Universal Graph Neural Networks without Message Passing

Anonymous authors
Paper under double-blind review

Abstract

Message Passing Graph Neural Networks (MP-GNNs) have become the *de facto* paradigm for learning on graph for years. Nevertheless, recent works also obtain promising empirical results with other kinds of architectures like global self-attention and even MLPs. This raises an important theoretical question: what is the minimal prerequisite for an expressive graph model? In this work, we theoretically show that when equipped with proper position encodings, even a simple Bag-of-Nodes (BoN) model (node-wise MLP followed by global readout) can be universal on graphs. We name this model as Universal Bag-of-Nodes (UBoN). Synthetic experiments on the EXP dataset show that UBoN indeed achieves expressive power beyond 1-WL test. On real-world graph classification tasks, UBoN also obtains comparable performance to MP-GNNs while enjoying better training and inference efficiency (50% less training time compared to GCN). We believe that our theoretical and empirical results might inspire more research on simple and expressive GNN architectures.

1 Introduction

In recent years, Message Passing Graph Neural Networks (MP-GNNs) become the prevailing paradigm in the development of Graph Neural Networks (GNNs). Numerous representative GNNs belong to this family, such as GCN (Welling & Kipf, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019). However, it is well known that due to the nature of message passing, the expressive power of MP-GNNs is fundamentally limited by 1-WL test (Weisfeiler & Leman, 1968; Xu et al., 2019). In other words, they can not distinguish a pair of non-isomorphic graphs whose 1-WL test fails, e.g., k-regular graphs. Furthermore, the frequent message passing between nodes also introduces a significant computation overhead in MP-GNNs, which is particularly problematic on large graphs, e.g., the neighbor exploration problem (Zeng et al., 2021). Thus, it seems that existing GNNs are largely limited by the message passing scheme and have both theoretical and empirical demerits, which motivates us to explore efficient and yet powerful alternative paradigms for designing GNNs.

Recently, there has been a surge of interest of directly applying Multi-Layer Perceptrons (MLPs) to graphs due to its simplicity and efficiency. Nevertheless, since most of them still have to pre-process (Wu et al., 2019) or post-process (Huang et al., 2021) the graphs using message-passing operations, these variants still inherent the theoretical limitations of MP-GNNs, as pointed out by Chen et al. (2021). Recently, Hu et al. (2021) show that with a proper training objective, MLPs *alone* can obtain comparable performance to MP-GNNs on node classification tasks while being much more efficient. However, it is easy to see that, theoretically, these MLPs are even more limited than MP-GNNs in expressive power because they do not have structural information. This fact draws our interest to the following problem: can we design a theoretically powerful graph MLP by simply incorporating the structural information as auxiliary input features?

In this paper, we give an affirmative answer to this question by proposing a powerful graph MLP for graph classification tasks, named Universal Bag-of-Nodes (UBoN). UBoN only contains three simple modules: a node-level MLP, a global readout layer (e.g., sum), and a graph-level MLP. As the name suggests, we prove that UBoN enjoys *universal* expressive power (the highest level of expressiveness among GNNs). Consequently, UBoN is not only more powerful than canonical MP-GNNs with 1-WL expressiveness, but also more powerful than high-order GNN variants with

k-WL expressiveness, such as 1-2-3-GNN (Morris et al., 2019). In fact, with universal expressiveness, UBoN is capable of distinguishing any two non-isomorphic graphs, making it one of the most powerful GNNs while being arguably the simplest.

The key part of the proposed UBoN lies in a newly designed permutation-invariant structural embedding, namely, the eigenvalue-aware spectral embedding (EASE). EASE is built upon the canonical spectral embedding (SE) that utilizes the eigenvectors of the Laplacian matrix, *a.k.a.*, Laplacian embedding. SE has demonstrated vital importance on many graph classification tasks (Dwivedi et al., 2020; Yun et al., 2019; Ying et al., 2019) while its drawback is that we cannot reconstruct the graph from it due to the loss of eigenvalue information. Therefore, we propose to incorporate eigenvalues to SE, based on which we can *already* demonstrate the universality of UBoN. Further concerning the non-uniqueness of eigen-decomposition (Wang et al., 2022), we resolve this problem through two extensions of EASE that ensure its sign invariance and basis invariance. We demonstrate that our proposed UBoN is not only theoretically powerful, but also practically efficient for many graph-level tasks, which offers a promising alternative paradigm to existing GNNs. Our contribution is summarized as follows:

- In view of the limitations of MP-GNNs, we advocate for a simpler but yet more powerful paradigm for designing GNNs without using message passing. In particular, we show that for the first time, we can achieve universal expressiveness on graphs using a simple Bag-of-Nodes classifier, named Universal Bag-of-Nodes (UBoN).
- To achieve the universality of UBoN, we design a new kind of structural embedding that preserves all structural information, named eigenvalue-aware spectral embedding (EASE). We show that a simple BoN can obtain universality with EASE, and further achieves sign and basis invariance through two extensions of EASE.
- We evaluate UBoN on both synthetic and real-world datasets. On the one hand, UBoN indeed demonstrates expressive power beyond the 1-WL test. On the other hand, UBoN also achieves comparable performance to MP-GNNs on five real-world graph classification tasks, while requiring less training time and fewer parameters.

2 Related work

Graph MLPs MLPs have been used in computer vision and shown promising results compared to prevalent models (Tolstikhin et al., 2021; Melas-Kyriazi, 2021). Recently, several works proposed to use MLPs on graph representation learning. Hu et al. (2021) proposed a neighboring contrastive loss to allow MLPs to utilize the adjacency information implicitly. Chen et al. (2021) proposed graph-augmented MLPs to augment node features with multi-hop operators before applying learnable node-wise functions, but their expressive power is less powerful than GNNs. Zhang et al. (2022) brought GNNs and MLPs together via knowledge distillation and showed their model can even outperform the teacher GNNs on smaller datasets, but their model is less expressive than GNNs as well. Compared with prior works, our proposed MLP-based model is universal and more efficient.

Expressive Power of GNNs It has been shown that MP-GNNs are not more powerful than the WL test (Xu et al., 2019; Morris et al., 2019). Since then, many attempts have been made to endow graph neural networks with expressive power beyond the 1-WL test, such as injecting random attributes (Murphy et al., 2019; Sato et al., 2021; You et al., 2019; Srinivasan & Ribeiro, 2020; Dwivedi et al., 2020), injecting positional encodings (Li et al., 2020; You et al., 2021), and designing higher-order GNNs (Morris et al., 2019; Maron et al., 2019b;c;a; Chen et al., 2019). Brüel Gabrielsson (2020) produced a framework for constructing universal function approximators on graph isomorphism classes. Most of these attempts still rely on the redundant message passing modules, and suffer from high computational cost or sophisticated network structure. In this paper, we propose a simple MLP-based network that is computationally efficient and has universal expressive power.

Positional Encodings Recently, there are many works that aim to improve the expressive power of GNNs by injecting positional encoding (PE). Murphy et al. (2019) assigned each node an identifier depending on its index. Li et al. (2020) proposed to use distances between nodes characterized by powers of the random walk matrix. You et al. (2019) proposed to use distances of nodes to a sampled anchor set of nodes. Using random node features can also improve the expressiveness of GNNs, even making them universal (Sato et al., 2021; Abboud et al., 2021). Dasoulas et al.

(2021) constructed node representations related to the structural identity of the neighborhood of each node. More detailed discussion of these PE methods can be found in Appendix A. Compared with previous methods, EASE is computationally efficient with fast convergence, while attaining universal expressiveness.

Accelerating Graph Learning There have been many works that aim to speed up graph learning in many ways, such as implementing simpler network structure (Wu et al., 2019; Hu et al., 2021), optimizing the training procedure (Kaler et al., 2022), etc. Wu et al. (2019) proposed a linear model that works well on node classification, but as the authors pointed out in their paper, such linear networks do not work well on graph-level tasks. Hu et al. (2021) proposed an MLP model with NContrast loss that provides the model with structural information, but computing the NContrast loss is computationally expensive during training. In our work, the computation of eigendecompositions of our MLP model happens during pre-processing, thus it is more computationally efficient.

3 Preliminaries

We denote a graph $\mathcal{G} = (\mathbb{V}, \mathbb{E}, \boldsymbol{X})$ where $\mathbb{V}$ is the vertex set of size n, $\mathbb{E}$ is the edge set, and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ are the input node features. We denote $\boldsymbol{A}$ as the adjacency matrix, and let $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{I}+\boldsymbol{A})\boldsymbol{D}^{-\frac{1}{2}}$ be the normalized adjacency matrix, where $\boldsymbol{I}$ denotes the augmented self-loop, $\boldsymbol{D}$ is the diagonal degree matrix of $\tilde{\boldsymbol{A}}$ defined by $D_{i,i} = \sum_{j=1}^{n} \tilde{\boldsymbol{A}}_{i,j}$. $\hat{\boldsymbol{A}}$ can be decomposed as $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$, where $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$ is the diagonal matrix of eigenvalues, the i-th column of $\boldsymbol{U}$ is the eigenvector corresponding to the eigenvalue λ_i, $\lambda_1 \leq \cdots \leq \lambda_n$. We denote concatenation along the last dimension with brackets, *i.e.*, for $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times d}$, $[\boldsymbol{A}, \boldsymbol{B}] \in \mathbb{R}^{n\times 2d}$.

The expressive power of GNNs is characterized by its ability to approximate permutation invariant functions on general graphs. Let $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$ be a compact set of graphs with $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$. Consider a function f acting on Ω. We define its permutation invariance as follows.

Definition 1. *A function $f([\boldsymbol{X}, \hat{\boldsymbol{A}}])$ operating on graphs $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$ is **permutation invariant** if for all permutation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$, $f([\boldsymbol{P}\boldsymbol{X}, \boldsymbol{P}\hat{\boldsymbol{A}}\boldsymbol{P}^{\mathrm{T}}]) = f([\boldsymbol{X}, \hat{\boldsymbol{A}}])$.*

In graph-level tasks like graph classification and graph regression, we learn a GNN to approximate the ground-truth function. A most expressive GNN can approximate arbitrary permutation-invariant functions on graphs up to arbitrary precision, for which we say it has universal expressive power.

Definition 2. *Let $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$ be a compact set of graphs with $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$, where $\boldsymbol{X}$ are the node features and $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix. A GNN is said to be **universal**, if for all continuous permutation invariant functions f defined over Ω and arbitrary $\varepsilon > 0$, there exists network parameters so that $|\operatorname{GNN}(\boldsymbol{X}, \hat{\boldsymbol{A}}) - f([\boldsymbol{X}, \hat{\boldsymbol{A}}])| < \varepsilon$, for all graphs $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$.*

3.1 Expressive power limit of MP-GNNs

In recent years, Message-Passing Graph Neural Networks (MP-GNNs) have become the *de facto* paradigm for graph representation learning. In MP-GNNs, graph convolution is often applied by message passing layers. Consider an MP-GNN with K message passing layers. Let $\boldsymbol{h}_v^{(k)}$ be the feature vector of node v at the k-th layer, we have $\boldsymbol{h}_v^{(k)} = \operatorname{COMBINE}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \operatorname{AGGREGATE}^{(k)}(\{\boldsymbol{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\}))$, for $k = 1, 2, \ldots, K$, where $\mathcal{N}(v)$ is the set of adjacent nodes to node v. For graph classification tasks, we need a readout layer to obtain the final graph representation vector $\boldsymbol{h}_{\mathcal{G}}$: $\boldsymbol{h}_{\mathcal{G}} = \operatorname{READOUT}(\{\boldsymbol{h}_v^{(K)} \mid v \in \mathcal{G}\})$, and the prediction of graph label is $y_{\mathcal{G}} = g(\boldsymbol{h}_{\mathcal{G}})$, where g can be a linear classifier, a Multi-Layer Perceptron, *etc*. For node classification tasks, the label of node v can be predicted as $y_v = f(\boldsymbol{h}_v)$.

However, it is well-known that the expressive power of MP-GNNs is limited by the 1-WL test (Xu et al., 2019). That is, if all node features are the same, and 1-WL test does not distinguish two graphs, then MP-GNNs will fail to distinguish them as well. For example, consider the two graphs in Figure 1. 1-WL test cannot distinguish these two graphs, despite them being non-isomorphic. Thus any MP-GNN will produce identical outputs with these graphs as inputs. If a graph function assigns different values to these two graphs, MP-GNNs will fail to approximate it. This fact inspires us to design more powerful GNNs with expressiveness beyond 1-WL, or even universal expressive power.

4 Building universal GNNs without message passing

In view of the limitations of existing MP-GNNs, in this section, we resort to a new approach and design a simple Bag-of-Node (BoN) classifier with universal expressive power. In Section 4.1, we introduce the vanilla BoN and discuss its limitations due to the lack of structural information. Next, in Section 4.2, we develop a new kind of structural embedding named EASE, based on which we prove that the universality of our Universal BoN (UBoN). In Section 4.3, we design two extensions that achieve the uniqueness of UBoN under different decompositions.

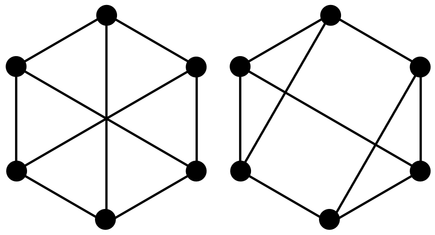
Figure 1: Non-isomorphic graphs that cannot be distinguished by 1-WL and hence by MP-GNNs.

4.1 Bag of Nodes (BoN)

As mentioned above, the expressive power of MP-GNNs is limited by 1-WL test, while the message passing modules in MP-GNNs also bring additional time consumption. So a natural idea is to remove the message passing modules to achieve higher efficiency and more powerful expressiveness.

As an alternative to MP-GNNs, we explore the use of MLPs on graph-level tasks. Given node features $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, a naive approach to generate graph-level embeddings or predictions with MLP is to simply feed $\boldsymbol{X}$ to a node-level MLP, followed by a readout function and a graph-level MLP. Both the node-level and graph-level MLPs are compositions of linear and activation layers. The node-level MLP first acts on each node feature independently to produce node-level embeddings $\boldsymbol{H} \in \mathbb{R}^{n\times d'}$, which is then passed to a readout function. The readout function (e.g., sum) aggregates all the node-level embeddings to produce a graph-level embedding $\boldsymbol{h} \in \mathbb{R}^{d'}$. Finally, the graph-level MLP acts on $\boldsymbol{h}$ to produce the final prediction $\boldsymbol{y}$. The entire model can be formulated as:

$$\boldsymbol{y} = \mathrm{MLP}_{\mathrm{graph}}(\boldsymbol{h}), \quad \boldsymbol{h} = \mathrm{READOUT}\big(\mathrm{MLP}_{\mathrm{node}}(\boldsymbol{X})\big). \tag{1}$$

Similar to Bag of Words (BoW) in natural language processing, the network in Equation 1 considers the node features of the graph as a multiset and aggregates them by a readout function disregarding their dependence. We denote the model described in Equation 1 as **Bag of Nodes** (BoN).

By removing the message passing modules, BoN is free from computational costs caused by neighborhood aggregation in MP-GNNs. However, as BoN has no information about the graph structure, it fails to achieve powerful expressiveness in terms of approximating graph functions dependent on the graph topology. Thus, we need to add structural information to enhance its expressive power.

4.2 Universal BoN (UBoN)

The lack of information about the graph structure limits the expressiveness of BoN. One way to address this issue is to incorporate the structural information as auxiliary input features, *a.k.a.*, positional encodings (PE). Among different PEs, the classical Spectral embedding (SE) (Belkin & Niyogi, 2001) has shown promising performance recently (Dwivedi et al., 2020; Yun et al., 2019). SE utilizes the eigenvectors of the Laplacian matrix as positional encodings of nodes and is equivariant to different graph permutations.

Definition 3 (SE). *The spectral embedding of a graph is constructed by (a partial set of) the eigenvectors $\boldsymbol{U}$ of the Laplacian matrix to the input node features, where $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ is the eigendecomposition of the Laplacian matrix.*

We can augment the input features $\boldsymbol{X}$ with the SE embedding, and get an augmented input $\boldsymbol{X}' = [\boldsymbol{X}, \boldsymbol{U}]$ for GNNs. In this way, SE provides the GNN with structural information about the graph, and provides significant benefits on graph-level tasks. However, combining BoN networks with SE does not give the network universal expressive power, since SE lacks crucial information about eigenvalues of the graph. For instance, consider two non-isomorphic graphs with identical

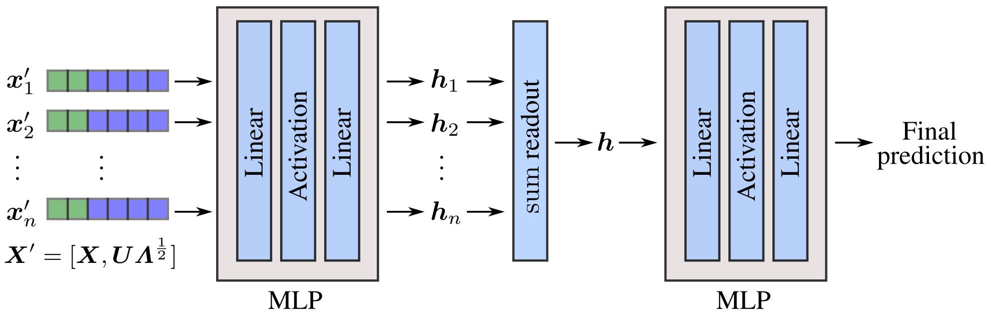

Figure 2: Overview of UBoN.

eigenvectors but different eigenvalues of the Laplacian matrix. Although they are not isomorphic, the spectral embeddings of these two graphs are the same, which indicates the BoN network is not able to tell them apart. For the BoN network to achieve universal expressive power, we propose **eigenvalue-aware spectral embedding** (EASE) to incorporate eigenvalues to SE.

Definition 4 (EASE)**.** *The eigenvalue-aware spectral embedding of a graph is constructed by a combination of the eigenvector and the eigenvalue matrix, i.e., $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, where $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ is the eigen-decomposition of $\hat{\boldsymbol{A}}$.*

Compared with SE, EASE provides the GNN with additional information of the eigenvalues. As we will prove, incorporating EASE to BoN networks already grants universal expressive power.

By applying EASE, we propose a universal and permutation-invariant MLP-based network, named Universal BoN (UBoN). The eigenvalue-aware spectral embeddings of nodes are generated in the preprocessing period, which are then fed into a permutation-invariant BoN network (Section 4.1). The outputs are used as the final graph-level prediction. The entire model is formulated as:

$$\boldsymbol{y} = \mathrm{BoN}([\boldsymbol{X}, \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}]) = \mathrm{MLP}_{\mathrm{graph}}(\boldsymbol{h}), \quad \boldsymbol{h} = \mathrm{READOUT}\big(\mathrm{MLP}_{\mathrm{node}}([\boldsymbol{X}, \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}])\big).$$

The overview of UBoN is shown in Figure 2. Its universality is guaranteed as follows.

Theorem 1. *Let $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$ be a compact set of graphs, $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$. Given a continuous invariant graph function f defined over Ω and arbitrary $\varepsilon > 0$, there exists a permutation-invariant BoN network with parameters such that for all graphs $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$,*

$$|\mathrm{BoN}([\boldsymbol{X}, \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}]) - f([\boldsymbol{X}, \hat{\boldsymbol{A}}])| < \varepsilon.$$

Theorem 1 shows UBoN can approximate arbitrary continuous invariant graph function, which indicates its universality on graph-level tasks. In this way, as simple as it looks like, UBoN is not only more powerful than most MP-GNNs with 1-WL expressiveness, but also more expressive than higher-order GNNs with k-WL expressiveness. Our proof is in Appendix B.1.

In practice, it is not necessary to use all eigenvectors and eigenvalues of the input graphs. Extensive studies have shown that the high-frequency components of the graph are often unhelpful for learning good representations and improving the model's generalization performance (Balcilar et al., 2021; Hoang & Maehara, 2020). Besides, using all eigenvectors and eigenvalues of the graph also increases the computational costs. Thus, we propose to use the first k low-frequency components only. Taking the largest k eigenvectors reduces overfitting caused by high frequency components of $\hat{\boldsymbol{A}}$ and improves the model's generalization performance. It also lessens the computational costs. We can prove that under mild conditions, the loss of expressive power induced by this action can be upper bounded, as shown in the following theorem.

Theorem 2. *Let $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$ be a compact set of graphs, $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$. Given an invariant graph function f defined over Ω that can be ε-approximated by an L_p-Lipschitz continuous function and arbitrary $\varepsilon > 0$, for any integer $0 < k \leq n$, there exists a permutation-invariant BoN network with parameters such that for all graphs $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$,*

$$|f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) - \mathrm{BoN}([\boldsymbol{X}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{:,-k:}, \boldsymbol{0}])| < \sqrt{n-k}L_p\lambda_{n-k} + \varepsilon.$$

Here the L_p-Lipschitz continuity of f is defined using the Frobenius norm on the input domain, $0 \leq \lambda_1 \leq \cdots \leq \lambda_n \leq 2$ are the eigenvalues of $\hat{\boldsymbol{A}}$, $\mathbf{0} \in \mathbb{R}^{n \times (n-k)}$.

We can see from Theorem 2 that the upper bound of the loss of expressive power decreases when k increases, and when $k = n$, our model becomes universal. Proof of Theorem 2 is in Appendix B.2.

Compared to prior works on graph MLPs (Hu et al., 2021; Chen et al., 2021; Zhang et al., 2022), our approach has several advantages. First, it overcomes the expressive power limitation of MLPs, making them universal on graph-level tasks. Secondly, the structure of our network is fairly simple and easy to implement. By removing the redundant message passing modules, our model costs less time to train and has less number of parameters, making them more efficient than traditional MP-GNNs, while still achieving comparable or even better performance on real-world datasets.

4.3 On the non-uniqueness of spectral embedding

By Theorem 1, we have shown that a permutation-invariant MLP-based network can achieve universality on arbitrary graphs. However, as we make use of the eigendecomposition of the adjacency matrix $\hat{\boldsymbol{A}}$, the non-uniqueness of the eigendecomposition could cause problems. The non-uniqueness of decomposition lies in two aspects. First is the ambiguity in sign. For instance, if $\boldsymbol{u}_{\lambda_i}$ is an eigenvector corresponding to the eigenvalue λ_i, then $-\boldsymbol{u}_{\lambda_i}$ is also an eigenvector corresponding to λ_i. Second is the ambiguity in bases. If a eigenvalue λ_i has multiplicity degree $d_i > 1$, and columns of $\boldsymbol{U}_{\lambda_i} \in \mathbb{R}^{n \times d_i}$ are the orthonormal basis of its eigenspace, then for any orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{d_i \times d_i}$, columns of $\boldsymbol{U}_{\lambda_i}\boldsymbol{Q}$ are also the orthonormal basis of its eigenspace. Prior works suggest that the two ambiguities can be potentially harmful for model's stability (Lim et al., 2022; Wang et al., 2022). Below, we give their formal definitions.

Definition 5. *A function f acting on eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$ is **sign invariant** if for all sign choices $s_i \in \{1, -1\}$, $f(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n) = f(s_1\boldsymbol{u}_1, s_2\boldsymbol{u}_2, \ldots, s_n\boldsymbol{u}_n)$.*

Definition 6. *A function f acting on eigenspaces $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n$ ($\boldsymbol{U}_i \in \mathbb{R}^{n \times d_i}$) is **basis invariant** if for all orthogonal matrices $\boldsymbol{Q}_i \in \mathbb{R}^{d_i \times d_i}$, $f(\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n) = f(\boldsymbol{U}_1\boldsymbol{Q}, \boldsymbol{U}_2\boldsymbol{Q}, \ldots, \boldsymbol{U}_n\boldsymbol{Q})$.*

One way to address the two ambiguities in sign and basis is to inject randomness. Inspired by Random Feature (Sato et al., 2021; Abboud et al., 2021) which generates random node features during training, we could use Random Sign (Dwivedi et al., 2020) and Random Basis. Random Sign randomly flips the signs of the eigenvectors during training to encourage the insensitivity to different sign choices. Similarly, Random Basis multiplies the eigenvectors $\boldsymbol{U}$ with a randomly sampled orthogonal matrix (Mezzadri, 2006) during each training period. UBoN enhanced with Random Sign and Random Basis achieves sign and basis invariance in expectation. This is similar to Random Feature, which preserves permutation invariance in expectation (Abboud et al., 2021).

We could also address these problems from the perspective of model structures. First, we consider sign invariance. We design a new sign-invariant extension of UBoN, named UBoN-sign, by combining both positive and negative variants of EASE. We give its universality theorem as follows.

Theorem 3. *Let $\Omega \subset \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$ be a compact set of graphs, $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$. Given a continuous invariant graph function f defined over Ω, there exists a permutation-invariant and sign-invariant BoN network with EASE that can approximate f to an arbitrary precision.*

Then we consider the stronger basis-invariant property. Suppose $\hat{\boldsymbol{A}}$ has k unique eigenvalues $\lambda_1 < \lambda_2 < \cdots < \lambda_k$, and each λ_i has multiplicity d_i. Let $\boldsymbol{U}_{\lambda_i} \in \mathbb{R}^{n \times d_i}$ be the eigenvectors that form a orthonormal basis of the eigenspace corresponding to λ_i. We design another extension of UBoN, named UBoN-basis, which also achieves basis invariance property by incorporating these different basis variants. As the basis invariance is strictly stronger than sign invariance, UBoN-basis is also sign-invariant, and therefore, UBoN-basis can produce unique outputs for different eigendecomposition, while preserving the universal expressive power on graphs.

Theorem 4. *Let $\Omega \subset \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n}$ be a compact set of graphs, $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$. Given a continuous graph function f defined over Ω, there exists a permutation-invariant and basis-invariant BoN network with EASE that can approximate f to an arbitrary precision.*

Theorem 3 and Theorem 4 indicate that by carefully designing the network structure, UBoN can be universal, permutation invariant, and sign/basis invariant at the same time. Our designs of UBoN-sign and UBoN-basis follow the idea of SignNet and BasisNet in Lim et al. (2022). More details and proofs can be found in Appendix B.3 and Appendix B.4.

5 EXPERIMENT

In this section, we evaluate the performance of UBoN on synthetic and real-world datasets. The input node features are fed into a two-layer MLP network, followed by a permutation invariant readout function (e.g., sum, mean, max) and a linear classifier. As mentioned in Section 4.2, we align the last dimension of input node features such that $\boldsymbol{X}' \in \mathbb{R}^{n\times(d+k)}$. We set $k = 32$ in all experiments. Detailed hyperparameter settings are in Appendix C. The experimental results show that UBoN achieves expressive power beyond the WL test and comparable performance with baselines.

5.1 EVALUATION ON SYNTHETIC DATASET

We conduct experiment on the EXP dataset proposed in Abboud et al. (2021), which is designed to explicitly evaluate the expressive power of GNNs. The dataset consists of a set of 1-WL indistinguishable non-isomorphic graph pairs, and each graph instance is a graph encoding of a propositional formula. The classification task is to determine whether the formula is satisfiable (SAT). Since the graph pairs in the EXP dataset are not distinguishable by 1-WL test, if a model shows above 50% accuracy on this dataset, it should have expressive power beyond the 1-WL algorithm.

The models we used on EXP dataset are as follows: an 8-layer GCN, GIN (Xu et al., 2019), PPGN (Maron et al., 2019a), 1-2-3-GCN-L (Morris et al., 2019), 3-GCN (Abboud et al., 2021), BoN-RNI (BoN with Random Node Initialization (RNI) (Abboud et al., 2021)), GCN-RNI (GCN with Random Node Initialization (RNI)), Linear-EASE (linear network with EASE), and UBoN. GCN and GIN belongs to the family of MP-GNNs whose expressive power is bounded by 1-WL, and RNI is a method to improve the expressive power of MP-GNNs. We verify the expressive power gain of EASE-based models by comparing them with GCN, and evaluate their efficiency by comparing them with other expressive models.

We evaluate all models on the EXP dataset using 10-fold cross-validation, and train each model for 500 epochs per fold. Mean test accuracy across all folds is measured and reported. The results are reported in Table 1. In addition, we also measure the learning curves of models to show their convergence rate, as shown in Figure 3.

Table 1: Accuracy results on EXP.

Model	Test Accuracy (%)
GCN	50.0
GIN	50.0
PPGN	50.0
1-2-3-GCN-L	50.0
3-GCN	99.7 ± 0.0
BoN-RNI	50.0
GCN-RNI	97.6 ± 2.5
Linear-EASE	99.1 ± 1.8
UBoN	$\mathbf{99.8} \pm \mathbf{0.5}$

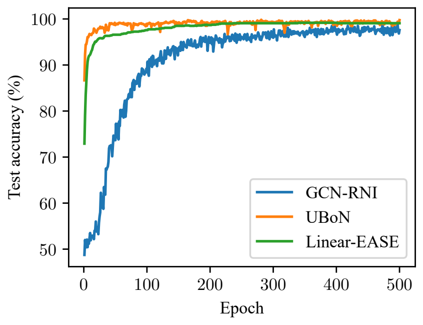

Figure 3: Learning curves on EXP.

In Table 1, we observe that vanilla GCN and BoN-RNI only achieves 50% accuracy, because they do not have expressive power beyond the 1-WL test. UBoN achieves the best performance among all models with a near 100% accuracy, which demonstrates the expressive power of EASE-based models is beyond the 1-WL test. Other models, namely Linear-EASE and GCN-RNI, also achieve comparable performance. It's worth mentioning that even a simple linear model (Linear-EASE) could achieve performance above 99%. This indicates that the universal expressiveness of models with EASE is mostly from EASE itself, rather than the network structure.

From Figure 3, we find that the convergence rate of EASE-based models are much faster than their RNI-based counterpart. This is because EASE-based models are deterministic, while RNI-based

Table 2: Accuracy results on real-world datasets. Models marked with * are implemented by us. The top 2 performance scores are highlighted in bold and underlined format respectively.

Method	MUTAG	PTC	COX2	PROTEINS	ENZYMES
GCN	85.6 ± 5.8	64.2 ± 4.3	-	76.0 ± 3.2	44.0
GIN	89.4 ± 5.6	64.6 ± 7.0	-	$\underline{76.2 \pm 2.8}$	$\underline{59.6 \pm 4.5}$
FDGNN	88.5 ± 3.8	63.4 ± 5.4	83.3 ± 2.9	$\mathbf{76.8 \pm 2.9}$	-
δ-2-LWL	-	-	-	75.1 ± 0.3	56.6 ± 1.2
1-2-3-GNN	86.1	60.9	-	75.5	-
3-hop GNN	87.6 ± 0.7	-	-	75.3 ± 0.4	-
Nested GIN	87.9 ± 8.2	54.1 ± 7.7	-	73.9 ± 5.1	29.0 ± 8.0
Graph-MLP*	69.7 ± 8.1	62.2 ± 6.1	79.4 ± 2.2	66.1 ± 3.0	18.8 ± 4.9
BoN*	70.2 ± 7.8	63.1 ± 7.0	79.7 ± 1.7	66.7 ± 3.5	19.5 ± 2.0
UBoN (Ours)	$\mathbf{91.0 \pm 5.4}$	$\underline{65.4 \pm 5.6}$	$\underline{84.1 \pm 3.9}$	75.7 ± 5.2	57.2 ± 5.1
UBoN + RS (Ours)	$\underline{90.9 \pm 4.8}$	$\mathbf{66.5 \pm 6.5}$	$\mathbf{84.4 \pm 5.2}$	75.7 ± 3.6	$\mathbf{65.7 \pm 6.3}$

models are random and require more training epochs to converge. The structures of UBoN and Linear-EASE are simpler, which means they also train much faster than GCN-RNI.

5.2 Evaluation on real-world datasets

We conduct graph classification experiments on five real-world bioinformatics datasets: MUTAG, PTC, COX2, PROTEINS, and ENZYMES (Yanardag & Vishwanathan, 2015). We perform 10-fold cross-validation and report the average accuracy and standard deviations. Our baselines include: (1) MP-GNNs, including GCN (Welling & Kipf, 2017), GIN (Xu et al., 2019), FDGNN (Gallicchio & Micheli, 2020); (2) expressive models, including δ-2-LWL (Morris et al., 2020), 1-2-3-GNN (Morris et al., 2019), 3-hop GNN (Nikolentzos et al., 2020), Nested GIN (Zhang & Li, 2021); (3) graph MLPs, including Graph-MLP (Hu et al., 2021) and BoN. The results are listed in Table 2. We also conduct experiments on larger scale datasets, and report the results in Appendix D.

As shown in Table 2, UBoN and its RS variant achieve overall comparable, and sometimes superior performance to previous MP-GNNs. In particular, it outperforms the MP-GNN baselines on 4 out of 5 datasets. On the PROTEINS dataset, although inferior to some models, UBoN is still superior to the expressive GNNs like δ-2 LWL, 1-2-3-GNN. Notably, UBoN improves the vanilla BoN by a large margin, up to 38% on ENZYMES and 21% on MUTAG. These results validate the strong expressive power and generalization capability of UBoN. Further comparing UBoN to its Random Sign (RS) variant, we notice that RS can bring some additional benefits in most cases, while the main improvement over vanilla BoN still comes from the proposed EASE in UBoN.

To further investigate the efficiency of UBoN, we train UBoN for 300 epochs on the PROTEINS dataset and summarize their training time. The results are listed in Table 3. We also count the total number of parameters of UBoN and some baselines, and report the results in Table 4.

Table 3: Comparison of training time.

Method	Preprocessing	Training	Total
GCN	0 s	1286.53 s	1286.53 s
GIN	0 s	2861.85 s	2861.85 s
1-2-3-GNN	15.00 s	3442.43 s	3457.43 s
UBoN	9.72 s	587.08 s	**596.80 s**

Table 4: Comparison of #param.

Method	#Parameters
GCN	34.6 k
GIN	30.9 k
1-2-3-GNN	64.3 k
UBoN	**17.5 k**

As shown in Table 3 and Table 4, UBoN is able to achieve comparable or better performance with significantly less training time and number of parameters. Removing the redundant message passing modules reduces the computational cost of UBoN, but not its expressive power. This exhibits the efficiency of UBoN, and we believe it could motivate further researches on other alternatives to MP-GNNs that can overcome the expressiveness bound of 1-WL test.

5.3 EMPIRICAL UNDERSTANDINGS OF UBoN

Effects of sign/basis invariance. In Section 4.3, we discussed the sign/basis invariance issue of BoN, and proved the universality of UBoN-sign and UBoN-basis. Here we test the performances of UBoN-sign and UBoN-basis on five datasets and compare them with vanilla UBoN. As shown in Table 5, the two variants can bring some bonuses on certain datasets, while generally speaking, their performance is on the same level of UBoN and UBoN + RS. This indicates that restricting the structure of UBoN to be explicitly sign/basis invariant does not bring significant accuracy increase, while bringing additional computation overhead, particularly the UBoN-basis variant. Therefore, we believe that the vanilla UBoN with random sign is enough for most real-world applications.

Table 5: Effects of sign and basis invariance.

Method	MUTAG	PTC	COX2	PROTEINS	ENZYMES
UBoN	$\mathbf{91.0 \pm 5.4}$	65.4 ± 5.6	84.1 ± 3.9	$\mathbf{75.7 \pm 5.2}$	$\mathbf{57.2 \pm 5.1}$
UBoN-sign	89.9 ± 5.6	64.5 ± 7.8	$\mathbf{84.2 \pm 2.6}$	$\mathbf{75.7 \pm 2.8}$	55.7 ± 6.2
UBoN-basis	$\mathbf{91.0 \pm 5.3}$	$\mathbf{66.3 \pm 6.8}$	$\mathbf{84.2 \pm 2.6}$	70.9 ± 4.1	44.5 ± 5.4

Effects of number of eigenvectors. In practice, we used only a part of all eigenvectors that corresponds to the largest k eigenvalues to generate EASE. Here, we study the effects of k by changing k and keep other hyperparameters unchanged. We observe in Table 6 that the performance of UBoN drops when k is too small or too large (except on ENZYMES). This is because when k is too small, our model does not have access to enough structural information; and when k is too large, the high frequency eigenvectors that are unhelpful to classification cause our model to overfit.

Table 6: Effects of the number of eigenvectors k adopted in UBoN.

k	0	4	8	16	32	64
MUTAG	69.1 ± 9.5	82.9 ± 7.2	84.1 ± 7.0	83.6 ± 7.5	$\mathbf{91.0 \pm 5.4}$	88.3 ± 5.2
PTC	61.9 ± 6.7	60.5 ± 8.7	62.5 ± 6.9	61.0 ± 6.7	$\mathbf{65.4 \pm 5.6}$	61.6 ± 5.8
COX2	79.0 ± 3.9	80.5 ± 3.8	81.2 ± 3.7	82.9 ± 3.3	$\mathbf{84.1 \pm 3.9}$	79.0 ± 5.4
PROTEINS	66.7 ± 4.9	73.5 ± 3.7	73.3 ± 3.5	73.9 ± 3.5	$\mathbf{75.7 \pm 5.2}$	73.9 ± 5.2
ENZYMES	$\mathbf{60.2 \pm 5.9}$	58.7 ± 5.7	57.5 ± 6.5	56.7 ± 6.8	57.2 ± 5.1	53.0 ± 6.1

Effects of eigenvalues in EASE. In Section 4.2, we defined two types of positional encoding methods, SE and EASE. Here we test the performance of UBoN with SE and EASE to study the effects of using eigenvalues, and report the results in Table 7. We observe that removing eigenvalues in EASE lowers the performance of our model, showing that providing the model with information about eigenvalues is beneficial for graph classification.

Table 7: Effects of the use of eigenvalues in UBoN.

PE method	MUTAG	PTC	COX2	PROTEINS	ENZYMES
SE	86.7 ± 6.5	62.8 ± 4.6	80.1 ± 3.2	75.2 ± 5.4	57.0 ± 6.5
EASE	$\mathbf{91.0 \pm 5.4}$	$\mathbf{65.4 \pm 5.6}$	$\mathbf{84.1 \pm 3.9}$	$\mathbf{75.7 \pm 5.2}$	$\mathbf{57.2 \pm 5.1}$

6 CONCLUSION

In this paper, we proposed eigenvalue-aware spectral embedding (EASE), a positional encoding method that improves the expressive power of graph learning models. We prove that a simple MLP-based BoN network with EASE can achieve universal expressive power while preserving permutation invariance. We name this model as UBoN. We also show that UBoN can further satisfy sign and basis invariance. Extensive experiments show that UBoN achieves stronger expressive power and higher accuracy on most datasets with less training time and fewer parameters. Our findings offer an alternative to the conventional MP-GNNs whose expressiveness is limited by the WL test.

REFERENCES

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *IJCAI*, 2021.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *KDD*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Muhammet Balcilar, Renton Guillaume, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *ICLR*, 2021.

Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NeurIPS*, 2001.

Rickard Brüel Gabrielsson. Universal Function Approximation on Graphs. In *NeurIPS*, 2020.

Lei Chen, Zhengdao Chen, and Joan Bruna. On Graph Neural Networks versus Graph-Augmented MLPs. In *ICLR*, 2021.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with GNNs. In *NeurIPS*, 2019.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *ICLR*, 2016.

George Dasoulas, Giannis Nikolentzos, Kevin Seaman, Aladin Virmaux, and Michalis Vazirgiannis. Ego-Based Entropy Measures for Structural Representations on Graphs. In *ICASSP*, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *arXiv preprint arXiv:2003.00982*, 2020.

Claudio Gallicchio and Alessio Micheli. Fast and Deep Graph Neural Networks. In *AAAI*, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 2017.

NT Hoang and Takanori Maehara. Frequency Analysis for Graph Convolution Network. In *ICLR*, 2020.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. Graph-MLP: Node Classification without Message Passing in Graph. *arXiv preprint arXiv:2106.04051*, 2021.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining Label Propagation and Simple Models out-performs Graph Neural Networks. In *ICLR*, 2021.

Tim Kaler, Nickolas Stathas, Anne Ouyang, Alexandros-Stavros Iliopoulos, Tao Schardl, Charles E Leiserson, and Jie Chen. Accelerating Training and Inference of Graph Neural Networks with Fast Sampling and Pipelining. In *MLSys*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance Encoding—Design Provably More Powerful GNNs for Structural Representation Learning. In *NeurIPS*, 2020.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. In *ICLR Workshop on Geometrical and Topological Representation Learning*, 2022.

Andreas Loukas. What Graph Neural Networks Cannot Learn: Depth vs Width. In *ICLR*, 2020.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. In *NeurIPS*, 2019a.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *ICLR*, 2019b.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the Universality of Invariant Networks. In *ICML*, 2019c.

Luke Melas-Kyriazi. Do You Even Need Attention? A Stack of Feed-Forward Layers Does Surprisingly Well on ImageNet. *arXiv preprint arXiv:2105.02723*, 2021.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *AAAI*, 2019.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *NeurIPS*, 2020.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational Pooling for Graph Representations. In *ICML*, 2019.

Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop Graph Neural Networks. *Neural Networks*, 130:195–205, 2020.

Omri Puny, Heli Ben-Hamu, and Yaron Lipman. Global Attention Improves Graph Networks Generalization. *arXiv preprint arXiv:2006.07846*, 2020.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random Features Strengthen Graph Neural Networks. In *SDM*, 2021.

Balasubramaniam Srinivasan and Bruno Ribeiro. On the Equivalence between Positional Node Embeddings and Structural Graph Representations. In *ICLR*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. MLP-Mixer: An All-MLP Architecture for Vision. In *NeurIPS*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *NeurIPS*, 2021.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and Stable Positional Encoding for More Powerful Graph Neural Networks. In *ICLR*, 2022.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Max Welling and Thomas N Kipf. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

Hassler Whitney. The Self-Intersections of a Smooth n-Manifold in $2n$-Space. *Annals of Mathematics*, pp. 220–246, 1944.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *ICML*, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019.

Pinar Yanardag and SVN Vishwanathan. Deep Graph Kernels. In *KDD*, 2015.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? In *NeurIPS*, 2019.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware Graph Neural Networks. In *ICML*, 2019.

Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware Graph Neural Networks. In *AAAI*, 2021.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph Transformer Networks. In *NeurIPS*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *NeurIPS*, 2017.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the Depth and Scope of Graph Neural Networks. In *NeurIPS*, 2021.

Muhan Zhang and Pan Li. Nested Graph Neural Networks. In *NeurIPS*, 2021.

Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less Neural Networks: Teaching Old MLPs New Tricks via Distillation. In *ICLR*, 2022.

A Other positional encoding methods

In this paper, we proposed EASE, which is a kind of positional encodings. In the field of graph representation learning, many other positional encoding methods have also been proposed. Murphy et al. (2019) proposed Relational Pooling (RP) that assigns each node with an identifier that depends on the index ordering. They showed that RP-GNN is strictly more expressive than the original WL-GNN. However, to ensure permutation equivariance, we have to account for all $n!$ node orderings, which is computationally expensive. You et al. (2019) proposed learnable position-aware embeddings by computing the distance of a target node to an anchor-set of nodes, and showed that P-GNNs have more expressive power than existing GNNs. However, the expressive power of their model depends on the random selection of anchor sets. Sato et al. (2021); Abboud et al. (2021) proposed to use full or partial random node features and proved that their model has universal expressive power, but it has several defects: 1) the loss of permutation invariance, 2) slower convergence, and 3) poor generalization on unseen graphs (You et al., 2019; Loukas, 2020). Li et al. (2020) proposed Distance Encoding (DE) that captures the distance between the node set whose representation is to be learned and each node in the graph. They proved that DE can distinguish node sets embedded in almost all regular graphs where traditional GNNs always fail. However, their approach fails on distance regular graphs, and computation of power matrices limits their model's scalability.

In particular, we will show that with Random Node Initialization (RNI), (1) A linear network is universal on a fixed graph, and (2) An MLP with just one additional message passing layer can be universal on arbitrary graphs.

In our work, we denote RNI as concatenating a random matrix to the input node features. The random matrix can be sampled from Gaussian distribution, uniform distribution, etc. Without loss of generality, we will assume that each entry of the random matrix is sampled independently from the standard Gaussian distribution $N(0, 1)$.

Definition 7. *A GNN with RNI is defined by concatenating a random matrix* $\mathbf{R}$ *to the input node features, i.e.,* $\boldsymbol{X}' = [\boldsymbol{X}, \mathbf{R}]$, *where* $\boldsymbol{X}$ *are the original node features,* $\boldsymbol{X}'$ *are the modified node features and each entry of* $\mathbf{R}$ *is sampled independently from the standard Gaussian distribution* $N(0, 1)$. *The value of* $\mathbf{R}$ *is sampled at every forward pass of GNN.*

To study the effects of RNI on the expressiveness of GNNs, we consider two types of tasks: tasks on a fixed graph (e.g., node classification) and tasks on arbitrary graphs (e.g. graph classification). On a fixed graph, we aim to learn a function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d'}$ that transforms the feature of each node v_i to a presentation vector $\boldsymbol{Z}_{i,:} \in \mathbb{R}^{1\times d'}$. We claim that a linear GNN with RNI in the form

$$[\boldsymbol{X}, \mathbf{R}]\boldsymbol{W} = \boldsymbol{Z} \tag{2}$$

is universal, where $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$ are the network parameters, and $\boldsymbol{Z} \in \mathbb{R}^{n\times d'}$ is the desired output. In other words, we have the universality theorem of linear GNNs with RNI on a fixed graph:

Theorem 5. *On a fixed graph* $\mathcal{G}$, *a linear GNN with RNI defined by Equation 2 is equivariant and can produce any prediction* $\boldsymbol{Z} \in \mathbb{R}^{n\times d'}$ *with probability* 1.

We prove Theorem 5 in Appendix B.5.

On arbitrary graphs, the target function is not only dependent on the node features, but on the graph structure $\hat{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ as well. Let $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$ be a compact set of graphs with $[\boldsymbol{X}, \hat{\boldsymbol{A}}] \in \Omega$, where $\boldsymbol{X}$ are the node features and $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix. We wish to learn a function $f : \Omega \to \mathbb{R}$ that transforms each graph to its label. Since f is also dependent on $\hat{\boldsymbol{A}}$, an MLP-based network with $\boldsymbol{X}'$ as input is not expressive enough, and we need additional graph convolutional layers to obtain information about the graph structure. However, Puny et al. (2020) proved that with just *one* additional message passing layer, an MLP with RNI can approximate any continuous invariant graph function f.

Theorem 6 (Puny et al. (2020))**.** *Given a compact set of graphs* $\Omega \subset \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n}$, *a GNN with one message passing layer, an MLP network and RNI can approximate an arbitrary continuous invariant graph function* $f : \Omega \to \mathbb{R}$ *to an arbitrary precision.*

Theorem 6 is a direct inference of the proof of Proposition 1 in Puny et al. (2020), where the authors constructed a RGNN that first transfers the graph structural information to the node features through a message passing layer, and then approximates f with a DeepSets network, which is an MLP-based network.

B Proofs

B.1 Proof of Theorem 1

We first prove that the EASE $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$ is a real-valued matrix.

Lemma 1. *Suppose $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix of a graph $\mathcal{G}$, and $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ is its spectral decomposition. Then, $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}} \in \mathbb{R}^{n\times n}$.*

Proof. Let $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$. It suffices to show that $\lambda_i \geq 0$ for $i = 1, 2, \ldots, n$.

Let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the vertex set and $\mathbb{E}$ is the edge set. For node i, we denote the degree of node i by d_i. For any $\boldsymbol{x} \in \mathbb{R}^n$, we have

$$\begin{aligned}\boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{A}}\boldsymbol{x} &= \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{I} + \tilde{\boldsymbol{A}})\boldsymbol{x} \\ &= \sum_{i\in\mathbb{V}} x_i^2 + \sum_{(i,j)\in\mathbb{E}} \frac{2x_i x_j}{\sqrt{d_i d_j}} \\ &= \sum_{(i,j)\in\mathbb{E}} \left(\frac{x_i}{\sqrt{d_i}} + \frac{x_j}{\sqrt{d_j}}\right)^2 \geq 0,\end{aligned}$$

thus the Rayleigh quotient of $\hat{\boldsymbol{A}}$ is bounded by $\frac{\boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{A}}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}} \geq 0$. The Rayleigh quotient gives the lower bound of eigenvalues of $\hat{\boldsymbol{A}}$, therefore we have $\lambda_i \geq 0$, and this completes the proof. □

Our construction of universal MLPs on graphs involves transforming a graph invariant function into a set invariant function, thus we also give the definition of invariance on sets. Let $\mathbb{X} = \{x_1, x_2, \ldots, x_M\}$ be a set, where $x_i \in \mathfrak{X}$, and $\mathcal{X} = 2^{\mathfrak{X}}$ be the power set of $\mathfrak{X}$. Consider a function f that transforms its input domain $\mathcal{X}$ to its range $\mathcal{Y}$. Since f acts on sets, we wish the output of f to be invariant to the ordering of elements in $\mathbb{X}$. Formally,

Definition 8. *A function acting on sets $f\colon 2^{\mathfrak{X}} \to \mathcal{Y}$ is **invariant** if for any permutation π,*

$$f(\{x_1, x_2, \ldots, x_M\}) = f(\{x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(M)}\}).$$

Zaheer et al. (2017) studied the property of invariant set functions and gave the following theorem.

Theorem 7 (Zaheer et al. (2017))**.** *Assume the elements are from a compact set in $\mathbb{R}^d$, i.e. possibly uncountable, and the set size is fixed to M. Then any continuous function operating on a set X, i.e. $f\colon \mathbb{R}^{M\times d} \to \mathbb{R}$ which is permutation-invariant to the elements in X can be approximated arbitrarily close in the form of $\rho\left(\sum_{x\in X}\phi(x)\right)$, for suitable transformations ϕ and ρ.*

In practice, we can approximate ρ and ϕ with MLPs. Using the universal approximation theorem of MLPs (Hornik et al., 1989), we have the following corollary.

Corollary 1. *Any continuous invariant set function $f(X)$ operating on a set X with fixed set size having elements from a compact set in $\mathbb{R}^d$ can be approximated by an MLP-based network.*

Then we give the proof of Theorem 1.

Proof. We can rewrite f such that it is a continuous set invariant function on the set consisting of the rows of its input:

$$\begin{aligned}f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) &= f([\boldsymbol{X}, \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}]) \\ &= f\left(\left[\boldsymbol{X}, \left(\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\right)\left(\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\right)^{\mathrm{T}}\right]\right) \\ &= F([\boldsymbol{X}, \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}]).\end{aligned}$$

Using the graph invariance property of f, we can verify that F is set invariant by observing:

$$\begin{aligned} F([\boldsymbol{PX}, \boldsymbol{PU\Lambda}^{\frac{1}{2}}]) &= f\left(\left[\boldsymbol{PX}, \left(\boldsymbol{PU\Lambda}^{\frac{1}{2}}\right)\left(\boldsymbol{PU\Lambda}^{\frac{1}{2}}\right)^{\mathrm{T}}\right]\right) \\ &= f([\boldsymbol{PX}, \boldsymbol{P}\hat{\boldsymbol{A}}\boldsymbol{P}^{\mathrm{T}}]) \\ &= f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) \\ &= F([\boldsymbol{X}, \boldsymbol{U\Lambda}^{\frac{1}{2}}]). \end{aligned}$$

By Lemma 1, the rows of $[\boldsymbol{X}, \boldsymbol{U\Lambda}^{\frac{1}{2}}]$ are in $\mathbb{R}^{d+n}$. Using the conclusion from Corollary 1, we can approximate F arbitrarily close with a DeepSets network, and this completes the proof. □

B.2 Proof of Theorem 2

By Lemma 1, we know that $\lambda_i \geq 0$ for $i = 1, 2, \ldots, n$. Next we prove that $\lambda_i \leq 2$.

Lemma 2. *Suppose $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix of a graph $\mathcal{G}$, and $\lambda_1 < \cdots < \lambda_n$ are its eigenvalues. Then $\lambda_i \leq 2$, for $i = 1, 2, \ldots, n$.*

Proof. In the proof of Lemma 1, we proved $\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{I} + \tilde{\boldsymbol{A}})\boldsymbol{x} \geq 0$. Similarly, we have

$$\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{I} - \tilde{\boldsymbol{A}})\boldsymbol{x} = \sum_{(i,j)\in\mathbb{E}} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}}\right)^2 \geq 0.$$

Thus,

$$\boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}}(-\boldsymbol{I} + \tilde{\boldsymbol{A}})\boldsymbol{x} + 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} \leq 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}.$$

This shows that the Rayleigh quotient is bounded by $\frac{\boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{A}}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}} \leq 2$, therefore $\lambda_i \leq 2$. □

Then we give the proof of Theorem 2.

Proof. Let $0 \leq \lambda_1 \leq \cdots \leq \lambda_n \leq 2$ be the eigenvalues of $\hat{\boldsymbol{A}}$ and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ be the corresponding eigenvectors. Then

$$\hat{\boldsymbol{A}} = \lambda_1\boldsymbol{u}_1\boldsymbol{u}_1^{\mathrm{T}} + \cdots + \lambda_n\boldsymbol{u}_n\boldsymbol{u}_n^{\mathrm{T}}.$$

We also define

$$\hat{\boldsymbol{A}}' \coloneqq \lambda_{n-k+1}\boldsymbol{u}_{n-k+1}\boldsymbol{u}_{n-k+1}^{\mathrm{T}} + \cdots + \lambda_n\boldsymbol{u}_n\boldsymbol{u}_n^{\mathrm{T}}.$$

By Theorem 1 and the assumptions in our theorem, we know that there exists a permutation-invariant BoN network such that

$$|F([\boldsymbol{X}, (\boldsymbol{U\Lambda}^{\frac{1}{2}})_{:,-k:}, \boldsymbol{0}]) - \mathrm{BoN}([\boldsymbol{X}, (\boldsymbol{U\Lambda}^{\frac{1}{2}})_{:,-k:}, \boldsymbol{0}])| < \varepsilon/2.$$

Since f can be approximated by an L_p-Lipschitz continuous function, we have

$$\begin{aligned} |f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) - F([\boldsymbol{X}, (\boldsymbol{U\Lambda}^{\frac{1}{2}})_{:,-k:}, \boldsymbol{0}])| &= |f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) - f([\boldsymbol{X}, \hat{\boldsymbol{A}}'])| \\ &\leq L_p\|[\boldsymbol{X}, \hat{\boldsymbol{A}}] - [\boldsymbol{X}, \hat{\boldsymbol{A}}']\|_{\mathrm{F}} + \varepsilon/2 \\ &= L_p\|[\boldsymbol{0}, \lambda_1\boldsymbol{u}_1\boldsymbol{u}_1^{\mathrm{T}} + \cdots + \lambda_{n-k}\boldsymbol{u}_{n-k}\boldsymbol{u}_{n-k}^{\mathrm{T}}]\|_{\mathrm{F}} + \varepsilon/2 \\ &= L_p\sqrt{\lambda_1^2 + \cdots + \lambda_{n-k}^2} + \varepsilon/2 \\ &\leq \sqrt{n-k}L_p\lambda_{n-k} + \varepsilon/2. \end{aligned}$$

Combining the two inequalities above gives us

$$|f([\boldsymbol{X}, \hat{\boldsymbol{A}}]) - \mathrm{BoN}([\boldsymbol{X}, (\boldsymbol{U\Lambda}^{\frac{1}{2}})_{:,-k:}, \boldsymbol{0}])| < \sqrt{n-k}L_p\lambda_{n-k} + \varepsilon.$$

□

B.3 PROOF OF THEOREM 3

Before proving Theorem 3, we first give some basis topology and algebra definitions for our theoretical results.

Definition 9 (topological space and topological embedding)**.** *A **topological space** $(\mathcal{X}, \tau)$ is defined by a set $\mathcal{X}$ and a collection τ of subsets of $\mathcal{X}$ such that*

1. *The empty set and $\mathcal{X}$ belong to τ.*
2. *Any arbitrary union of members of τ belongs to τ.*
3. *The intersection of any finite number of members of τ belongs to τ.*

*We will omit τ and refer to a topological space as $\mathcal{X}$. A function $f \colon \mathcal{X} \to \mathcal{Y}$ is called a **topological embedding** if it is a homeomorphism from $\mathcal{X}$ to its image.*

Definition 10 (group and topological group)**.** *A **group** is a set G along with a binary operation $\times$ on G such that*

1. *The binary operation satisfies associativity.*
2. *There exists an identity element $e \in G$ such that $e \times a = a \times e = a$, for all $a \in G$.*
3. *For each $a \in G$, there exists an inverse of a in G such that $a \times a^{-1} = a^{-1} \times a = e$.*

*A **topological group** is a topological space that is also a group such that the group operation and the inverse map are continuous.*

Definition 11 (quotient space and quotient map)**.** *A group G may act on a set $\mathcal{X}$ by a function such that the output is also an element of $\mathcal{X}$, which we denote as gx. We define $Gx = \{gx \colon x \in \mathcal{X}\}$ as an equivalence class of $\mathcal{X}$ under G. The **quotient space** is defined by $\mathcal{X}/G = \{Gx \colon x \in \mathcal{X}\}$, which is the set of all such equivalence classes. The **quotient map** is a function $\pi \colon \mathcal{X} \to \mathcal{X}/G$ that maps elements in $\mathcal{X}$ to their equivalence classes.*

Next we give the Decomposition Theorem (Lim et al., 2022), which is useful for representing and approximating invariant functions.

Theorem 8 (Decomposition Theorem)**.** *Let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be topological spaces, and let G_i be a topological group acting continuously on $\mathcal{X}_i$ for each i. Assume that there is a topological embedding $\psi \colon \mathcal{X}_i/G_i \to \mathbb{R}^{a_i}$ of each quotient space into a Euclidean space $\mathbb{R}^{a_i}$ for some dimension a_i. Then, for any continuous function $f \colon \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \to \mathbb{R}^s$ that is invariant to the action of $G = G_1 \times \cdots \times G_k$, there exists continuous functions $\phi_i \colon \mathcal{X}_i \to \mathbb{R}^{a_i}$ and a continuous function $\rho \colon \mathcal{Z} \subseteq \mathbb{R}^a \to \mathbb{R}^s$, where $a = \sum_i a_i$ such that*

$$f(v_1, \ldots, v_k) = \rho\big(\phi_1(v_1), \ldots, \phi_k(v_k)\big).$$

Furthermore: (1) each ϕ_i can be taken to be invariant to G_i, (2) the domain $\mathcal{Z}$ is compact if each $\mathcal{X}_i$ is compact, (3) if $\mathcal{X}_i = \mathcal{X}_j$ and $G_i = G_j$, then ϕ_i can be taken to be equal to ϕ_j.

See Lim et al. (2022) for the proof of Theorem 8. Using the Decomposition Theorem, we have the following lemma.

Lemma 3. *Suppose $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}} \in \mathbb{R}^{n \times n}$. A continuous function f is sign invariant w.r.t. rows of $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, i.e.,*

$$f\left(\left\{\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:} \mid i \in [n]\right\}\right) = f\left(\left\{\boldsymbol{X}_{i,:}, s_i(\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:} \mid i \in [n]\right\}\right)$$

for all $s_i \in \{1, -1\}$, iff there exists continuous ρ and ϕ such that

$$f\left(\left\{\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:} \mid i \in [n]\right\}\right) = \rho\left(\left\{\phi\big(\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big) + \phi\big(\boldsymbol{X}_{i,:}, -(\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big) \mid i \in [n]\right\}\right).$$

Proof. The continuous function

$$f\left(\left\{\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:} \mid i \in [n]\right\}\right)$$

acts on $(\mathbb{R}^d \times \mathbb{R}^n)^n$ and is invariant to the action of $(\{1\} \times \{1,-1\})^n$. Since

$$\mathbb{R}^d/\{1\} \cong \mathbb{R}^d, \quad \mathbb{R}^n/\{1,-1\} \cong \mathbb{R}^n,$$

by Theorem 8, there exists continuous ρ and ϕ' such that

$$f\left(\left\{\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:} \mid i \in [n]\right\}\right) = \rho\left(\left\{\phi'\big(\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big) \mid i \in [n]\right\}\right).$$

Furthermore, we can take ϕ' to be invariant to the action of $\{1\} \times \{1,-1\}$, *i.e.*,

$$\phi'\big(\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big) = \phi'\big(\boldsymbol{X}_{i,:}, -(\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big).$$

Letting $\phi\big(\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big) = \phi'\big(\boldsymbol{X}_{i,:}, (\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}})_{i,:}\big)/2$, we are done. □

In Lemma 3, f being permutation invariant indicates that ρ is also permutation invariant. By Corollary 1 and the universal approximation theorem of MLPs (Hornik et al., 1989), we can approximate ρ with a DeepSets network and ϕ with an MLP. This gives the proof of Theorem 3.

B.4 Proof of Theorem 4

Before proving Theorem 4, we first give the definitions of the Grassmannian and Stiefel manifold.

Definition 12 (Grassmannian)**.** *The Grassmannian,* $\mathrm{Gr}(d,n)$*, is a space that parameterizes all d-dimensional linear subspaces of the n-dimensional vector space $\mathbb{R}^n$.*

The Grassmannian $\mathrm{Gr}(d,n)$ is a smooth manifold of dimension $d(n-d)$.

Definition 13 (Stiefel manifold)**.** *The Stiefel manifold* $\mathrm{St}(d,n)$ *is the set of all orthonormal tuples* $[\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_d] \in \mathbb{R}^{n\times d}$ *of d vectors in* $\mathbb{R}^n$.

Then we give the First Fundamental Theorem of $O(d)$ (Villar et al., 2021) and the Whitney Embedding Theorem (Whitney, 1944).

Theorem 9 (First Fundamental Theorem of $O(d)$)**.** *A continuous function $f\colon \mathbb{R}^{n\times d} \to \mathbb{R}^s$ is orthogonally invariant, i.e. $f(\boldsymbol{V}\boldsymbol{Q}) = f(\boldsymbol{V})$ for all $\boldsymbol{Q} \in O(d)$, iff $f(\boldsymbol{V}) = h(\boldsymbol{V}\boldsymbol{V}^{\mathrm{T}})$ for some continuous h.*

Theorem 10 (Whitney Embedding Theorem)**.** *Every smooth manifold $\mathcal{M}$ of dimension $n > 0$ can be smoothly embedded in $\mathbb{R}^{2n}$.*

See Villar et al. (2021) for the proof of Theorem 9, Whitney (1944) for the proof of Theorem 10. Using the Decomposition Theorem, we have the following lemma.

Lemma 4. *Suppose $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, $\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}} \in \mathbb{R}^{n\times n}$, where $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ is the spectral decomposition of $\hat{\boldsymbol{A}}$. A continuous function f is basis invariant w.r.t. eigenspaces, i.e.,*

$$f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right) = f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}\boldsymbol{Q}_i, \lambda_i \mid i \in [k]\}\right)$$

for all orthogonal $Q_i \in O(d_i) \subseteq \mathbb{R}^{d_i\times d_i}$, iff there exists continuous ρ, ψ, ϕ_{d_i} such that

$$f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right) = \rho\left(\psi(\boldsymbol{X}), \left\{\phi_{d_i}(\boldsymbol{U}_{\lambda_i}\boldsymbol{U}_{\lambda_i}^{\mathrm{T}}, \lambda_i) \mid i \in [k]\right\}\right).$$

Proof. The continuous function

$$f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right)$$

acts on $\mathbb{R}^{n\times d} \times (\mathrm{St}(d_i,n) \times \mathbb{R})^k$ and is invariant to the action of $\{1\} \times (O(d_i) \times \{1\})^n$. Since

$$\mathbb{R}^{n\times d}/\{1\} \cong \mathbb{R}^{n\times d}, \quad \mathrm{St}(d_i,n)/O(d_i) \cong \mathrm{Gr}(d_i,n), \quad \mathbb{R}/\{1\} \cong \mathbb{R},$$

and by Theorem 10, there is a topological embedding $\psi_{d_i}\colon \mathrm{Gr}(d_i,n) \to \mathbb{R}^{2d_i(n-d_i)}$, by Theorem 8, there exists continuous ρ, ψ, ϕ'_{d_i} such that

$$f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right) = \rho\left(\psi(\boldsymbol{X}), \left\{\phi'_{d_i}(\boldsymbol{U}_{\lambda_i}, \lambda_i) \mid i \in [k]\right\}\right).$$

Furthermore, we can take ϕ'_{d_i} to be invariant to the action of $O(d_i) \times \{1\}$, *i.e.*,

$$\phi'_{d_i}(\boldsymbol{U}_{\lambda_i}, \lambda_i) = \phi'_{d_i}(\boldsymbol{U}_{\lambda_i}\boldsymbol{Q}, \lambda_i)$$

for all $\boldsymbol{Q} \in O(d_i)$. By Theorem 9, there exists a continuous ϕ_{d_i} such that

$$\phi'_{d_i}(\boldsymbol{U}_{\lambda_i}, \lambda_i) = \phi_{d_i}(\boldsymbol{U}_{\lambda_i}\boldsymbol{U}_{\lambda_i}^{\mathrm{T}}, \lambda_i).$$

This completes the proof. □

In Lemma 4, if f is permutation invariant, *i.e.*, for all permutation matrices $\boldsymbol{P} \in \mathbb{R}^{n\times n}$,

$$f\left(\boldsymbol{X}, \{\boldsymbol{U}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right) = f\left(\boldsymbol{PX}, \{\boldsymbol{PU}_{\lambda_i}, \lambda_i \mid i \in [k]\}\right),$$

by Theorem 8, ψ and ϕ'_{d_i} can be taken to be permutation invariant, thus ϕ_{d_i} is graph invariant. Since ψ can be ϵ-approximated by an invariant set network and ϕ_{d_i} can be ϵ-approximated by an invariant graph network (Lim et al., 2022), Theorem 4 is proved.

B.5 Proof of Theorem 5

We first prove the following lemmas.

Lemma 5. *Let* $\mathbf{R} \in \mathbb{R}^{n\times n}$ *be a random matrix, and each entry of* $\mathbf{R}$ *is sampled independently from the standard Gaussian distribution* $N(0,1)$*. Then with probability* 1*,* $\mathbf{R}$ *has full rank.*

Proof. We denote the first column of $\mathbf{R}$ by $\mathbf{R}_{:,1}$. It is linearly independent because $\mathbf{R}_{:,1} = \mathbf{0}$ with probability 1. Then we view $\mathbf{R}_{:,1}$ as fixed, and consider the second column $\mathbf{R}_{:,2}$. The probability that $\mathbf{R}_{:,2}$ falls into the span of $\mathbf{R}_{:,1}$ is 0, thus with probability 1, $\mathbf{R}_{:,1}$ and $\mathbf{R}_{:,2}$ are linearly independent.

Generally, let us consider the k-th column $\mathbf{R}_{:,k}$. The first $k-1$ columns of $\mathbf{R}$ forms a subspace in $\mathbb{R}^n$ whose Lebesgue measure is 0. Thus $\mathbf{R}_{:,k}$ falls into this subspace with probability 0. By inference, we have all the columns of $\mathbf{R}$ are linearly independent with probability 1, *i.e.*, $P(\operatorname{rank}(\mathbf{R}) = n) = 1$. □

Lemma 6. *Let* $\boldsymbol{A} \in \mathbb{R}^{s\times n}$*,* $\boldsymbol{B} \in \mathbb{R}^{s\times m}$ *be two matrices. Then the equation* $\boldsymbol{AX} = \boldsymbol{B}$ *has a solution iff.* $\operatorname{rank}(\boldsymbol{A}) = \operatorname{rank}([\boldsymbol{A}, \boldsymbol{B}])$.

Proof. First we prove the necessity. Suppose $\boldsymbol{AX} = \boldsymbol{B}$ has a solution. Then,

$$[\boldsymbol{A}_{:,1}, \boldsymbol{A}_{:,2}, \ldots, \boldsymbol{A}_{:,n}]\boldsymbol{X}_{:,i} = \boldsymbol{B}_{:,i},$$

where $\boldsymbol{M}_{:,i}$ denotes the i-th column of matrix $\boldsymbol{M}$. This means each column of $\boldsymbol{B}$ can be expressed as a linear combination of the columns of $\boldsymbol{A}$, and therefore each column of $[\boldsymbol{A}, \boldsymbol{B}]$ can be expressed as a linear combination of the columns of $\boldsymbol{A}$.

On the other hand, it is obvious that each column of $\boldsymbol{A}$ can be expressed as a linear combination of the columns of $[\boldsymbol{A}, \boldsymbol{B}]$. Thus we have $\operatorname{rank}(\boldsymbol{A}) = \operatorname{rank}([\boldsymbol{A}, \boldsymbol{B}])$.

Then we prove the sufficiency. Since $\operatorname{rank}(\boldsymbol{A}) = \operatorname{rank}([\boldsymbol{A}, \boldsymbol{B}])$, and each column of $\boldsymbol{A}$ can be expressed as a linear combination of columns of $[\boldsymbol{A}, \boldsymbol{B}]$, we have the columns of $\boldsymbol{A}$ and the columns of $[\boldsymbol{A}, \boldsymbol{B}]$ are equivalent. Therefore, each column of $\boldsymbol{B}$ can be expressed as a linear combination of columns of $\boldsymbol{A}$, *i.e.*, $\boldsymbol{Ax} = \boldsymbol{B}_{:,i}$ has a solution for $i = 1, \ldots, m$. Thus the equation $\boldsymbol{AX} = \boldsymbol{B}$ has a solution. □

Then we give the proof of Theorem 5.

Proof. For any prediction $\boldsymbol{Z} \in \mathbb{R}^{n\times d'}$, we wish to prove that with probability 1, there exists parameters of a linear GNN with RNI $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$ such that

$$[\boldsymbol{X}, \mathbf{R}]\boldsymbol{W} = \boldsymbol{Z}. \tag{3}$$

By Lemma 6, the necessary and sufficient condition that Equation 3 has a solution $\boldsymbol{W}$ is $\operatorname{rank}([\boldsymbol{X}, \mathbf{R}]) = \operatorname{rank}([\boldsymbol{X}, \mathbf{R}, \boldsymbol{Z}])$.

By Lemma 5, with probability 1, $\operatorname{rank}(\mathbf{R}) = n$, therefore $\operatorname{rank}([\boldsymbol{X}, \mathbf{R}]) = \operatorname{rank}([\boldsymbol{X}, \mathbf{R}, \boldsymbol{Z}]) = n$.

Thus, in conclusion, with probability 1, there exists parameters of a linear GNN with RNI $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$ such that the GNN produces $\boldsymbol{Z}$.

We can also prove linear GNNs' equivariance by observing that for any permutation matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$,

$$[\boldsymbol{PX}, \mathbf{R}]\boldsymbol{W} = \boldsymbol{P}[\boldsymbol{X}, \mathbf{R}]\boldsymbol{W} = \boldsymbol{PZ},$$

where we used $\boldsymbol{P}\mathbf{R} = \mathbf{R}$ because each entry of $\boldsymbol{P}\mathbf{R}$ is also sampled from the standard Gaussian matrix $N(0,1)$. □

C Hyperparameter settings

We use Optuna (Akiba et al., 2019) to optimize the hyperparameters of our models.

In experiments on synthetic dataset, the hyperparameters of our models are as follows:

- **GCN-SE**: the learning rate $\lambda = 0.000135737656276214$, the hidden dimension $w = 87$, the number of GCN layers $l = 9$, whether to normalize the GCN layers $\mathrm{normLayers} = \mathrm{False}$, the weight decay of Adam optimizer (Kingma & Ba, 2015) $wd = 0.0005$.
- **UBoN**: the learning rate $\lambda = 0.002385602941230316$, the hidden dimension of the first linear layer $w_1 = 60$, the hidden dimension of the second linear layer $w_2 = 76$, the dropout rate (Srivastava et al., 2014) $p = 0.13592575703525184$, the weight decay of Adam optimizer $wd = 0.0005$.
- **Linear-SE**: the learning rate $\lambda = 0.0006867736568978745$, the hidden dimension $w = 109$, the weight decay of Adam optimizer $wd = 0.0001$.

In experiments on real-world datasets, we tune the following hyperparameters: the learning rate λ, the hidden dimension of the first linear layer w_1, the hidden dimension of the second linear layer w_2, whether to use layer norm (Ba et al., 2016) $\mathrm{normLayers}$, whether to normalize the adjacency matrix $\mathrm{normAdj}$, the dropout rate p and the weight decay of Adam optimizer wd. The values are listed in Table 8 and Table 9.

Table 8: Hyperparameter settings of UBoN on real-world datasets.

Datasets	λ	w_1	w_2	normLayers	normAdj	p	wd
MUTAG	0.0007796404775632826	115	107	True	False	0.005306946761854224	0.001
PTC	2.8075805032171297e-05	127	52	True	False	0.08906092283547215	0.0005
COX2	0.0007746225551008818	104	59	False	False	0.3879977534990946	0.0001
PROTEINS	0.0002	119	106	False	True	0.07	0.0005
ENZYMES	0.0009086946949921901	80	128	False	False	0.01618402806942577	0.001

Table 9: Hyperparameter settings of UBoN + RS on real-world datasets.

Datasets	λ	w_1	w_2	normLayers	normAdj	p	wd
MUTAG	0.0009	93	122	True	False	0.003	0.0001
PTC	0.00027018422728643227	127	46	True	False	0.3808908566310215	0.0
COX2	0.000864272829565321	122	88	False	False	0.16	5e-06
PROTEINS	0.00014	97	119	False	True	0.08	5e-06
ENZYMES	0.0009275933776833255	113	123	True	False	0.00028393342663080457	0.0001

We use ELU non-linearities (Clevert et al., 2016) in all experiments. We set the batch size to be 36 on PROTEINS and 20 on all other datasets.

D Experiment results on large scale datasets

We conduct experiments on the following three large scale datasets: DD (Yanardag & Vishwanathan, 2015), ogbg-molhiv and ogbg-moltoxcast (Hu et al., 2020). On the DD dataset, we use the same settings as in Section 5.2, *i.e.*, perform 10-fold cross-validation and report the average accuracy. On the OGB datasets, we use the settings provided by the OGB team, *i.e.*, split the datasets into train, validation and test set, and report the test accuracy at the best validation epoch. We compare the performance of UBoN with GIN (Xu et al., 2019), which is a widely used MP-GNN, and also with vanilla BoN to show the effects of UBoN. The results are listed in Table 10.

As shown in Table 10, UBoN greatly improves the performance of BoN, demonstrating the efficacy of applying EASE, while also achieving comparable performance with its MP-GNN counterpart. On the other hand, the number of parameters of UBoN is significantly less than GIN, showing that our UBoN is indeed a promising efficient alternative to message passing GNNs.

We also report the dataset details and the pre-processing and training time (300 epochs) of UBoN in Table 11.

Table 10: Accuracy results on large scale datasets. We also list the number of parameters of these models.

Method	#params	ogbg-molhiv	ogbg-moltoxcast	DD
GIN	1885 k	75.58 ± 1.40	63.41 ± 0.74	75.3 ± 2.9
BoN	16 k	69.02 ± 2.30	58.02 ± 0.25	68.8 ± 3.5
UBoN	19 k	73.41 ± 0.49	62.50 ± 0.32	76.1 ± 2.3

Table 11: Dataset details and the pre-processing and training time of UBoN on large scale datasets.

Dataset	#Graphs	#Avg Nodes	Pre-processing time	Training time
ogbg-molhiv	41127	25.5	37.04 s	10124.55 s
ogbg-moltoxcast	8576	18.8	8.87 s	7636.93 s
DD	1178	284.3	26.44 s	3746.40 s

In Table 11, it can be seen that even on datasets with larger number of graphs or nodes, the pre-processing time is relatively less than training time, showing that the computation cost of eigen-decomposition is acceptable. On the other hand, the main training cost of our UBoN is simply an MLP network, which is very fast to train without any message passing steps. This demonstrates the efficiency and scalability of UBoN.