DOI: 10.21917/ijsc.2021.0344

GRAPH NEURAL NETWORK LEARNING IN LARGE GRAPHS - A CRITICAL REVIEW

Ashish Gavande and Sushil Kulkarni
Department of Computer Science, University of Mumbai, India

Abstract

Graph Neural Networks have been extensively used to learn non-Euclidian structures like graphs. There have been several attempts to improve the training efficiency and to reduce the learning complexity in modelling of large graph datasets. In this paper we have reviewed the approaches which perform convolutions to model large graphs for classification and prediction. We have critically analysed each of these approaches and veracity of their claims of reduced complexity and have reported their shortcomings. We have further analysed the approaches from graph-dataset perspective.

Keywords:

Graph Neural Networks, Graph Convolutional Networks, Graph Representation Learning, Large Graph Dataset

1. INTRODUCTION

Graph-based associations are widely experienced in everyday life. From gene expression to animal-skeletal motions, from chemical bonding of compounds to stellar planetary compositions of universe, and from social networks to physical networks of the real world; all containing interacting and possibly mutually dependent nodes.

A graph element may influence its associated elements. In technical terms, the features' value of one graph element could be dependent on features' value of another graphical element. Any effort to model graph structure and its elements will require learning of these dependencies. It is essential to learn these dependencies to accurately model any graph and its elements. Further, modelling complex graphs will require inductive learning against time-consuming transductive reasoning.

The traditional deep-learning approaches such as convolutional neural network (CNNs) or recurrent neural networks (RNN) model problem using low-dimensional structures. An image, for example, is convoluted multiple times to extract features which are then used to classify other images. The approaches assume uniformity in associations between neighbouring nodes (pixels in case of images) while convoluting matrix-like data. However, not all problems have uniformly associated nodes, and certainly not graphs. These problems can be more effectively solved using Graph Neural Networks (GNN).

A GNN represents problem domain in terms of a graph where node represent entities and edges represent relationships between the entities. A series of convolutions are performed to extract features of nodes and the degree of influence of their neighbouring nodes on the same. GNNs perform contextual transduction in graph structures to obtain high-level representation of its nodes. Further certain GNNs perform inductive learning for classification or prediction. The differences in the different models of GNNs lie in the approaches they implement to learn and predict these dependencies.

GNNs are primarily classified as Recurrent Graph Neural Networks (RecGNNs) or Convolutional Graph Neural Networks (ConvGNNs). RecGNNs have recurrent learning architectures where neighbouring nodes continuously share information till the network is stabilised. ConvGNNs, on the other hand, involve convolutions over neighbouring nodes of a node for learning latter's representation. ConvGNNs are further classified into spectral-based approaches - based on spectral graph theory and the spatial-based approaches - based on spatial convolution of neighbourhood nodes. As spectral computation required eigen decomposition of adjacency matrices which then affects learning complexity, the later approaches have mostly used spatial based learning. The latter is also extensively used in spatio-temporal GNN models which aim to learn dynamic graphs.

Although GNNs have been well developed to processed unstructured data, the challenge is to efficiently model complex graphs which include large or dense graphs. In fact, the real-world datasets are humungous, sometimes consisting of millions of nodes. There have been some attempts to process such challenging graphs based on approaches of sampling, clustering, etc. and this work is an effort to critical analyse all such implementations.

2. RELATED WORKS

The earliest effort to review GNNs was by [1] which gave overviews of then existing graph convolutional neural networks. [2] work was on studying network embedding models while [3] focused only on attention networks. [4] was the first attempt to consider GNNs in their entirety but only analysed the models from the perspective of relational reasoning and combinatorial generalization. [5] and [6] works presented a comprehensive picture of GNNs with their categorization based on underlying learning principle of their algorithms. Another comprehensive work was produced by [7] which additionally surveyed reinforcement learning and adversarial GNNs.

Although these surveys explained and compared several GNNs, none provided a detailed critical analysis of GNNs dealing with large datasets. Their emphasis rather was on GNNs' categorization to provide hierarchical streamlined view of all GNN models. This paper analyses the working of each model designed to analyse large graph datasets and critically reviews the veracity of the claims made.

2.1 CRITICAL REVIEW

This work not merely discusses working and results of existing models for learning large datasets, but does a thorough critical analysis of the same to learn their effectiveness and shortcomings in learning and reducing its complexity.

2.2 DATASET-WISE ANALYSIS

This work further analysis the datasets that are used in the existing works and their suitability with respect to the complex graphs. We have provided dataset statistics and argued the performances of different approaches with respect to true real-world datasets.

The main sections which follow are as following: Section 2 Defines graph and dataset related concepts, details evolution of GNNs and lists its commonly used notations. Section 3 details the working, algorithm and solutions to issues in previous models of the existing works which are primarily focused to reduce learning complexity. Their advantages, complexity readings and drawbacks are also been stated. Section 3 discusses models which provide improvised technique for learning but not with intent to reduce complexity. The techniques discussed could be used for learning in large graphs. Section 4 discusses the common shortcomings of existing approaches and uses dataset statistics to analyses their effectiveness in learning real-world datasets. Section 5 provides conclusion of the review.

3. DEFINITIONS

- *Graph*: A graph $G = (V, E)$ is consists of vertex set $V = \{v_1, v_2, \ldots\}$, with each element identified as a vertex, and edge set $E = \{e_1, e_2, \ldots\}$, with each element identified as an edge; and where any edge of G links or connects one or two vertices.
- *Receptive Field*: In GNN, the receptive field of a node is the set of nodes that contribute to the determination of its final node representation.
- *Seed-Node*: The root node from which sampling begins and whose representation has to be learned.

3.1 EVOLUTION OF GRAPH NEURAL NETWORKS

The very first work to learn graphs was [8] to study directed acyclic graphs. However, the concept of graph neural network was introduced first in [9] and later extended in [10]. Both of these were recurrent type Graph Neural Networks.

The first ConvGNN type model was [11] and it was a spectral based approach. This was followed by [12] which proved that not only the dimensionality of a graph but also the cost of its Fourier transformation can be reduced by performing simple mean/max pooling at the beginning. [13] later optimised the max/min pooling strategy. This was followed by [14] which created model which performed semi-supervised learning for classification of nodes while [15] used complex spectral filters, the Cayley polynomials, for improvised learning. The most significant foundations of GNNs were laid by [16] which introduced message passing concept in ConvGNNs. Other models were subsequently created which combined convolution with techniques like diffusion, attention, etc., to improve learning.

3.2 NOTATIONS

The commonly used GNN notations are produced in Table.1.

Table.1. GNN notations

Notation	Description
G	A graph
V	The vertex set of G
v, u	Nodes belonging to V
n	The number of nodes, $n = \|V\|$
E	The edge set of G
e	An edge $e \in E$
m	The number of edges, $m = \|E\|$
$N(v)$	The neighbourhood set of v
d	The dimension of a node feature vector.
b	The dimension of a hidden node feature vector.
K	Number of layers in GNN
c	The dimension of an edge feature vector.
k, l	The layer index
t	The time step/iteration index
s	The batch size
r	The number of neighbors sampled for each node
$\sigma(\cdot)$	The sigmoid activation function
$\sigma_h(\cdot)$	The tangent hyperbolic activation function
A	The graph adjacency matrix.
A^T	The transpose of the matrix A.
A^n, $n \in Z$	The n^{th} power of A
D	The degree matrix of A
$X \in R^{n \times d}$	The feature matrix of a graph.
$x \in R^n$	The feature vector of a graph in the case of $d = 1$.
$x_v \in R^d$	The feature vector of the node v.
$X^e \in R^{n \times c}$	The edge feature matrix of a graph.
$x^e_{(v,u)} \in R^c$	The edge feature vector of the edge (v, u).
$X^{(t)} \in R^{n \times d}$	The feature matrix of a graph at time step t
$H \in R^{n \times b}$	The node hidden feature matrix
$h_v \in R^b$	The hidden feature vector of node v
W, Θ, w, θ	Learnable model parameters.

4. INDUCTIVE REPRESENTATION LEARNING ON LARGE GRAPHS (GRAPHSAGE)

The Graph Convolutional Network (GCN) [14] performs full-batch gradient to convolute graphs and which requires all nodes to be present in the memory. As the approach was unscalable to large graphs, GraphSAGE [17] was introduced which performed mini-batch gradient learning to reduce memory requirements and also allowed multiple parameter-updates per epoch leading to faster convergence.

GraphSAGE is an inductive spatial based convolutional GNN that uniformly samples seed-node's neighbourhood nodes for learning. Its works by first performing uniform sampling from the entire node-neighbourhood at each iteration. Then for each of

these minibatch-nodes as seed, the algorithm expands outward in the neighbourhood for a certain pre-determined level of nodes from the seed node. At each expansion it randomly selects certain nodes for learning which are then used for further expansion. After reaching the peripheral nodes, it learns their representation which is then recursively passed back to their sampled parent nodes. The representation is then learnt at these parent nodes and aggregated with the previously learnt representation and again passed higher up in the hierarchy towards the seed node. Its convolution is represented as in Eq.(1)

$$h_v^{(k)} = \sigma(W^{(k)} \cdot f_k(h_v^{(k-1)}, \{h_u^{(k-1)}, \forall u \in S_{N(v)}\})) \quad (1)$$

where $h_v^{(0)} = x_v, f_k(\cdot)$ is an aggregation function, $S_{N(v)}$ is a random sample of the node v's neighbours [6].

The work considers four variations of GraphSAGE, viz., GraphSAGE-GCN, GraphSAGE-mean, GraphSAGE-LSTM, and GraphSAGE-pool; these vary on the aggregator functions used. Here GraphSAGE-GCN is the base version modelled on GCN; GraphSAGE-mean uses element-wise mean of representations; GraphSAGE-LSTM is based on the LSTM architecture [18]; and GraphSAGE-pool where element-wise max-pooling is applied for aggregation.

GraphSAGE variants comparisons was with four baselines, viz., a random classifier, a logistic regression feature-based classifier, the DeepWalk algorithm [19], and model with concatenation of the raw features and DeepWalk embeddings. The comparisons found that GraphSAGE learns larger and denser graphs substantially better than the baselines but is slower when on learning on smaller graphs. The time complexity of GraphSAGE is $O(r^K nd^2)$ and memory complexity is $O(sr^K d + Kd^2)$ [20]. Accuracy-wise, GraphSAGE logged superior micro-averaged F1 scores then the baselines and in some cases the gain was of more than of 50%.

Although GraphSAGE uses nodes' features in learning algorithm to learn topological structures as well as its distribution in the neighbourhood, it can be applied even to graphs without rich node features. Its major contribution is inductive learning which can extend learning to unseen nodes and generalize the learned-node's feature-embeddings to newer or evolving (dynamic) graphs. Further, instead of leaning embedding of each node distinctly, it aggregates the learning over the node's neighbourhood to generate the embeddings. It can be operated in both supervised as well as unsupervised mode.

GraphSAGE performs just random sampling of neighbourhood nodes and does not sample "important" nodes for the seed nodes which affects learning. The prediction algorithm of GraphSAGE and gradient descent function is not unbiased and therefore there is no guarantee of convergence and which therefore mandates a large sample size [21]. And as the sampling size increases the training requires more resources which increases complexity. It also suffers from high time complexity due to recursive (layer-wise) node-embedding processing which increases with increase in depth. Further, except for recursive-learning, neither any graph theoretic approach is used nor any local mutual information is learnt to captured global structural information to improve learning or reduce complexity.

4.1 FAST LEARNING WITH GRAPH CONVOLUTIONAL NETWORK VIA IMPORTANCE SAMPLING (FASTGCN)

FastGCN [22] is graph convolution approach that learns features representations on graph's vertices and interpret convolutions as integral transformation of vertex embedding functions. FastGCN samples using importance-based sampling, unlike in GraphSAGE, a fixed number of vertices and not neighbours for each graph convolutional layer; it samples layer-wise. As the sampling is importance-based, the nodes which influence the seed node majorly, are selected. The model can be represented as in Eq.(2):

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \quad (2)$$

where $\hat{A}$ is normalized graph adjacency matrix, $H(l)$ contains the embedding (row-wise) of the graph vertices in the l^{th} layer, $W(l)$ is a parameter matrix, and σ is nonlinearity [22].

FastGCN is significantly faster compare to GCN [14] and GraphSAGE although is at par with respect to classification accuracy in terms of micro-F1 scores. It also has smaller sampling variance than GraphSAGE. The time complexity of FastGCN is O(Krnd2) and its memory complexity is $O(Ksrd + Kd^2)$ [20].

FastGCN compromises on structural integrity in the sense that the nodes sampled at each layer are independent of nodes sampled in the previous layer. This may lead to filtering of connection information in between layers of graph and affects stability and accuracy of the model. As in GraphSAGE, its convergence is only guaranteed if the sample size goes to infinity [21] which of course will make training very expensive. Further, FastGCN cannot classify entire graphs or sub-graphs.

4.2 ADAPTIVE SAMPLING TOWARDS FAST GRAPH REPRESENTATION LEARNING (ADAPT)

A variant of FastGCN, this general framework is an inductive top-down layer-wise sampling-based convolution framework that approximates optimal sampling by conditionally selecting lower layer nodes based on upper layer nodes [23]. The approach is based on the premises of common neighbourhood for nodes across a layer, i.e., all parent nodes have same sampled neighbours. The effect is of having similar sampling flow for neighbourhood of nodes of a layer which intends to reduce learning complexity.

As Adapt follows a top-down approach, a node in a layer is selected using a self-dependent importance learning function that tells its "importance" for lower layer nodes. The function is computed on the node's features. The process is recursively repeated up to the bottom-most (input) layer. After having sampled the nodes, hidden features of these nodes are learned and passed through activation functions. The sampling is also controlled by keeping an upper limit of sample nodes at each layer. The layer-wise sampling at each layer can be described as in Eq.(3):

$$h^{(l+1)}(v_i) = \sigma_W^{(l)}(N(v_i) E_{q(u|v_1,\ldots,v_n)}[(p(u_j|v_i)/q(u_j|v_1,\ldots,v_n)) h^{(l)}(u_j)]) \quad (3)$$

where $q(u_j|v_1,\ldots,v_n)$ is defined as the probability of sampling u_j given all the nodes of the current layer [23].

The advantage in Adapt is in not re-embedding nodes which may have been already learnt for previous seed-nodes. And where FastGCN ignores structural relationships between nodes of different layers, Adapt's sampling is actually based on lower layer nodes and therefore maintains the relational integrity and captures between-layer correlations. Further, to reduce variance to improve learning as high variance in sampling impedes learning, the sampling is adaptable and can be parameterized, unlike in GraphSAGE and FastGCN. In fact, these both can be represented as node-wise sampler and special layer-wise method variants of Adapt respectively [23].

Adapt also employs attention mechanism [24] for improvised learning. Its variations involve skip-connection technique of message passing across distant nodes and 2-hop neighbourhood sampling which are slightly more accurate than the basic Adapt.

The accuracy scores comparisons reported in [23] indicate that Adapt and its variations are the most accurate than other compared models viz., full-batch gradient GCN, GraphSAGE, FastGCN, graph kernel method KLED [25] and Diffusion Convolutional Network (DCN) [26]. The comparisons were also done with modified GraphSAGE and FastGCN models for fair comparisons. However, the work does not clarify what type of accuracy scores are specified for its own model. It is second only to FastGCN sampling technique in per epoch training for PubMed and Reddit datasets.

4.3 GATED ATTENTION NETWORKS FOR LEARNING ON LARGE AND SPATIOTEMPORAL GRAPHS (GAAN)

GaAN [27] is a gated attention based convolutional GNN, modelled on [24], performs sampling like GraphSAGE, however, with two major differences. At each sampling step it samples minimum of number of neighbouring nodes or, certain maximum number of nodes determined by a hyperparameter. It also merges any repeatedly sample node for a different seed node but of the same mini-batch. The gated attention, which is the main modification to previous models, can modulate the amount of attended content via the introduced gates [27]. The model also involves transforming graph aggregators into Gated Graph Recurrent Unit (GGRU) which can be used for spatial-temporal learning.

GaAN is only slightly more accurate than FastGCN and Adapt. However, there is one inconsistency in the scores reported; one table reports the accuracy as micro-F1 scores while the same are reported as F1 scores in another table.

In traffic speed forecasting analysis done with METR-LA dataset, GaAN performed best on 15- and 30-minutes time horizons in comparison with eight other models. It can model spatio-temporal relationships.

The major drawback is that attention mechanism requires extra computation in terms of pairs of feature vectors, resulting in excessive memory and computational resource requirements [28] which increases complexity.

4.4 STOCHASTIC TRAINING OF GRAPH CONVOLUTIONAL NETWORKS WITH VARIANCE REDUCTION (STOGCN)

StoGCN [21] is a stochastic approximation based convolutional GNN which improvises by employing the historical representation of nodes' activations to reduce variance in the sampled nodes in order to reduce the receptive-field size. Instead of recursively calculating a node's activation representation every time using its neighbours' activations at previous layers, it maintains an approximated representation for each node which is updated at every layer with newly learned representations. The model estimator has a zero variance and is referred to as control variate. A graph convolution layer in StoGCN is represented as in Eq.(4):

$$Z^{(l+1)} = PH^{(l)}W^{(l)},\ H^{(l+1)} = \sigma(\mathrm{Z}^{(l+1)}) \quad (4)$$

where $H(l)$ is the activation matrix in the l^{th} layer, whose each row is the activation of a graph node [21]. Here, $H(0) = X$ is the input feature matrix, and $W(l)$ is a trainable weight matrix.

The theoretical guarantees of the framework allow quality learning even with two nodes sampling [20]. The time complexity of StoGCN is $O(Kmd + Knd^2 + r^K nd^2)$ and its memory complexity is $O(Knd + Kd^2)$ [20].

A variation of model is control variate for dropout (CVD), to work with networks that employ dropout – a node dropping regularization procedure to avoid overfitting. The dropout is performed after neighbour averaging.

A drawback of this model is that the selection requires the all the intermediate embeddings of all the nodes be present in the memory, making learning in large graphs restrictive. The work has reported micro-F1 comparisons only for PPI dataset and accuracy scores for other datasets and no justification is provided for the same. The reported convergence is obtained in far less time and in far less epochs but is reported only for Reddit dataset and not for all the used datasets. Here again the type of accuracy scores is not clarified.

4.5 GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR WEB-SCALE RECOMMENDER SYSTEMS (PINSAGE)

PinSage [29] is GCN based recommender algorithm that performs low-latency random walks on graphs for importance-based neighbourhood sampling of nodes. Here, the importance-based sampling is performed by selecting nodes with highest normalized count visits of multiple random walks. The model applies multiple convolutions in a localized set-up of small neighbourhood nodes to learn embeddings of each node for multiple features. The information gain in each convolution with respect to feature-type is stacked to get more comprehensive embeddings. The algorithm uses max-margin based loss function with an intent to maximize the inner product of embedding of the query item and the corresponding related item and minimize the inner product of the query item and an unrelated item [29].

PinSage is the only model that employs CPU-GPU co-ordination wherein a CPU-bound producer efficiently samples node network neighbourhoods and identifies necessary features for local convolutions which are then used by a GPU-bound TensorFlow model to efficiently run stochastic gradient decent.

Further, a MapReduce process ensures that the latent vector for each node is computed only once.

The performance comparisons done are only with other recommender systems and not with any other GNN-based model. Also, generally used datasets by other approaches are not used to evaluate its performance and so comparisons cannot be analysed.

4.6 LARGE-SCALE LEARNABLE GRAPH CONVOLUTIONAL NETWORK (LGCN)

LGCN [28] is a spatial based GCN that transforms generic graphs into grid-structure to apply standard one-dimensional CNN convolution for feature learning of graph nodes. It learns representations of neighbourhood nodes for the seed node and arranges them in matrix with features forming the columns and rows populated with feature-values for each neighbourhood node. The rows are then sorted based on the feature-values and top few rows and correspondingly nodes are selected for defining representation of the seed node.

LGCN divides an entire graph into sub-graphs to counter computational bottlenecks and memory constraints that limit processing of large graphs. Due to sub-graphing and transformation to lower dimensional structure, deeper learning is also possible in this arrangement.

The layer-wise propagation rule of LGCN is formulated as in Eq.(5) and Eq.(6):

$$X_l = g(X_l, A, k) \tag{5}$$

$$X_{i+1} = c(X_l) \tag{6}$$

where A is the adjacency matrix, $g(\cdot)$ is an operation that performs the k-largest node selection to transform generic graphs to data of grid-like structures, and $c(\cdot)$ denotes a regular 1-D CNN that aggregates neighbouring information and outputs a new feature [28].

The comparison is done only with GraphSAGE on PPI dataset and reported micro-F1 scores indicate better accuracy compare to GraphSAGE.

4.7 CLUSTER-GCN - AN EFFICIENT ALGORITHM FOR TRAINING DEEP AND LARGE GRAPH CONVOLUTIONAL NETWORKS

Cluster-GCN [20] is a spatial based convolutional GNN which uses graph clustering algorithm to sample block of nodes from a graph. Cluster-GCN uses normalized adjacent matrices whose diagonal values are amplified to reflect higher contribution of nearby nodes than distant nodes. The normalization is done to avoid instability due to exponential growth in amplified values with increase in layers. The embeddings of node's neighbours can be described as in Eq.(7) and Eq.(8):

$$\hat{A} = (D + I)^{-1}(A + I) \tag{7}$$

$$X^{(l+1)} = \sigma((\hat{A} + \lambda \mathrm{diag}(\hat{A}))X^{(l)}W^{(l)}) \tag{8}$$

where $\hat{A}$ is the normalized adjacency matrix, $X(l) \in R(n \times dl)$ is the embedding at the l^{th} layer for all the n nodes.

The convolution on nodes is restricted to subgraphs only; the learning does not cross the boundaries of a sub-graph. As only a subgraph node is required in the memory during convolution, memory and time complexities are reduced. This allows to handle larger graphs and also deeper levels learning. Further, sub-graphing improves embedding utilization as the previously learned embeddings can and are re-utilized in densely linked nodes of a subgraph.

Cluster-GCN further incorporates stochastic multiple partitioning to reduce variance between batches of nodes. Otherwise, similar nodes within a batch and different nodes across batches may defer convergence of the model.

Cluster-GCN performs far better than existing approaches in terms of time and memory complexities. The time complexity of Cluster-GCN is $O(Kmd + Knd^2)$ and its memory complexity is $O(Ksd+Kd^2)$ [20]. The model gives results even in graph with 2 million plus nodes and 61 million plus edges.

The accuracy values which are reported are in terms of F1 score and not micro-F1 score. The clustering has to be done prior to processing which makes this arrangement unsuitable for dynamic graphs.

5. GEOMETRIC GRAPH CONVOLUTIONAL NETWORKS

Geometric Graph Convolutional Networks (Geom-GCN) [30] was designed to tackle two fundamental problems in Message-passing neural networks (MPNNs) which are loss of structural information of graphs in "blindly" aggregating messages for neighbourhood nodes and inability to learn long-range dependencies in disassortative graphs. In later, it means that aggregational representation ignores distant but influential nodes which consequently affects learning. Geom-GCN's solution is a geometric aggregation scheme which maps a graph to a continuous latent space via node embedding, and then use the geometric relationships defined in the latent space to build structural neighbourhoods for aggregation [30]. Additionally, a bi-level aggregator operates on the structural neighbourhoods to update the hidden features representations of nodes.

The structural neighbourhood for the next aggregations is described as in Eq.(9) and Eq.(10):

$$N(v) = (\{N_g(v), N_s(v)\}, \tau) \tag{9}$$

$$N_s(v) = \{u | u \in V, d(z_u, z_v) < \rho\} \tag{10}$$

where $N_g(v)$ is actual neighbourhood of v, $N_s(v)$ is latent space neighbourhood, τ is a relational operator on neighbourhoods, and ρ is a pre-given parameter.

The bi-level aggregations are described as in Eq.(11), Eq.(12) and Eq.(13):

$$e_{(i,r)}{}^{v,l+1} = p(\{h^l_u | u \in N_i(v), \tau(z_u, z_v) = r\}) \tag{11}$$

$$m_v{}^{l+1} = q(e_{(i,r)}{}^{v,l+1}, (i,r)) \tag{12}$$

$$h_v{}^{l+1} = \sigma(W_l \cdot m_v{}^{l+1}) \tag{13}$$

where $e_{(i,r)}{}^{v,l+1}$ is the features of virtual nodes of i neighbourhood and r relationship with p aggregating function. Here, Eq.(11) is the low-level aggregation, Eq.(12) is the high-level aggregation and Eq.(13) is the non-linear transformation.

Geom-GCN has three main variants viz., Geom-GCN-I, Geom- GCN-P, and Geom-GCN-S which are differ in the type of embedding they employ. Geom-GCN-I uses Isomap embedding [31]; Geom- GCN-P uses Poincare embedding [32] and Geom-GCN-S uses struc2vec embeddings [33].

The time complexity of Cluster-GCN is $O(nm^2|R|)$ where n is the size of input representations, m is the number of hidden units in non-linear transform, and $2|R|$ is the number of virtual nodes [30]. The run-time comparisons show that GCN is better than Geom-GCN in learning representations.

Geom-GCN variants are compared only with GCN and GAT with better mean classification accuracy in most cases. However, other models such as GraphSAGE, Adapt, StoGCN and Cluster-GCN are not compared. Further, the datasets used have no more than 20,000 nodes with some having only few hundred nodes; it did not used large datasets. And also, micro-f1 accuracy scores are not mentioned.

The main drawback of Geom-GCN is that it is a transductive learning model by virtue of which it cannot extend learning on some nodes to all the nodes; each node has to be learned independently making it unscalable for large graphs.

5.1 COMMUNITY ENHANCED GRAPH CONVOLUTIONAL NETWORKS (CE-GCN)

CE-GCN [34] integrated neighbourhood and community information in learning graph representations. It first does graph convolution and then uses modularity to measure community strength of the convolution network. The two are then combine in the cost function for optimization of the network. CE-GCN uses [14] for graph convolution.

For measuring community strength between two communities, it uses the modularity function as described as in Eq.(14)

$$Q = (1/4e)\sum_{ij}(A_{ij} - (k_i k_j/2e))\, s_i s_j \quad (14)$$

where $s_i = 1$ if node i belongs to the first community, otherwise $s_i = -1$. k_i is the degree of node i and $(k_i k_j/2e)$ is the expected number of edges between nodes i and j if edges are placed at random and where $s=[s_i] \in R_n$ is the community membership indicator [34]. The modularity Q actually measures the difference between the number of edges falling within communities and the expected number in an equivalent random network [34].

The time complexity of proposed algorithm can be considered as $O(|E| \sum_{l=0}^{L} f_1 + |V|\sum_{l=1}^{L} f_1 f_{l-1} f_l)$ which is linear [34]. CE-GCN uses GCN to perform convolution which performs full-batch gradient to convolute graphs and which requires all nodes to be present in the memory. As GCN is unscalable to large graphs CE-GCN itself will be unscalable to large graphs. Further, usage of any other model for convolution may make the time complexity non-linear. It, like in Geom-GCN, does not exhibit learning of large datasets as graphs of less than 20,000 nodes are only used. Except for GraphSAGE, other models such as, Adapt, StoGCN and Cluster-GCN are not used in comparison and also micro-f1 accuracy scores are not mentioned.

6. PERFORMANCE EVALUATION

It is evident in the above analysis the existing algorithms primarily depend on sampling to convolute large graphs with reduce complexity. The main drawback of standard sampling approach could be missing of important nodes which could negatively affect learning. Like-wise, in clustering, with sub-graphing there could loss of structural information which could also affect learning. However, sub-graphing allows deeper learning.

We observed that graph-coarsening techniques which reduces nodes by "combining" multiple neighbour nodes into a super-node is not being used in for learning large dataset. The primarily reason could be that graph coarsening requires eigenvector computations on graph's Laplacian matrix after each coarsening step which increases computation. Further fitting the entire matrix for large graphs makes memory complexity intractable.

The earlier approaches have reported micro-F1 scores for accuracy comparisons. However, StoGCN and Cluster-GCN, which claim superior performances to earlier approaches, have reported micro-F1 scores for one dataset only; for rest the results are reported in terms of F1-scores or mere accuracy scores. There is lack of clarity in the type of scores reported by Adapt and GaAN.

The F1 scores do not give class-wise weighted comparisons [35] and therefore the accuracy claims cannot be fully accepted. Instead, micro-F1 scores should have been reported for correct accuracy conclusions and comparisons. Consequently, the reported reduced memory and time complexities for and on the basis of F1 scores also cannot be fully accepted. A true good algorithm should reduce complexities along with good micro-F1 scores. We tabulate the reported best accuracy scores of all the implementations for different datasets in the Table 2 as reported in original papers. However, some newer works had modified the original implementations and had obtained slightly different scores for the same models. It should be noted that the mentioned accuracy scores are not of same type and contain mixture of "plain" accuracy, F1 and micro-F1 scores and therefore are not directly comparable.

Table.2. Accuracy scores with standard deviation (in percentage)

Algorithm	Cora	Cite-Seer	Pub-Med	PPI	Reddit
GraphSAGE	83.9	-	-	61.2	95.4
FastGCN	85		88	-	93.7
Adapt	87.44 ±0.34	79.66 ±0.18	90.6 ±0.16	-	96.27 ±0.32
GaAN	-	-	-	98.71 ±0.03	96.83 ±0.03
StoGCN	82.0	70.9	78.9	97.8	96.3
LGCN	-	-	-	77.2 ±0.002	-
Cluster-GCN	-	-	-	99.36	96.6

6.1 DATASETS REVIEW

It is observed that different data-statistics have been reported by the existing works for the same datasets. While where FastGCN, LCGN and StoGCN reported same n of 2,708 and 19,717 for Cora and PubMed datasets respectively; StoGCN reported higher m for both the datasets. In case of Reddit dataset, which is used by FastGCN, GaAN, StoGCN and Cluster-GCN, the same n is reported. However, while for the same dataset where FastGCN and Cluster-GCN mentioned the same m value of 11.6 million approximately, StoGCN and GaAN reported it to be 23.4 million and 114.6 million respectively. Incidentally, Cluster-GCN

has mentioned reported accuracy scores of StoGCN and GaAN as it is for accuracy comparison in spite of these algorithms apparently dealing with more complex Reddit dataset. In case of PPI dataset, the *n* and *m* values reported are – StoGCN: 14,755 and 4,58,973; LCGN: 56,944 (only *n* reported); GaAN: 56.9K and 806.2K and Cluster-GCN: 56,944 and 818,716. The data of Table 3 from [36] summarizes attributes of commonly used graphs.

Table.3. Dataset Statistics

Dataset	Cora	CiteSeer	Reddit 5K
Type	Citation	Citation	Social
Nodes	2.7K	3.3K	2.5M
Edges	5.4K	4.5K	11.9M
Graph Density	1.4812E-03	8.51796E-04	3.68E-06
Max. degree	169	99	8K
Min. degree	1	1	4
Avg. degree	4	2	9

The computed graph density for PubMed, PPI, Amazon and Amazon2M datasets for highest reported n and m values are 5.575E-04, 1.699E-07, 7.812E-07 and 1.28E-09 respectively.

It is evident all these datasets have very low average density. It could be then ascertained none of the existing algorithms have learned highly dense graph datasets.

7. CONCLUSION

In this review, we have detailed the approaches to model large graphs. We have critically analysed each of these approaches and their claims of learning and reducing complexity in large graphs. It is observed that Adapt gives the best micro-F1 accuracy for comparatively smaller datasets Cora, CiteSeer and PubMed while GaAN has the best score in case of larger Reddit dataset. It is to be noted that Adapt's Cora and PubMed datasets has lesser edges then the same datasets used by StoGCN; but StoGCN has reported only accuracy scores and not micro-F1 scores. Cluster-GCN has reported the best processing time with best memory and time complexity. However, they have processed only highly sparse datasets and not performed learning in dense graphs.

In the final analysis, it cannot be definitely stated that any of the existing approaches does quality learning along with substantially reduction of complexity in true real-world graphs.

REFERENCES

[1] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean Data", *IEEE Signal Processing Magazine*, Vol. 34, No. 4, pp. 18-42, 2017.

[2] W. L. Hamilton, R. Ying and J. Leskovec, "Representation Learning on Graphs: Methods and Applications", *IEEE Data Engineering Bulletin*, Vol. 24, No. 2, pp. 1-24, 2017.

[3] J.B. Lee, R.A. Rossi, S. Kim, N.K. Ahmed and E. Koh, "Attention Models in Graphs: A Survey", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-13, 2018.

[4] P. W. Battaglia, "Relational Inductive Biases, Deep Learning, and Graph Networks", *Proceedings of International Conference on Artificial Intelligence and Machine Learning*, pp. 1-14, 2018.

[5] J. Zhou and S. Hu, "Graph Neural Networks: A Review of Methods and Applications", *Proceedings of International Conference on Machine Learning*, pp. 1-22, 2018.

[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks", *Proceedings of International Conference on Network Embedding and Graph Neural Networks*, pp. 1-22, 2019.

[7] Z. Zhang, P. Cui and W. Zhu, "Deep Learning on Graphs: A Survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 8, pp. 1-24, 2018.

[8] A. Sperduti and A. Starita, "Supervised Neural Networks for the Classification of Structures", *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, pp. 714-735, 1997.

[9] M. Gori, G. Monfardini and F. Scarselli, "A New Model for Learning in Graph Domains", *Proceedings of International Conference on Neural Networks*, Vol. 2, No. 2, pp. 729-734, 2005.

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model", *IEEE Transactions on Neural Networks*, Vol. 20, No. 1, pp. 61-80, 2009.

[11] J. Bruna, W. Zaremba, A. Szlam and Y. Le Cun, "Spectral Networks and Locally Connected Networks on Graphs", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 1-14, 2013.

[12] M. Henaff, J. Bruna and Y. Le Cun, "Deep Convolutional Networks on Graph-Structured Data", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 1-10, 2015.

[13] M. Defferrard, X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", *Proceedings of International Conference on Advances in Neural Information Processing Systems*, pp. 3844-3852, 2016.

[14] T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *Proceedings of International Conference on Machine Learning*, pp. 1-14, 2016.

[15] R. Levie, F. Monti, X. Bresson and M.M. Bronstein, "CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters", *IEEE Transactions on Signal Processing*, Vol. 67, No. 1, pp. 97-109, 2019.

[16] A. Micheli, "Neural Network for Graphs: A Contextual Constructive Approach", *IEEE Transactions on Neural Networks*, Vol. 20, No. 3, pp. 498-511, 2009.

[17] W.L. Hamilton, R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs", *Proceedings of International Conference on Advances in Neural Information Processing Systems*, pp. 1025-1035, 2017.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computing*, Vol. 9, No. 8, pp. 1735-1780, 1997.

[19] B. Perozzi, R. Al-Rfou and S. Skiena, "Deep Walk: Online Learning of Social Representations", *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 701-710, 2014.

[20] W.L. Chiang, Y. Li, X. Liu, S. Bengio, S. Si and C. J. Hsieh, "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks", *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 257-266, 2019.

[21] J. Chen, J. Zhu and L. Song, "Stochastic Training of Graph Convolutional Networks with Variance Reduction", *Proceedings of International Conference on Machine Learning*, pp. 1503-1532, 2018.

[22] M. T. Chen Jie and X. Cao, "FASTGCN : Fast Learning With GCN vis Importance Sampling", *Proceedings of International Conference on Machine Learning*, pp. 1-15, 2018.

[23] W. Huang, T. Zhang, Y. Rong and J. Huang, "Adaptive Sampling Towards Fast Graph Representation Learning", *Proceedings of International Conference on Advances in Neural Information Processing Systems*, pp. 4558-4567, 2018.

[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph Attention Networks", *Proceedings of International Conference on Artificial Intelligence*, pp. 1-12, 2017.

[25] F. Fouss, K. Francoisse, L. Yen, A. Pirotte and M. Saerens, "An Experimental Investigation of Kernels on Graphs for Collaborative Recommendation and Semisupervised Classification", *Neural Networks*, Vol. 31, pp. 53-72, 2012.

[26] J. Atwood and D. Towsley, "Diffusion-Convolutional Neural Networks", Proceedings of International Conference on Advances in Neural Information Processing Systems, pp. 1-15, 2016.

[27] J. Zhang, X. Shi, J. Xie, H. Ma, I. King and D.Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs", *Proceedings of International Conference on Artificial Intelligence*, pp. 339-349, 2018.

[28] H. Gao, Z. Wang and S. Ji, "Large-Scale Learnable Graph Convolutional Networks", *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 1416-1424, 2018.

[29] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems", *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 974-983, 2018.

[30] H. Pei, B. Wei, K. C.C. Chang, Y. Lei and B. Yang, "Geom-GCN: Geometric Graph Convolutional Networks", *Proceedings of International Conference on Artificial Intelligence and Machine Learning*, pp. 1-12, 2020.

[31] V. De, L.J. Tenenbaum and S. Joshua, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Proceedings of International Conference on Geometric Structure of High-Dimensional Data and Dimensionality Reduction*, pp. 151-180, 2012.

[32] M. Nickel and D. Kiela, "Poincare Embeddings for Learning Hierarchical Representations", *Proceedings of International Conference on Advances in Neural Information Processing Systems*, pp. 6339-6348, 2017.

[33] L.F.R. Ribeiro, P.H.P. Saverese and D.R. Figueiredo, "Struc2vec: Learning Node Representations from Structural Identity", *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 385-394, 2017.

[34] Y. Liu, "Community Enhanced Graph Convolutional Networks", *Pattern Recognitions Letters*, Vol. 138, pp. 462-468, 2020.

[35] B. Shmueli, "Multi-Class Metrics Made Simple, Part II: the F1-Score", Available at https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1, Accessed at 2019.

[36] Ryan Rossi and Nesreen Ahmed, "Network Statistics", Available at http://networkrepository.com/.