NENN: Incorporate Node and Edge Features in Graph Neural Networks

Yulei Yang and Dongsheng Li

May 25, 2020

NENN: Incorporate Node and Edge Features in Graph Neural Networks

ABSTRACT

Graph neural networks (GNNs) have attracted an increasing attention in recent years. However, most existing state-of-the-art graph learning methods only focus on node features and largely ignore the edge features that contain rich information about graphs. In this paper, we propose a novel model to incorporate **N**ode and **E**dge features in graph **N**eural **N**etworks (NENN) based on a hierarchical dual-level attention mechanism. Specifically, the node-level attention layer and edge-level attention layer are alternately stacked to learn the importance of the node based neighbors and edge based neighbors for each node and edge. Leveraging the proposed NENN, the node and edge embeddings can be mutually reinforced. Extensive experiments on academic citation and molecular networks have verified the effectiveness of our proposed graph embedding model.

CCS CONCEPTS

• **Information systems** → **Data mining**; *Data mining*; • **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

Graph Neural Network, Attention Mechanism, Edge Features

ACM Reference Format:
. 2020. NENN: Incorporate Node and Edge Features in Graph Neural Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have become very useful and successful techniques to process various data with regular grid-like structure [10, 11, 17, 22, 25]. However, the non-Euclidean graphs containing all kinds of nodes and edges are ubiquitous in the real world, such as social networks, bioprotein networks and citation networks, which can not be easily represented due to the complex and irregular structure.

In recent years, there is a growing interest in graph presentation learning methods. Inspired by spectral graph theory, [2] generalizes CNNs to the graph domain based on the feature decomposition of graph Laplace matrix. In order to reduce the overhead of the decomposition, ChebNet [5] is proposed to approximate convolution kernel with Chebyshev polynomials. As a pilot work, [16] proposes a graph convolutional network (GCN) as the first order approximation of ChebNet, which greatly simplifies the convolution filters by limiting the receptive field to the 1-hop neighbors for each node. Finally, the GCN model is successfully applied to semi-supervised node classification and achieves state-of-the-art performance. The basic idea behind GCN is to map a high-dimensional node representation to a low-dimensional vector by transforming, propagating, aggregating and updating node features across edges in a graph. Nevertheless, GCN model is essentially a spectral approach working on transductive learning tasks. As a result, GCN can not run on large and dynamic graphs effectively. To address the limitations of GCN, GraphSAGE [9] extends GCN from a transductive approach to an inductive one using a spatial-based method to train node embeddings. GraphSAGE restricts neighborhood sampling to learn how to aggregate node features rather than train fixed node embeddings. Graph Attention Network (GAT) [26], a newfangled attention-based graph neural network, trains weight coefficients associated with neighbors for each node to learn node embeddings. It has demonstrated the effectiveness in graph embedding and shown the superiority over the previous methods.

Despite the success of existing graph neural networks, there are two enormous challenges. On the one hand, almost all previous literatures only leverage the node features and completely ignore the edge features that are completely likely to contain important information. For example, in molecule networks, a node represents an atom while an edge represents a bond connecting two atoms. A bond usually has some simple edge features (e.g., bond type, atom pair type, bond order, conjugated, ring status, aromaticity), which are closely related to atom features. On the other hand, how to measure the importance of neighborhood as well as the connecting edges or nodes is not fully considered.

In order to address the aforementioned challenges, we propose a novel graph neural network, named NENN, which incorporates node and edge features based on a dual-level attention mechanism, including node-level and edge-level attentions. Specifically, we aim to to learn the importance of node based neighbors and edge based neighbors and aggregate embeddings for each node in the node-level attention layer. On the contrary, the embedding of each edge is generated in the edge-level attention layer.

We conduct extensive experiments on node classification, graph classification and graph regression to verify the effectiveness of our proposed NENN. For node classification, we use the benchmark citation network datasets: Cora, Citeseer [23] and Pubmed [6]. For graph classification and graph regression, we demonstrate the proposed NENN is able to effectively generatere node and edge embeddings by incorporating node and edge features on multiple molecular datasets: Tox21 [27], HIV [27], Freesolv [19], and Lipophilicity [27]. The results show that the proposed NENN outperforms relevant baselines by a significant margin.

In a nutshell, our main contributions are summarized as follows:

- We propose a novel graph neural network (NENN), which incorporates both node and edge features simultaneously to learn node and edge embeddings, where the identity of node and edge are alternated in each layer.
- To the best of our knowledge, this is the first attempt to take the influences of neighbors for each node and edge into

Conference'17, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

consideration based on a hierarchical dual-level attention mechanism , including node-level and edge-level attentions.

- Various graph-related tasks, including graph classification, graph regression, and node classification, are used to verify the scalability and generality for NENN.
- We conduct extensive experiments on benchmark data sets.The citation networks [6, 23] and multiple molecular networks [19, 27] are applied to demonstrate the effectiveness of our model. The results show the superiority of the proposed NENN compared with the state-of-the-art baselines.

2 RELATED WORK

The real-world data usually appears in the form of non-Euclidea graphs. On of the most severe challenges in graph representations is to efficiently exploit the node and topology information. At present, graph representation learning methods can be roughly divided into three parts: matrix factorization, random walks and graph neural networks.

Factorization-based approaches. The key idea of matrix factorization is that the relation matrix (e.g. adjacency matrix and Laplace matrix) is decomposed to yield the low-dimentional representations. For example, Grarep [3] reduces the dimension of the relation matrix by SVD decomposition to get the k-step network vertex representation for weighted graphs. HOPE [20] preserves high-order proximities and captures the asymmetric transitivity for directed graphs. However, these methods can not efficiently process large-scale graphs since they have huge performance overheads for matrix factorization.

Random walk. Just as its name implies, a random walk on a graph starts with a node and recursively connects with a randomly selected neighbor until a threshold. DeepWalk [21] is the first attempt to learn latent representations leveraging truncated random walks. node2vec [8] further designs a biased random walk to efficiently explore diverse neighborhoods based on DFS and BFS strategies. By recursively compressing the input graph to smaller but structurally similar graphs, HARP [4] captures the global topological information about the input graph and generates presentations on this smaller graph during a random walk. Nevertheless, random walk methods are not the most successful methods so far. The shallow embedding methods use unshareable parameters and functions, which makes it impossible to embed nodes of a large-scale graphs to a low-dimensional space. Besides, they are only applied to learn embeddings for fixed graphs in the transductive setting, consequently, do not naturally generalize to unseen nodes.

Graph neural networks. In order to address the limitations of the previous methods, in recent years, graph neural networks are proposed to learn node embeddings. Graph neural network has many branches so far, but we mainly focus on the methods based on graph convolution. Inspired by the successes of CNNs in image recognition, GCN [16] stacks graph convolutional layers to aggregate local information from neighbors and encodes nodes into vectors. GraphSAGE [9] is a novel inductive approach that generates latent embeddings for unseen nodes. Attention mechanisms have been widely applied to many tasks in deep learning. GAT [26] is introduced to learn the importance coefficients for nodes and its neighbors rather than treats the neighborhood information equally. However, the above graph neural networks not only ignore the essential edge features but also fail to differentiate the influences of their connecting edges.

Edge-related Work. To address the above issues, some models are proposed. Message passing neural network (MPNN) [13], a generalized model, including two phases: multiple message passing phases and readout phase, is proposed to predict molecular properties. In MPNN, it is inefficient to learn edge embeddings according to the similar method of learning node embedding in message passing phases. Also, the types of data sets and experiments are too scarce to verify the scalability of MPNN. EGNN [7] introduces attention mechanism to explore edge features, but in the later layer, the edge feature vectors are converted to attention coefficients, which leads to the loss of edge information. NENN adopts a hierarchical dual-level attention mechanism, where the the role of edge and node are alternated, keeps the feature of an edge as a vector rather than an attention coefficient. CensNet [12] embeds both nodes and edges to a latent feature space by using line graph of the original undirected graph. However, the CensNet uses approximated spectral graph convolution in the layer-wise propagation, which makes the CensNet can not process large graphs and directed graphs. On the contrary, the basic idea behind of the proposed NENN is based on spatial domain, which enables various graphs to be processed.

3 THE PROPOSED METHOD: NENN

3.1 NENN Architecture Overview

In this section, we propose a novel model to incorporate **N**ode and **E**dge features in graph **N**eural **N**etworks (NENN) based on a hierarchical dual-level attention mechanism. For a graph $G(V, E)$ with node features and edge features, where V defines a set of $N_v = |V|$ nodes, E is a set of $N_e = |E|$ edges. Let $\mathbf{X} = \{\mathbf{x}_i | i \in N_v\} \in \mathbb{R}^{N_v \times d_v}$ be node feature matrix, where $\mathbf{x}_i \in \mathbb{R}^{d_v}$ represents d_v-dimensional feature vector of node i. Let $\mathbf{E} = \{\mathbf{e}_i | i \in N_e\} \in \mathbb{R}^{N_e \times d_e}$ be edge feature matrix, where $\mathbf{e}_i \in \mathbb{R}^{d_e}$ denotes d_e-dimensional feature vector of edge i.

Definition 1 (Node Based Neighbors). *Given a graph $G(V, E)$, the node based neighbors $\mathcal{N}_i$ of a node i or an edge i are defined as the set of nodes which connect with node i or edge i. Specially, the node based neighbors of node i include itself.*

Example 1. *As shown in Figure 1, nodes are represented by circles while edges are represented by squares. In node-level attention layer, the node based neighbors $\mathcal{N}_1$ of the red node whose feature vector is $\mathbf{x}_1$ denote the neighboring node set $\{2, 3, 5, 6, 7\}$.*

Definition 2 (Edge Based Neighbors). *Given a graph $G(V, E)$, the edge based neighbors $\mathcal{E}_i$ of a node i or an edge i consist of the edges connecting with node i or edge i. Similarly, the edge based neighbors of edge i include itself.*

Example 2. *As shown in Figure 1, the edge based neighbors of the brown edge (i.e., $\mathbf{e}_6$) denote the neighboring edge set $\{1, 2, 5, 7\}$.*

Figure 1 shows the overall process of the proposed NENN for node embeddings generation. The proposed NENN consists of two types of attention layers, node-level attention layer and edge-level

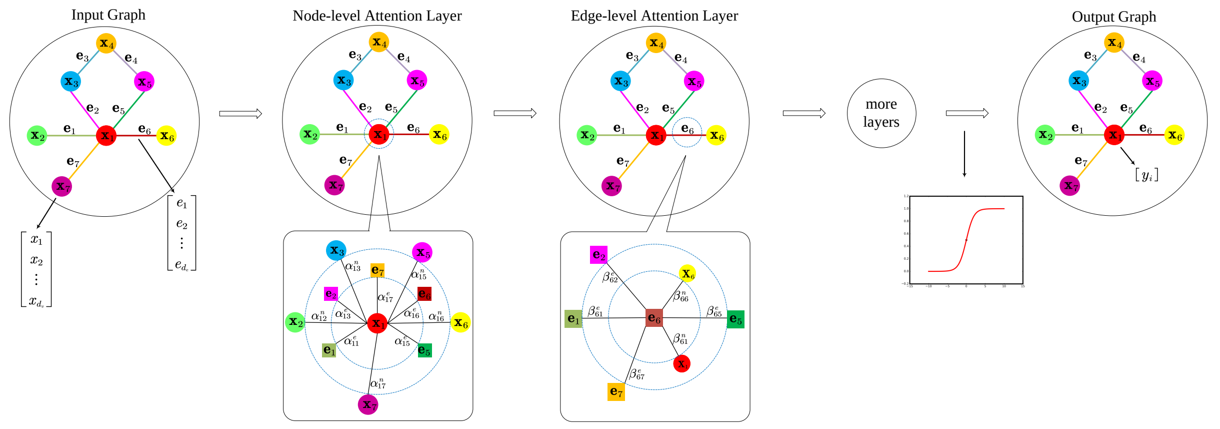


Figure 1: The overall framework of the proposed NENN for node embedding generation.

attention layer. The dual-level attention layers are alternately stacked to learn node and edge embeddings. In the node-level attention layer, we aim to learn the node based neighbors importance α^n_{ij} and edge based neighbors importance α^e_{ij} for each node. In the node-level attention layer, we aim to learn the node based neighbors importance β^n_{ij} and edge based neighbors importance β^e_{ij} for each edge. With the learned importance coefficients, we can aggregate and update node and edge embeddings in order.

3.2 Node-level Attention Layer

In the node-level attention layer, we aim to learn node embeddings with the help of edge features that contain significant information. It is clear to observe that different neighbors of each node play a different role and show different importance in generating node embedding. For this reason, we introduce a node-level attention to learn the importance coefficients of node based neighbors and edge based neighbors for node i.

In the l-th layer, suppose the input features consist of node features, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_v}\}$, $\mathbf{x}_i \in \mathbb{R}^{d_v^{(l)}}$, and edge features, $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{N_e}\}$, $\mathbf{e}_i \in \mathbb{R}^{d_e^{(l)}}$. The importance of node j or edge k ($j \in \mathcal{N}_i, k \in \mathcal{E}_i$) to node i can be reformulated as follows:

$$\begin{aligned} \mathbf{e}^n_{ij} &= Att^n_{node}(W_n\mathbf{x}_i, W_n\mathbf{x}_j) \\ \mathbf{e}^e_{ik} &= Att^e_{node}(W_n\mathbf{x}_i, W_e\mathbf{e}_k) \end{aligned} \tag{1}$$

Here, Att^n_{node} and Att^e_{node} denote the deep neural networks, which perform node-level attention for node i. $W_n \in \mathbb{R}^{d_v^l \times d_v^{l+1}}$ and $W_e \in \mathbb{R}^{d_e^l \times d_e^{l+1}}$ are the learnable weight matrices that linearly transform the input features into high-level features. The importance coefficient $\mathbf{e}^n_{ij}$ means how important node j is to node i, while $\mathbf{e}^e_{ij}$ represents the influence of edge j to node i.

Then the structure information is integrated into the proposed NENN via masked attention, which means the embedding of node i depends only on neighboring nodes j or edges k. Next, the importance coefficient of node j to node i is normalized via the softmax function:

$$\alpha^n_{ij} = softmax_j(\mathbf{e}^n_{ij}) = \frac{\exp\left(\sigma\left(a_n^T\left[W_n\mathbf{x}_i || W_n\mathbf{x}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\sigma\left(a_n^T\left[W_n\mathbf{x}_i || W_n\mathbf{x}_k\right]\right)\right)} \tag{2}$$

where $||$ is the concatenation operation, $a_n \in \mathbb{R}^{2d_v^{l+1}}$ is the parameter vector of a single-layer feed-forward network and σ denotes the activation function (e.g. LeakyReLU).

Then, the importance coefficient α^e_{ik} of node i and edge k can be derived as :

$$\alpha^e_{ik} = softmax_k(\mathbf{e}^e_{ik}) = \frac{\exp\left(\sigma\left(a_e^T\left[W_n\mathbf{x}_i || W_e\mathbf{e}_k\right]\right)\right)}{\sum_{j\in\mathcal{E}_i}\exp\left(\sigma\left(a_e^T\left[W_n\mathbf{x}_i || W_e\mathbf{e}_j\right]\right)\right)} \tag{3}$$

where $a_e \in \mathbb{R}^{d_v^{l+1}+d_e^{l+1}}$ is the parameter vector of a single-layer feed-forward network.

After learning the importance α^n_{ij} and α^e_{ik}, the embedding $\mathbf{x}_{\mathcal{N}_i}$ of node i's node based neighbors can be aggregated with the corresponding importance coefficients:

$$\mathbf{x}_{\mathcal{N}_i} = \sigma\Big(W_n \cdot \mathrm{MEAN}(\{\alpha^n_{ij}\mathbf{x}_j, \forall_j \in \mathcal{N}_i\})\Big) \tag{4}$$

where MEAN is a mean aggregator. Then, the embedding $\mathbf{x}_{\mathcal{E}_i}$ of node i's edge based neighbors can be aggregated as follows:

$$\mathbf{x}_{\mathcal{E}_i} = \sigma\Big(W_e \cdot \mathrm{MEAN}(\{\alpha^e_{ik}\mathbf{e}_k, \forall_k \in \mathcal{E}_i\})\Big) \tag{5}$$

Finally, with the edge based neighbors' embedding $\mathbf{x}_{\mathcal{E}_i}$ and node based neighbors' embedding $\mathbf{x}_{\mathcal{N}_i}$, the embedding of node i in the $(l+1)$-th layer can be combined:

$$\mathbf{x}_i^{(l+1)} = \mathrm{CONCAT}(\mathbf{x}^{(l)}_{\mathcal{N}_i}, \mathbf{x}^{(l)}_{\mathcal{E}_i}) \tag{6}$$

where CONCAT represents concatenation operation. $\mathbf{x}_i^{(l+1)}$ is the returned embedding for node i in the (l)-th layer.

3.3 Edge-level Attention Layer

We use edge features to enhance the node embeddings in node-level attention layer while the node features are fused to learn edge embeddings in edge-level attention layer. To update the edge embeddings, we first learn the importance of node based neighbors and edge based neighbors for each edge.

The importance coefficient β^e_{ij} of edge k ($k \in \mathcal{E}_i$) to edge i is normalized via the softmax function:

$$\beta^e_{ik} = \frac{\exp\Big(\sigma\Big(q_e^T\Big[W_e\mathbf{e}_i || W_e\mathbf{e}_k\Big]\Big)\Big)}{\sum_{j\in\mathcal{E}_i}\exp\Big(\sigma\Big(q_e^T\Big[W_e\mathbf{e}_i || W_e\mathbf{e}_j\Big]\Big)\Big)} \tag{7}$$

where $q_e \in \mathbb{R}^{2d_e^{l+1}}$ is an attention vector. The importance coefficient β^n_{ij} of node j ($j \in \mathcal{N}_i$) to edge i is normalized via the softmax function:

$$\beta^n_{ij} = \frac{\exp\Big(\sigma\Big(q_n^T\Big[W_e\mathbf{e}_i || W_n\mathbf{x}_j\Big]\Big)\Big)}{\sum_{k\in\mathcal{N}_i}\exp\Big(\sigma\Big(q_n^T\Big[W_e\mathbf{e}_i || W_n\mathbf{x}_k\Big]\Big)\Big)} \tag{8}$$

where $q_n \in \mathbb{R}^{d_v^{l+1}+d_e^{l+1}}$ is an attention vector.

Then, edge i's middle node based neighbors embedding $\mathbf{e}_{\mathcal{E}_i}$ and edge based neighbors $\mathbf{e}_{\mathcal{N}_i}$ can be generated by a mean aggregator:

$$\begin{aligned}\mathbf{e}_{\mathcal{E}_i} &= \sigma\Big(W_e \cdot \mathrm{MEAN}(\{\beta^e_{ik}\mathbf{e}_k, \forall_k \in \mathcal{E}_i\})\Big)\\ \mathbf{e}_{\mathcal{N}_i} &= \sigma\Big(W_n \cdot \mathrm{MEAN}(\{\beta^n_{ij}\mathbf{x}_j, \forall_j \in \mathcal{N}_i\})\Big)\end{aligned} \tag{9}$$

Similarly, the embedding of edge i in the $(l+1)-th$ layer can be derived as follows:

$$\mathbf{e}_i^{(l+1)} = \mathrm{CONCAT}(\mathbf{e}^{(l)}_{\mathcal{N}_i}, \mathbf{e}^{(l)}_{\mathcal{E}_i}) \tag{10}$$

With the learned importance, the proposed NENN can pay more attention to some meaningful nodes or edges for the specific task. Note that the importance coefficients are asymmetric which means the influence between different roles in a graph can be quite different (i.g., $\alpha^n_{ij} \neq \alpha^n_{ji}$, $\beta^e_{ij} \neq \beta^e_{ij}$).

The computation of attention can be easily parallelized across all edges and nodes, which means the hierarchical dual-level attention of the proposed NENN is highly efficient. The overall time complexity is linear to the number of nodes and edges.

To process large-scale graphs in the real world, we construct some subgraphs $G' \in G$ according to [9]. The mini-batch training process of the proposed NENN for node embedding generation is shown in Algorithm 1.

3.4 Variants of NENN

In order to verify the validity of the proposed NENN in more detail, we implement our model with different settings according to how to aggregate and update the node and edge features. Specifically, there are four different variants: NENN-NCEC, NENN-NCEA, NENN-NAEC and NENN-NAEA (abbreviated as NENN). Specifically, N, E, C, A represent nodes, edges, convolution operation and attention mechanism, respectively. For example, NENN-NCEA represents node features aggregated by convolution operation and edge features aggregated by attention mechanism.

Algorithm 1: Mini-batch NENN node embeddings generation algorithm

```
Input: Subgraph $G'(V', E')$;
       node features $\{\mathbf{x}_i, \forall_i \in V'\}$;
       edge features $\{\mathbf{e}_i, \forall_i \in E'\}$;
       network depth $L$;
Output: final node embeddings $\{\mathbf{x}_i^{(L)}, \forall_i \in V'\}$;
$\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}_i, \forall i \in V'^{(0)}$;
for $l = 0 \cdots L$ do
    find the node based neighbors $\mathcal{N}_i$ and edge based neighbors $\mathcal{E}_i$;
    if layer $l$ is a node-level attention layer or $l = L$ then
        for each node $i \in V'^{(l)}$ do
            Calculate the importance coefficient $\alpha^{n\ (l)}_{ij}$ and $\alpha^{e\ (l)}_{ik}$;
            Calculate the embedding of node based neighbors $\mathbf{x}^{(l)}_{\mathcal{N}_i}$ and edge based neighbors $\mathbf{x}^{(l)}_{\mathcal{E}_i}$;
            $\mathbf{x}_i^{(l+1)} \leftarrow \mathrm{CONCAT}(\mathbf{x}^{(l)}_{\mathcal{N}_i}, \mathbf{x}^{(l)}_{\mathcal{E}_i})$;
        endfor
    end
    if layer $l$ is an edge-level attention layer then
        for each edge $i \in E'^{(l)}$ do
            Calculate the importance coefficient $\beta^{n\ (l)}_{ij}$ and $\beta^{e\ (l)}_{ik}$;
            Calculate the embedding of node based neighbors $\mathbf{e}^{(l)}_{\mathcal{N}_i}$ and edge based neighbors $\mathbf{e}^{(l)}_{\mathcal{E}_i}$;
            $\mathbf{e}_i^{(l+1)} \leftarrow \mathrm{CONCAT}(\mathbf{e}^{(l)}_{\mathcal{N}_i}, \mathbf{e}^{(l)}_{\mathcal{E}_i})$;
        endfor
    end
endfor
$\mathbf{x}_i^{(L)} \leftarrow \mathrm{CONCAT}(\mathbf{x}^{(L-1)}_{\mathcal{N}_i}, \mathbf{x}^{(L-1)}_{\mathcal{E}_i})$;
```

4 EXPERIMENTS

We evaluate the proposed NENN on three benchmark tasks: (i) semi-supervised node classification on citation networks Cora, Citeseer and Pubmed; (ii) multi-task graph classification on multiple molecular datasets Tox21 and HIV; (iii) graph regression on molecular datasets Lipophilicity and Freesolv.

4.1 Benchmark datasets

We conduct extensive experiments on citation networks and molecular networks to demonstrate the effectiveness of the proposed NENN. Generally, each citation network corresponds to a graph where nodes represent documents and edges represent citation relationships between documents. Different from citation networks, each kind of molecular dataset consists of multiple graphs. Specifically, compounds, atoms, bonds represent graphs, nodes and edges, respectively. More detailed dataset statistics are shown in Table 1 and Table 2.

	Cora	Citeseer	Pubmed
# Nodes	2708	3327	19717
# Edges	5429	4732	44338
# Node Features	1433	3703	500
# Edge Features	2	2	2
# Node Classes	7	6	3

Table 1: Dataset statistics of citation networks for semi-supervised node classification.

	Tox21	HIV	Lipophilicity	Freesolv
# Graphs	7,831	41127	4,200	642
# Node Features	25	25	25	25
# Edge Features	55	80	34	21
# Graph Classes	12	3	-	-

Table 2: Dataset statistics of molecular networks for graph classification and regression.

Cora, Citeseer, and Pubmed. Cora, Citeseer [23] and Pubmed [6], are citation networks and widely used as benchmark datasets for semi-supervised node classification in GCN, GraphSAGE, GAT, EGNN, CensNet, etc.

Tox21 and HIV. Tox21 [27] is a public dataset of toxicity measurements, which comprises 7831 compounds on 12 different quantitative toxicity measurements including AR, AhR, AR-LBD, etc. The HIV [27] dataset originates from the Drug Therapeutics Program AIDS Antiviral Screen, which measures the ability of HIV replication for 41127 compounds. The two datasets are usded to activity prediction (i.e. binary graph classification) that labels compounds as either "active" or "inactive".

Lipophilicity and Freesolv. Lipophilicity [27] is a public dataset usded to measure the affets both membrane permeability and solubility, which provides experimental results of octanol/water distribution coefficient (logD at pH 7.4) of 4,200 compounds. Also, the Free Solvation Database (Freesolv) [19] is a common dataset providing experimental and calculated results of hydration free energies for 642 small molecules in water. According to the characteristics of the two datasets, we conduct extensive graph regression experiments to predict solvation energies or solubility.

4.2 Experimental Setup Parameter Settings

Our experiments are all implemented by TensorFlow [1] and run on Ubuntu Linux 16.04 with NVIDIA RTX 2080 Ti. We use the Adam algorithm [15] for training the models with the learning rate 0.0001 and batch size 128, number of epochs 200. The window size of an early stopping strategy is 200. We implement three layers NENN (i.e. node-level attention layer-edge-level attention layer - node-level attention layer). For all baselines, we split exactly the same training set, validation set and test set to ensure fairness. As shown in Table 5, we follow the same splitting strategy in [12] and conduct experiments on citation networks with different label rate. For all molecular networks, training, validation and test dataset are split into with a ratio of 8:1:1. We run our models 5 times and report mean performance for each experiment.

4.3 Semi-supervised Node Classification

For semi-supervised node classification, we evaluate and report classification accuracies of the proposed NENN. We compared the proposed NENN with the representative models of the following five methods: GCN, GraphSAGE, GAT, CensNet, EGNN.

Table 3 reports classification accuracies of five baselines and the proposed NENN on citation networks. In all cases, the proposed NENN performs consistently much better than all baselins in 7 out of 9 experiments. From t-SNE [18] visualization in Figure 2, we can find that the proposed NENN can achieve more separated clusters than GCN, MPNN, EGNN, CensNet. It demonstrates that via incorporating node and edge features, the proposed NENN can learn a more meaningful node embedding.

4.4 Graph Classification

For graph classification, we predict molecular activity on the Tox21 and HIV datasets. We report the Area Under Curve (AUC) scores and compare with some state-of-art baselines, including Random Forest (RF), Weave [14], GCN, CensNet, EGNN.

We also compare some variants of our model NENN to validate the effectiveness. Table 4 reports the performance of all baselines and our models on four molecular networks. It is clear to observe that five baselines have rooms to improve on Tox21 and HIV datasets compared with our NENN models. Remarkably, NENN improves upon GCN by a margin of 15.0% for the validation and 7.0% for the test on HIV dataset.

4.5 Graph Regression

For graph regression, we predict the solvation energies or solubility on Lipophilicity and Freesolv datasets. Root mean square error (RMSE) are adopted as the evaluation metric. The baselines of graph regression are the same as graph classification.

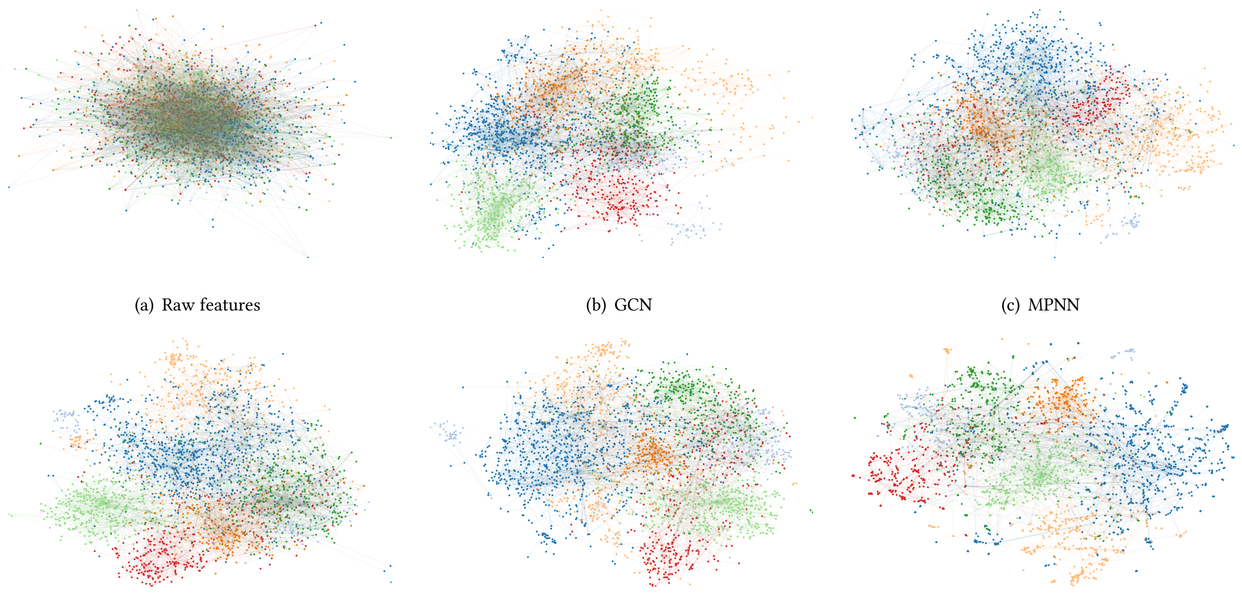
Table 4 shows the experimental results of RMSE. We highlight the best performance for all molecular networks. It's obvious that all of the variants of NENN significantly outperform the compared methods in RMSE. Figure 3(a) shows that the NENN obtains the best AUC score in validation set for HIV networks after around 50 epochs compared with other baselines. Figure 3(b) shows that the NENN greatly reduces RMSE. The remarkable results imply that incorporateing node and edge features in the molecule can improve the quality of node embedding.

5 CONCLUSION

In this paper, we introduce an efficient embedding architecture, named NENN, which incorporates node and features to enhance the node and edge embeddings across neural network layer. The proposed NENN leverages node-level and edge-level attention to learn the importance of node based neighbors and edge based neighbors. Extensive experiments on semi-supervised node classification, graph classification and graph regression demonstrate the effectiveness of NENN.

Dataset	Cora			Citeseer			Pubmed		
Label rate	0.5%	1%	3%	0.3%	0.5%	1%	0.03%	0.05%	0.1%
GCN	52.9 ± 7.4	61.0 ± 7.2	74.0 ± 2.8	39.2 ± 6.3	47.7 ± 4.4	58.3 ± 4.0	57.9 ± 8.1	64.6 ± 7.5	73.0 ± 5.5
GraphSAGE	37.5 ± 5.4	49.0 ± 5.8	64.2 ± 4.0	25.7 ± 6.1	33.8 ± 7.0	51.0 ± 5.7	45.4 ± 5.5	53.0 ± 9.7	59.6 ± 9.5
GAT	41.4 ± 6.9	48.6 ± 8.0	56.8 ± 7.9	30.9 ± 6.9	38.2 ± 7.1	46.5 ± 9.3	50.9 ± 8.8	50.4 ± 8.0	59.6 ± 9.5
MPNN	47.2 ± 3.4	49.1 ± 6.0	55.8 ± 3.7	32.4 ± 7.3	39.1 ± 6.6	48.6 ± 8.2	49.9 ± 6.7	51.2 ± 4.0	60.4 ± 7.8
CensNet	57.7 ± 3.9	67.1 ± 1.3	79.4 ± 1.0	49.4 ± 3.6	57.6 ± 3.0	62.5 ± 1.5	61.4 ± 2.8	65.7 ± 1.2	69.9 ± 2.1
EGNN	59.2 ± 5.8	**67.4 ± 6.9**	78.4 ± 5.4	**49.6 ± 7.6**	55.8 ± 6.3	65.8 ± 6.2	60.2 ± 6.2	62.5 ± 5.9	72.4 ± 7.8
NENN-NAEA	**61.4 ± 7.6**	67.0 ± 6.4	**80.0 ± 6.8**	49.4 ± 8.6	**59.6 ± 7.8**	**66.8 ± 8.9**	**63.2 ± 6.7**	**67.3 ± 7.9**	**75.8 ± 8.9**
NENN-NCEC	58.3 ± 6.3	66.0 ± 3.5	79.0 ± 4.8	47.3 ± 8.5	57.4 ± 6.7	64.6 ± 7.8	62.2 ± 6.4	65.2 ± 8.4	73.8 ± 5.9
NENN-NCEA	59.2 ± 7.3	67.2 ± 8.6	79.7 ± 7.9	48.4 ± 8.7	57.9 ± 5.8	63.5 ± 6.7	63.4 ± 7.5	66.1 ± 4.6	74.0 ± 5.7
NENN-NAEC	59.4 ± 5.7	65.0 ± 4.5	79.9 ± 7.9	48.7 ± 6.7	58.4 ± 7.5	65.4 ± 5.5	63.1 ± 3.4	65.4 ± 7.9	72.1 ± 6.5

Table 3: Classification accuracies on citation networks.


(a) Raw features

(b) GCN

(c) MPNN

(d) EGNN

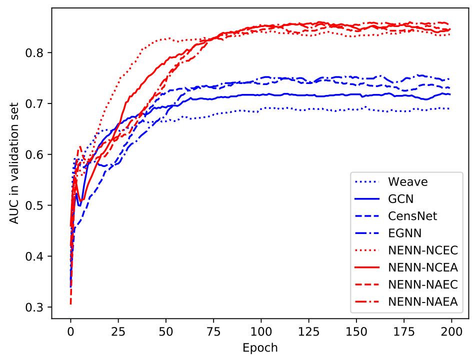
(e) CensNet

(f) NENN (our)

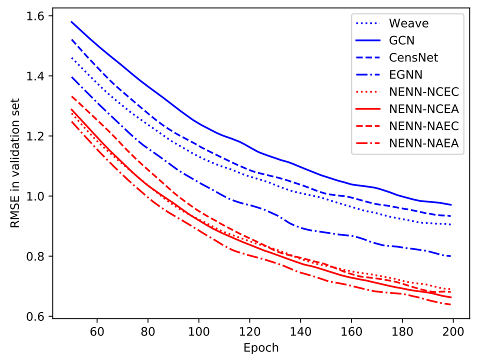
Figure 2: t-SNE visualization of semi-supervised node classification on Cora dataset.

Evaluation	AUC (Classification)				RMSE(Regression)			
Dataset	Tox21		HIV		Lipophilicity		Freesolv	
Data Split	Validation	Test	Validation	Test	Validation	Test	Validation	Test
RF	0.78 ± 0.01	0.75 ± 0.03	0.83 ± 0.02	0.82 ± 0.02	0.87 ± 0.02	0.86 ± 0.04	1.98 ± 0.07	1.62 ± 0.14
Weave	0.79 ± 0.02	0.80 ± 0.02	0.68 ± 0.03	0.71 ± 0.05	0.88 ± 0.06	0.89 ± 0.04	1.35 ± 0.22	1.37 ± 0.14
GCN	0.82 ± 0.02	0.84 ± 0.01	0.70 ± 0.05	0.77 ± 0.02	0.96 ± 0.05	0.98 ± 0.03	1.30 ± 0.09	1.35 ± 0.26
EGNN	0.82 ± 0.01	0.82 ± 0.01	0.73 ± 0.06	0.71 ± 0.05	0.79 ± 0.02	0.75 ± 0.01	1.07 ± 0.08	1.01 ± 0.12
CensNet	0.78 ± 0.00	0.79 ± 0.00	0.74 ± 0.01	0.73 ± 0.02	0.94 ± 0.02	0.83 ± 0.02	1.22 ± 0.02	1.46 ± 0.01
NENN-NAEA	**0.86 ± 0.02**	0.85 ± 0.01	0.84 ± 0.05	**0.84 ± 0.01**	0.67 ± 0.06	**0.67 ± 0.03**	**1.02 ± 0.04**	**1.01 ± 0.01**
NENN-NCEC	0.84 ± 0.02	0.82 ± 0.02	0.83 ± 0.01	0.82 ± 0.01	0.67 ± 0.04	0.73 ± 0.05	1.05 ± 0.14	1.02 ± 0.02
NENN-NCEA	0.85 ± 0.02	0.84 ± 0.01	**0.85 ± 0.02**	0.83 ± 0.01	**0.66 ± 0.02**	0.70 ± 0.08	1.03 ± 0.08	1.01 ± 0.04
NENN-NAEC	0.85 ± 0.01	**0.86 ± 0.02**	0.84 ± 0.01	0.83 ± 0.01	0.69 ± 0.02	0.71 ± 0.01	1.04 ± 0.06	1.01 ± 0.06

Table 4: Prediction results for the four molecular networks.



(a)



(b)

Figure 3: AUC in validation set for HIV networks (left) and RMSE in validation set for Lipophilicity networks (right).

6 ACKNOWLEDGMENTS

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 891–900.

[4] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[6] Lise Getoor Galileo Mark Namata, Ben London and Bert Huang. 2012. Query-driven active survey-ing for collective classification.

[7] Liyu Gong and Qiang Cheng. 2019. Exploiting Edge Features for Graph Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9211–9219.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[10] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. 2018. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4834–4843.

[11] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. 2017. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 11–19.

[12] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. 2019. Censnet: convolution with edge-node switching in graph neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2656–2662.

[13] Patrick F. Riley Oriol Vinyals George E. Dahl Justin Gilmer, Samuel S. Schoenholz. 2017. Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning (ICML)*.

[14] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design* 30, 8 (2016), 595–608.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[18] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[19] David L Mobley and J Peter Guthrie. 2014. FreeSolv: a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design* 28, 7 (2014), 711–720.

[20] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[23] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[24] Martin Simonovsky and Nikos Komodakis. 2017. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *CVPR*. https://arxiv.org/abs/1704.02901

[25] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ accepted as poster.

[27] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.