) Training



(b) Testing



(c) Validation

Figure 5

Graph creation and distribution of news for Training Testing and Validation

```python
class GCN(torch.nn.Module):
  def __init__(self, args):
    """
    Initialize a simple GCN with specific parameters.

    By default, this GCN has 2 convolutional layers and 1 linear layer.
    It also has 1 layer of batch normalization between the two conv-layers.

    If num_layers is provided by user, then this function initializes
    corresponding number of convolutional layers, with batch normalization
    in between. Num_layers must be >= 2.

    ------------------------------------
    self: GCN object
    args["num_features"]: dimension of the input
    args["hidden_dim"]: dimension of the hidden layer(s)
    args["num_classes"]: dimension of the output (i.e. number of classes)
    args["dropout"]: percentage of neurons being zeroed, can be None
    args["num_layers"]: number of convolutional layers, must be >= 2

    """

    assert args.num_layers >= 2, "num_layers must be >= 2."

    super(GCN, self).__init__()

    # Initialize parameters
    self.num_layers = args.num_layers
    self.dropout = args.dropout
```

Figure 6

GCN Model Configuration Code

```
-------------------------------------------
    self: GCN as an object
    args["InputDimension"]:
    args["Hidden_layer_dimension"]
    args["dimensions_output"]:
    args["dropout"]:
    args["num_layers"]:
-------------------------------------------
```

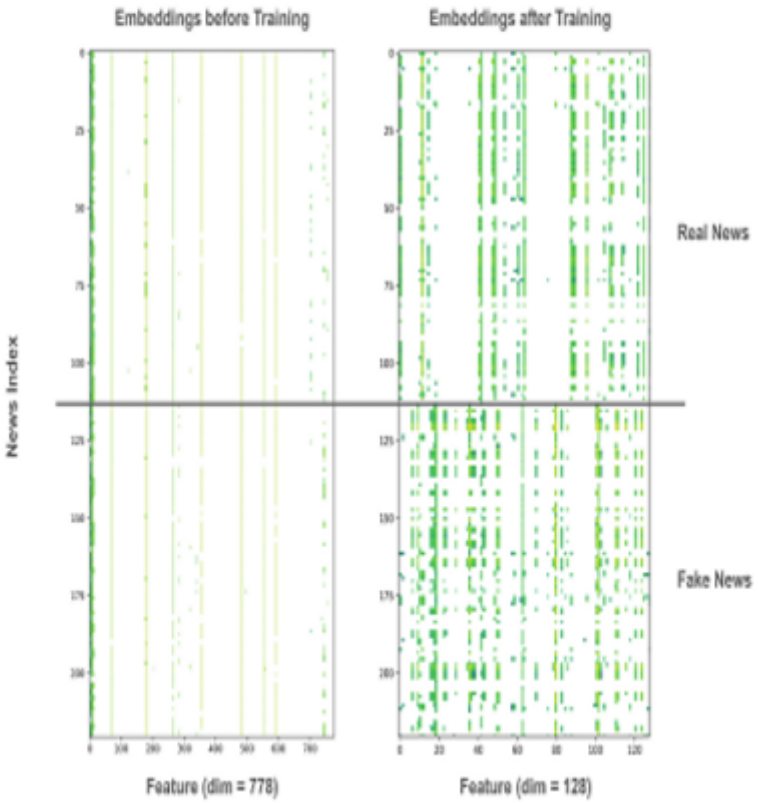


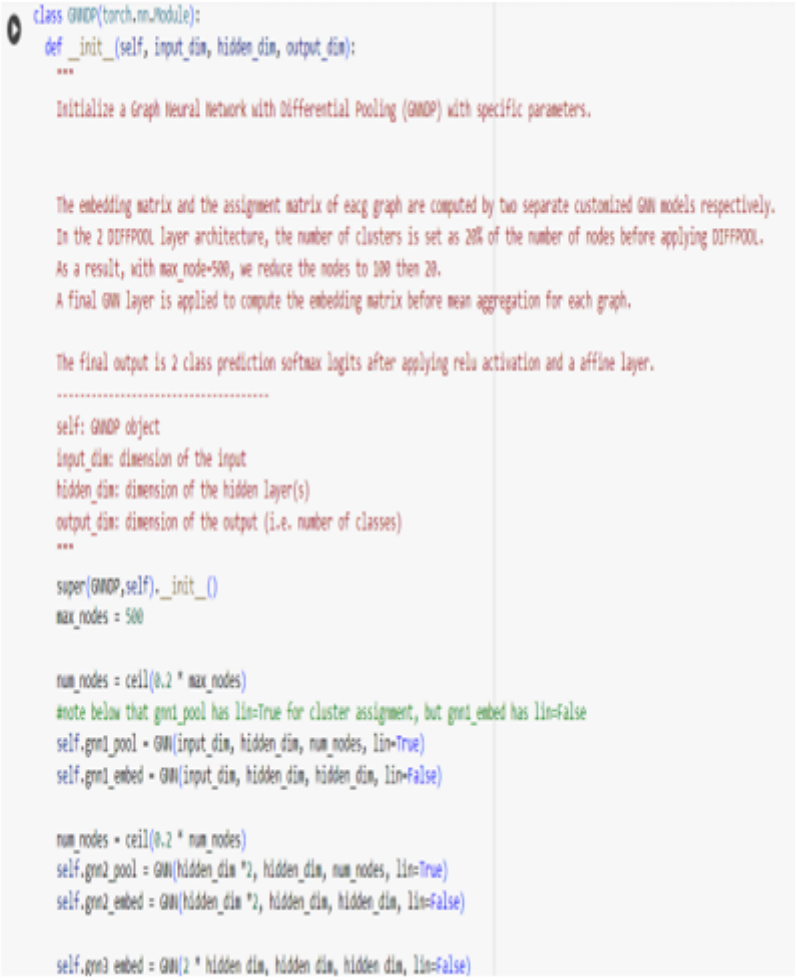
Figure 7

Results Generated by using GCN Model

```
from torch.nn.modules.batchnorm import BatchNorm1d

class GNN(torch.nn.Module):
  def __init__(self, input_dim, hidden_dim, output_dim, normalize = False, lin = False):
    """
    Initialize a customoized GNN with specific parameters.

    By default, this GCN has 2 convolutional layers and 1 linear layer if argument lin is set to be True.
    It also has 1 layer of batch normalization between the two conv-layers.

    The final output is the concatenated sequantial outputs of the DenseSageConv and Batchnorms layers.

    ---------------------------------
    self: GNN object
    input_dim: dimension of the input
    hidden_dim: dimension of the hidden layer(s)
    output_dim: dimension of the output (i.e. number of classes)
    normalize: True or False
    lin: True or False
    """
    super(GNN, self).__init__()

    self.conv1 = DenseSAGEConv(input_dim, hidden_dim, normalize)
    self.bn1= BatchNorm1d(hidden_dim)

    self.conv2 = DenseSAGEConv(hidden_dim, output_dim, normalize)
    self.bn2= BatchNorm1d(output_dim)
```

Figure 8

GNN Model Configuration Code



```
class GNNDP(torch.nn.Module):
  def __init__(self, input_dim, hidden_dim, output_dim):
    """
    Initialize a Graph Neural Network with Differential Pooling (GNNDP) with specific parameters.


    The embedding matrix and the assignment matrix of each graph are computed by two separate customized GNN models respectively.
    In the 2 DIFFPOOL layer architecture, the number of clusters is set as 20% of the number of nodes before applying DIFFPOOL.
    As a result, with max_node=500, we reduce the nodes to 100 then 20.
    A final GNN layer is applied to compute the embedding matrix before mean aggregation for each graph.

    The final output is 2 class prediction softmax logits after applying relu activation and a affine layer.
    ----------------------------------
    self: GNNDP object
    input_dim: dimension of the input
    hidden_dim: dimension of the hidden layer(s)
    output_dim: dimension of the output (i.e. number of classes)
    """
    super(GNNDP,self).__init__()
    max_nodes = 500

    num_nodes = ceil(0.2 * max_nodes)
    #note below that gnn1_pool has lin=True for cluster assignment, but gnn1_embed has lin=False
    self.gnn1_pool = GNN(input_dim, hidden_dim, num_nodes, lin=True)
    self.gnn1_embed = GNN(input_dim, hidden_dim, hidden_dim, lin=False)

    num_nodes = ceil(0.2 * num_nodes)
    self.gnn2_pool = GNN(hidden_dim *2, hidden_dim, num_nodes, lin=True)
    self.gnn2_embed = GNN(hidden_dim *2, hidden_dim, hidden_dim, lin=False)

    self.gnn3_embed = GNN(2 * hidden_dim, hidden_dim, hidden_dim, lin=False)
```

Figure 9

GNN-DP Model Configuration Code

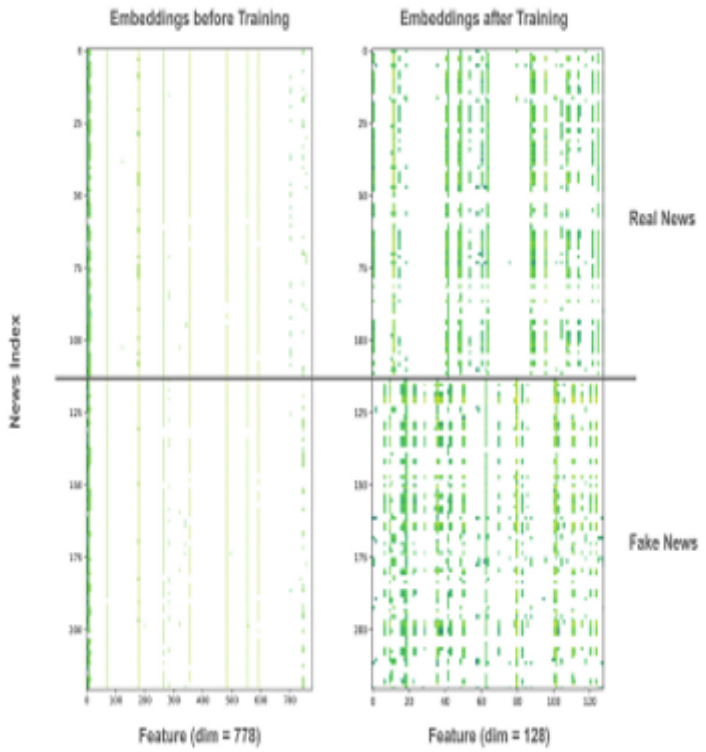


Figure 10

Results Generated by using GNN DP Model

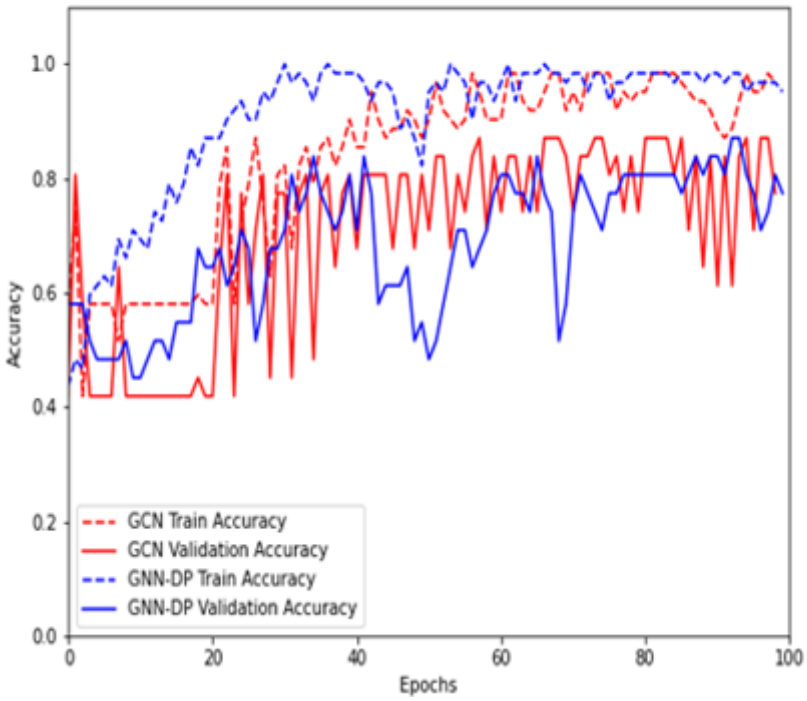


Figure 11

Accuracy for GCN and GNN with based DiffPool Model

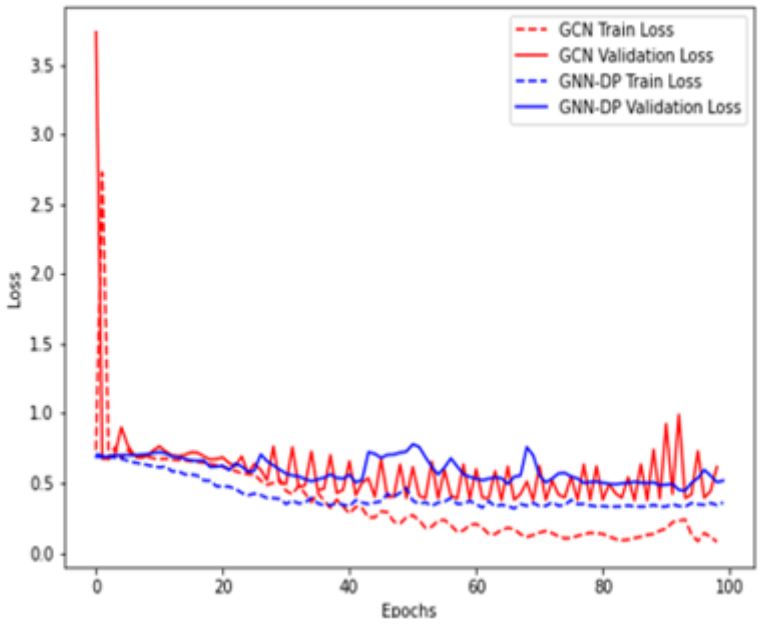


Figure 12

Loss for GCN and GNN with DiffPool Model