IsoNN: Isomorphic Neural Network for Brain Graph Representation Learning and Classification

Lin Meng, *Student Member, IEEE* and Jiawei Zhang

***Abstract*—The studies of brain graphs have been an interesting topic in the neuroimaging and healthcare. However, it is hard to apply traditional deep learning models like CNN on the brain graph data due to the 'node-orderless' property. Normally, adjacency matrices will cast an artificial and random node-order on the graphs, which renders the performance of deep models on graph classification tasks extremely erratic, and the representations learned by such models lack clear interpretability. To eliminate the unnecessary node-order constraint, we propose a novel model named Isomorphic Neural Network (IsoNN), which learns the brain graph representations by extracting its isomorphic features via the subgraph matching between input graph and templates. IsoNN has two main components: graph isomorphic feature extraction component and classification component. The graph isomorphic feature extraction component utilizes a set of subgraph templates as the kernel variables to learn the possible subgraph patterns existing in the input graph and then computes the isomorphic features. A set of permutation matrices is used in the component to break the node-order brought by the matrix representation. Three fully-connected layers are used as the classification component in IsoNN. Extensive experiments are conducted on real-world brain graph datasets, the experimental results can demonstrate the effectiveness of IsoNN.**

***Index Terms*—Representation Learning, Graph Neural Network; Brain Graph**

I. Introduction

In the fields of neuroscience and healthcare, the analyses of the human brain have been always a focus of researchers [1]–[3]. Functional magnetic resonance imaging (fMRI) and structural diffusion tensor imaging (DTI) have been two of many techniques for analyzing human brains. In recent years, many brain image works [1], [2], [4] apply graph theory to analyze the brain image data. The brain graph can be obtained by computing pairwise correlations between fMRI or DTI time series of different regions of interests (ROIs) [3]. In this paper, we will focus on the brain graph classification for brain-related diseases, *e.g.*, HIV infection, attention-deficit/hyperactivity disorder (ADHD), and bipolar disorder (BP), based on the constructed brain graphs.

To address the aforementioned task, many graph learning methods can be applied. One way to estimate the usefulness of subgraph features is feature evaluation criteria based on both labeled and unlabeled graphs [5]. Some other works also proposed to design a pattern exploration approach based on pattern co-occurrence and build the classification model [6] or develop a boosting algorithm [7]. However, such works based on BFS or DFS cannot avoid computing a large number of possible subgraphs, which causes high computational complexity though the explicit subgraphs are maintained. Recently, deep learning models are also widely used to solve the graph-oriented problems. Although some deep models like MPNN [8] and GCN [9] learn implicit structural features, the explicit structural information cannot be maintained for further research. Besides, most existing works on graph classification use the aggregation of the node features in graphs as the graph representation [10], [11], but simply doing aggregation on the whole graph cannot capture the substructure precisely. While there are other models can capture the subgraphs, they often need more complex computation and mechanism [12], [13] or need additional node labels to find the subgraph strcuture [14], [15].

However, we should notice that when we deal with the graph-structured data to explore substructures, different node-orders will result in very different adjacency matrix representations for most existing deep models (*e.g.*, CNN) which take the adjacency matrices as input if there is no other information on graphs. Therefore, compared with the original graph, the matrix naturally poses a redundant constraint on the graph node-order. Such a node-order is usually unnecessary and manually defined. The different graph matrix representations brought by the node-order differences may render the learning performance of the existing models to be extremely erratic. Formally, we summarize the encountered challenges in the detecting substructures existing in brain graphs as follows:

- **Explicit useful subgraph extraction.** The existing works have proposed many discriminative models to discover useful subgraphs for graph classification, and most of them require manual efforts. Nevertheless, how to select the contributing subgraphs automatically without any additional manual involvement is a challenging problem.
- **Brain graph representation learning.** Representing graphs in the vector space is an important task since it facilitates the storage, parallelism and the usage of machine learning models for the graph data. Extensive works have been done on node representations [11], [16]–[18], whereas learning the representation of the whole graph with clear interpretability is still an open problem requiring more explorations.
- **Node-order elimination for subgraphs.** Nodes in graphs are orderless, whereas the matrix representations of graphs cast an unnecessary order on nodes, which also renders the features extracted with the existing learning

L. Meng is with the Department of Computer Science, Florida State University, Tallahassee FL, 32303, USA (email: lin@ifmlab.org).

J. Zhang is with the Department of Computer Science, University of California, Davis CA, 95616, USA (email: jiawei@ifmlab.org).

Manuscript received April 19, 2021; revised August 16, 2021.

models, e.g., CNN, to be useless for the graphs. For subgraphs, this problem also exists. Thus, how to break such a node-order constraint for subgraphs is challenging.
- **Efficient matching for large subgraphs.** To break the node-order, we will try all possible node permutations to find the best permutation for a subgraph. Clearly, trying all possible permutations is a combinatorial explosion problem, which is extremely time-consuming for finding large subgraph templates. The problem shows how to accelerate the proposed model for large subgraphs that also need to be solved.

In this paper, we propose a novel model, namely **Iso**morphic **N**eural **N**etwork (ISONN) and its variant, to address the aforementioned challenges in the brain graph representation learning and classification problem. ISONN is composed of two components: the graph isomorphic feature extraction component and the classification component, aiming at learning isomorphic features and classifying graph instances, respectively. In the graph isomorphic feature extraction component, ISONN automatically learns a group of subgraph templates of useful patterns from the input graph. ISONN makes use of a set of permutation matrices, which act as the node isomorphism mappings between the templates and the input graph. With the potential isomorphic features learned by all the permutation matrices and the templates, ISONN adopts one min-pooling layer to find the best node permutation for each template and one softmax layer to normalize and fuse all subgraph features learned by different kernels, respectively. Such features learned by different kernels will be fused together and fed as the input for the classification component. ISONN further adopts three fully-connected layers as the classification component to project the graph instances to their labels. Moreover, to accelerate the proposed model when dealing with large subgraphs, we also propose the fast version and the deep model of ISONN to guarantee efficiency.

II. Terminology and Problem Definition

In this section, we will define the notations and the terminologies used in this paper and give the formulation for the brain graph classification problem.

A. Notations

In the following sections, we will use lower case letters like x to denote scalars, lower case bold letters (e.g. $\mathbf{x}$) to represent vectors, bold-face capital letters (e.g. $\mathbf{X}$) to show the matrices. For tensors or sets, capital calligraphic letters are used to denote them. We use $\mathbf{x}_i$ to represent the i-th element in $\mathbf{x}$. Given a matrix $\mathbf{X}$, we use $\mathbf{X}(i,j)$ to express the element in i-th row and j-th column. For i-th row vector and j-th column vector, we use $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ to denote respectively. Moreover, notations $\mathbf{x}^\top$ and $\mathbf{X}^\top$ denote the transpose of vector $\mathbf{x}$ and matrix $\mathbf{X}$ respectively. Besides, the F-norm of matrix $\mathbf{X}$ can be represented as $\|\mathbf{X}\|_F = (\sum_{i,j} |X_{i,j}|^2)^{\frac{1}{2}}$.

B. Problem Formulation

DEFINITION *1:* (Brain Graph): Formally, a brain graph can be represented as weighted graph $G = (\mathcal{V}, \mathcal{E})$, where the sets $\mathcal{V}$ and $\mathcal{E}$ denote the nodes and links involved in the graph, respectively. Each node $i \in \mathcal{V}$ denotes a region of interests (ROI) and each edge $e_{ij} = (i, j, w_{ij}) \in \mathcal{E}$ represents the connectivity degree between two node i and j is w_{ij}.
Now, we can define our problem with the brain graph defination.

DEFINITION *2:* Problem Definition: Formally, given a brain graph set $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ with a small number of labeled brain graph instances, the graph classification problem aims at learning a mapping, i.e., $f : \mathcal{G} \rightarrow \mathcal{Y}$, to project each brain graph instance into a pre-defined label space $\mathcal{Y} = \{+1, -1\}$.
In this paper, we will take the graph binary classification as an example to illustrate the problem setting for ISONN. A simple extension of the model can be applied to handle more complicated learning scenarios with multi-class or multi-label as well.

III. Proposed Method

The overall architecture of ISONN is shown in Figure 1. The ISONN framework includes two main components: graph isomorphic feature extraction component and classification component. The graph isomorphic feature extraction component includes a graph isomorphic layer, a min-pooling layer as well as a softmax layer and the classification component is composed by three fully-connected layers. They will be discussed in detail in the following subsections.

A. Graph Isomorphic Feature Extraction Component

Graph isomorphic feature extraction component targets at learning the graph features. To achieve that objective, ISONN adopts an automatic feature extraction strategy for graph representation learning. In ISONN, one graph isomorphic feature extraction component involves three layers: the graph isomorphic layer, the min-pooling layer and the softmax layer. In addition, we can further construct a deep graph isomorphic neural network by applying multiple isomorphic feature extraction components on top of each other, *i.e.*, apply the combination of "graph isomorphic layer, min pooling layer, softmax layer" several times. For the second and latter components, they will be used on every feature matrix learned by the combination of channels of all former components.

1) Graph Isomorphic Layer: Graph isomorphic layer is the first effective layer in deep learning that handles the node-order restriction in graph representations. Assume we have a graph $G = \{\mathcal{V}, \mathcal{E}\}$, and its adjacency matrix to be $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and each element $\mathbf{A}(i,j) = w_{ij}$. In order to find the existence of specific subgraph patterns in the input graph, ISONN matches the input graph with a set of subgraph templates. Each template is denoted as a kernel variable $\mathbf{K}_i \in \mathbb{R}^{k \times k}, \forall i \in \{1, 2, \cdots, c\}$. Here, k denotes the node number in subgraphs and c is the channel number (i.e., total template count). Meanwhile, to match one template with regions in the input graph (i.e., sub-matrices in $\mathbf{A}$), we use a

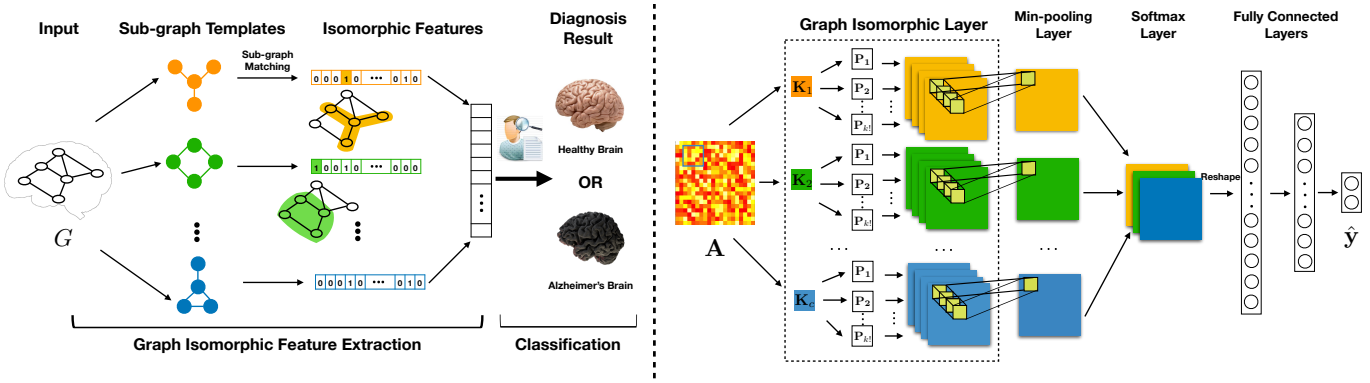


Fig. 1: IsoNN Framework Architecture. (The left subplot provides the outline of the proposed framework, including the *graph isomorphic feature extraction* component and the *classification* component respectively. Meanwhile, the right subplot illustrates the detailed architecture of the proposed framework, where the *graph isomorphic features* are extracted with the *graph isomorphic layer*, *min-pooling layer* and *softmax layer*, and the graphs are further classified with three fully-connected layers.)

set of permutation matrices, which map both rows and columns of the kernel variable to the subgraphs effectively. The permutation matrix can be represented as $\mathbf{P} \in \{0,1\}^{k\times k}$ that shares the same dimension with the kernel variable. Therefore, given a kernel $\mathbf{K}_i$ and a sub-matrix $\mathbf{M}_{(s,t)} \in \mathbb{R}^{k\times k}$ in $\mathbf{A}$ (*i.e.*, a region in the input graph G and $s,t \in \{1,2,\cdots,(|\mathcal{V}|-k+1)\}$ denotes a starting index pair in $\mathbf{A}$), there may exist $k!$ different such permutation matrices. The optimal should be the matrix $\mathbf{P}^*$ that minimizes the following term.

$$\mathbf{P}^* = \arg\min_{\mathbf{P}\in\mathcal{P}} \left\|\mathbf{P}\mathbf{K}_i\mathbf{P}^\top - \mathbf{M}_{(s,t)}\right\|_F^2, \tag{1}$$

where $\mathcal{P} = \{\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_{k!}\}$ covers all the potential permutation matrices. Formally, the isomorphic feature extracted based on the kernel $\mathbf{K}_i$ for the regional sub-matrix $\mathbf{M}_{(s,t)}$ in $\mathbf{A}$ can be represented as

$$\begin{aligned} z_{i,(s,t)} &= \left\|\mathbf{P}^*\mathbf{K}_i(\mathbf{P}^*)^\top - \mathbf{M}_{(s,t)}\right\|_F^2 \\ &= \min\{\left\|\mathbf{P}\mathbf{K}_i\mathbf{P}^\top - \mathbf{M}_{(s,t)}\right\|_F^2\}_{\mathbf{P}\in\mathcal{P}} \\ &= \min(\bar{\mathbf{z}}_{i,(s,t)}(1:k!)), \end{aligned} \tag{2}$$

where vector $\bar{\mathbf{z}}_{i,(s,t)}$ contains entry $\bar{\mathbf{z}}_{i,(s,t)}(j) = \left\|\mathbf{P}_j\mathbf{K}_i\mathbf{P}_j^\top - \mathbf{M}_{(s,t)}\right\|_F^2$, $\forall j \in \{1,2,\cdots,k!\}$ denoting the isomorphic features computed by the j-th permutation matrix $\mathbf{P}_j \in \mathcal{P}$.

As indicated by the Figure 1, IsoNN computes the final isomorphic features for the kernel variable $\mathbf{K}_i$ via two steps: (1) computing all the potential isomorphic features via different permutation matrices with the graph isomorphic layer, and (2) identifying and fusing the optimal features with the min-pooling layer and softmax layer to be introduced as follows. By shifting one kernel matrix $\mathbf{K}_i$ on regional sub-matrices, IsoNN extracts the isomorphic features on the matrix $\mathbf{A}$, which can be denoted as a 3-way tensor $\bar{\mathcal{Z}}_i \in \mathbb{R}^{k!\times(|\mathcal{V}|-k+1)\times(|\mathcal{V}|-k+1)}$, where $\bar{\mathcal{Z}}_i(1:k!,s,t) = \bar{\mathbf{z}}_{i,(s,t)}(1:k!)$. In a similar way, we can also compute the isomorphic feature tensors based on the other kernels, which can be denoted as $\bar{\mathcal{Z}}_1, \bar{\mathcal{Z}}_2, \cdots, \bar{\mathcal{Z}}_c$ respectively.

2) Min-pooling Layer: Given the tensor $\bar{\mathcal{Z}}_i$ computed by $\mathbf{K}_i$ in the graph isomorphic layer, IsoNN will identify the optimal permutation matrices via the min-pooling layer. Formally, we can represent results of the optimal permutation selection with $\bar{\mathcal{Z}}_i$ as matrix $\mathbf{Z}_i$:

$$\mathbf{Z}_i(s,t) = \min\{\bar{\mathcal{Z}}_i(1:k!,s,t)\}. \tag{3}$$

The min-pooling layer learns the optimal matrix $\mathbf{Z}_i$ for kernel $\mathbf{K}_i$ along the first dimension (i.e., the dimension indexed by different permutation matrices), which can effectively identify the isomorphic features created by the optimal permutation matrices. For the remaining kernel matrices, we can also achieve their corresponding graph isomorphic feature matrices as $\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_c$ respectively.

3) Softmax Layer: Based on the above descriptions, an optimal matching between the subgraph templates with the input graph will lead to a very small isomorphic feature, e.g., a value approaching to 0. If we feed the small features into the classification component, the useful information will vanish and the relative useless information (*i.e.*, features learned by the subgraphs dismatch the kernels) dominates the learning feature vector in the end. Meanwhile, the feature values computed in Equation (3) can also be in different scales for different kernels. To effectively normalize these features and determine the most similar kernel for a subgraph, we propose to apply the softmax function to matrices $\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_c$ across all c kernels. Compared with the raw features, e.g., $\mathbf{Z}_i$, softmax as a non-linear mapping can also effectively highlight the useful features in $\mathbf{Z}_i$ by rescaling them to relatively larger values especially compared with the useless ones. Formally, we can represent the fused graph isomorphic features after rescaling by all the kernels as a 3-way tensor $\mathcal{Q}$, where slices along first dimension can be denoted as:

$$\hat{\mathcal{Q}}(i,:,:) = -\mathbf{Z}_i \tag{4}$$

$$\mathcal{Q}(i,s,t) = \frac{\exp(\hat{\mathcal{Q}}(i,s,t))}{\sum_{i=1}^{c}(\exp\left(\hat{\mathcal{Q}}(i,s,t)\right))} \tag{5}$$

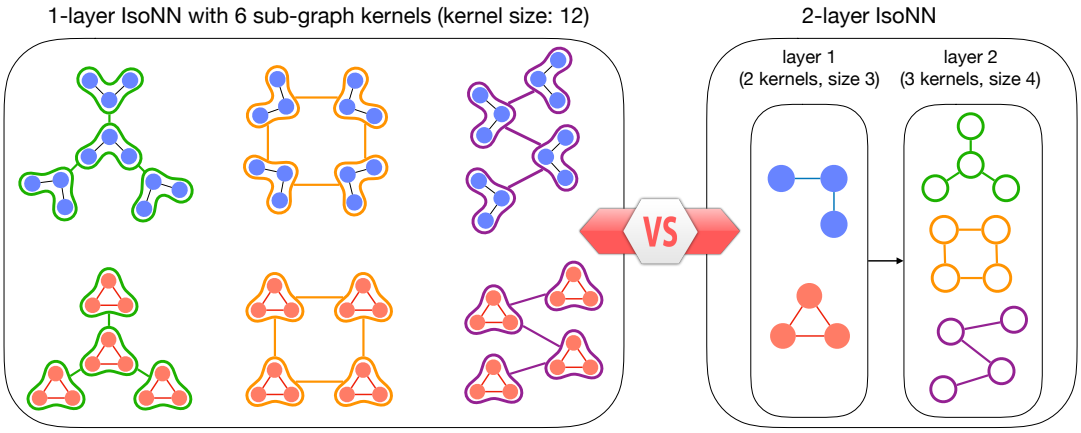


Fig. 2: An Illustration of Deep Architecture of IsoNN. (The plot provides possible subgraph templates for both one isomorphic layer with kernel size 12 and two isomorphic layers with kernel size 3 and 4, showing the similar representation power of single layer with large kernel size and the multiple layers with small kernel sizes.)

However, if $c = 1$, then the softmax normalization will lead to a constant feature matrix (*i.e.*, a constant matrix full of 1). To avoid it, we propose to normalize the feature matrix within itself. So the final features will be $\mathcal{Q}(1, :, :) = \text{softmax}(-\mathbf{Z}_1)$.

B. Classification Component

After the isomorphic feature tensor $\mathcal{Q}$ is obtained, we feed it into a classification component. Let $\mathbf{q}$ denote the flattened vector representation of feature tensor $\mathcal{Q}$ whose value are sorted in a descending order so as to handle the node orderless property for the input graph. We pass it to three fully-connected layers to get the predicted label vector $\hat{\mathbf{y}}$. For the graph binary classification, suppose we have the ground truth $\mathbf{y} = (y_1^g, y_2^g)$ and the predicted label vector $\hat{\mathbf{y}}^g = (\hat{y}_1^g, \hat{y}_2^g)$ for the sample g from the training batch set $\mathcal{B}$. We use cross-entropy as the loss function in IsoNN. Formally, the fully-connected (FC) layers and the objective function can be represented as follows respectively:

$$\text{FC Layers: } \begin{cases} \mathbf{d}_1 &= \sigma(\mathbf{W}_1\mathbf{q} + \mathbf{b}_1), \\ \mathbf{d}_2 &= \sigma(\mathbf{W}_2\mathbf{d}_1 + \mathbf{b}_2), \\ \hat{\mathbf{y}} &= \sigma(\mathbf{W}_3\mathbf{d}_2 + \mathbf{b}_3), \end{cases} \tag{6}$$

$$\text{Objective Function: } \mathcal{L} = -\sum_{g\in\mathcal{B}}\sum_{j=1}^{2} y_j^g \log \hat{y}_j^g, \tag{7}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ represent the weights and biases in i-th layer respectively for $i \in \{1, 2, 3\}$. The σ denotes the adopted the relu activation function. To train the proposed model, we adopt the back propagation algorithm to learn both the subgraph templates and the other involved variables.

C. More Discussions on Graph Isomorphic Feature Extraction in IsoNN

Before introducing the empirical experiments to test the effectiveness of IsoNN, we would like to provide more discussions about the computation time complexity of the graph isomorphic feature extraction component involved in IsoNN. Formally, given the input graph G with $n = |\mathcal{V}|$ nodes, by shifting the kernel variables (of size $k \times k$) among the dimensions of the corresponding graph adjacency matrix, we will be able to obtain $(n - k + 1)^2$ regional sub-matrices (or $\mathcal{O}(n^2)$ regional sub-matrices for notation simplicity). Here, we assume IsoNN has only one isomorphic layer involving c different kernels. In the forward propagation, the introduced time cost in computing the graph isomorphic features can be denoted as $\mathcal{O}(ck!k^3n^2)$, where term $k!$ is introduced in enumerating all the potential permutation matrices and k^3 corresponds to the matrix multiplication time cost.

According to the notation, we observe that n is fixed for the input graph. Once the kernel channel number c is decided, the time cost notation will be mainly dominated by k. To lower down the above time complexity notation, in this part, we propose to further improve IsoNN from two perspectives: (1) compute the optimal permutation matrix in a faster manner, and (2) use deeper model architectures with small-sized kernels.

1) Fast Permutation Matrix Computation: Instead of enumerating all the permutation matrices in the graph isomorphic feature extraction as indicated by Equations (2)-(3), here we introduce a fast way to compute the optimal permutation matrix for the provided kernel variable matrix, e.g., $\mathbf{K}_i$, and input regional sub-matrix, $\mathbf{M}_{(s,t)}$, directly according to the following theorem.

THEOREM *1:* Formally, let the kernel variable $\mathbf{K}_i$ and the input regional sub-matrix $\mathbf{M}_{(s,t)}$ be $k \times k$ real symmetric matrices with k distinct eigenvalues $\alpha_1 > \alpha_2 > \cdots > \alpha_k$ and $\beta_1 > \beta_2 > \cdots > \beta_k$, respectively, and their eigendecomposition be represented by

$$\mathbf{K}_i = \mathbf{U}_{K_i}\mathbf{\Lambda}_{K_i}\mathbf{U}_{K_i}^\top, \text{ and } \mathbf{M}_{(s,t)} = \mathbf{U}_{M_{(s,t)}}\mathbf{\Lambda}_{M_{(s,t)}}\mathbf{U}_{M_{(s,t)}}^\top \tag{8}$$

where $\mathbf{U}_{K_i}$ and $\mathbf{U}_{M_{(s,t)}}$ are orthogonal matrices of eigenvectors and $\mathbf{\Lambda}_{K_i} = diag(\alpha_j), \mathbf{\Lambda}_{M_{(s,t)}} = diag(\beta_j)$. The minimum of $||\mathbf{P}\mathbf{K}_i\mathbf{P}^\top - \mathbf{M}_{(s,t)}||^2$ is attained for the following $\mathbf{P}$'s:

$$\mathbf{P}^* = \mathbf{U}_{M_{(s,t)}}\mathbf{S}\mathbf{U}_{K_i}^\top \tag{9}$$

where $\mathbf{S} \in \mathcal{S} = \{diag(s_1, s_2, \cdots, s_k) | s_i = 1 \text{ or } -1\}$.

Before giving the proof of Theorem 1, we need to introduce Lemma 1 first.

LEMMA *1:* If $\mathbf{A}$ and $\mathbf{B}$ are Hermitian matrices with eigenvalues $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$ and $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$ respectively, then $||\mathbf{A}-\mathbf{B}|| \geq \sum_{i=1}^{n}(\alpha_i - \beta_i)^2$.
Based on Lemma 1, we can derive the proof of Theorem 1 as follows.

PROOF *1:* From Lemma 1, Equation 10 holds for any orthogonal matrix $\mathbf{R}$ since the eigenvalues of $\mathbf{R}\mathbf{K}_i\mathbf{R}^\top$ are the same as those of $\mathbf{K}_i$.

$$||\mathbf{R}\mathbf{K}_i\mathbf{R}^\top - \mathbf{M}_{(s,t)}||^2 \geq \sum_{j=1}^{n}(\alpha_j - \beta_j)^2 \tag{10}$$

On the other hand, if we use $\mathbf{P}$ in Equation (9), we have

$$\begin{aligned}
&||\mathbf{P}\mathbf{K}_i\mathbf{P}^\top - \mathbf{M}_{(s,t)}||^2 \\
&= ||\mathbf{U}_{M_{(s,t)}}\mathbf{S}\mathbf{U}_{K_i}^\top\mathbf{U}_{K_i}\mathbf{\Lambda}_{K_i}\mathbf{U}_{K_i}^\top\mathbf{U}_{K_i}\mathbf{S}\mathbf{U}_{M_{(s,t)}}^\top \\
&\quad - \mathbf{U}_{M_{(s,t)}}\mathbf{\Lambda}_{M_{(s,t)}}\mathbf{U}_{M_{(s,t)}}^\top||^2 \\
&= ||\mathbf{U}_{M_{(s,t)}}(\mathbf{S}\mathbf{\Lambda}_{K_i}\mathbf{S} - \mathbf{\Lambda}_{M_{(s,t)}})\mathbf{U}_{M_{(s,t)}}^\top||^2 \\
&= ||\mathbf{S}\mathbf{\Lambda}_{K_i}\mathbf{S} - \mathbf{\Lambda}_{M_{(s,t)}}||^2 \\
&= ||\mathbf{\Lambda}_{K_i} - \mathbf{\Lambda}_{M_{(s,t)}}||^2 \\
&= \textstyle\sum_{j=1}^{n}(\alpha_j - \beta_j)^2
\end{aligned}$$

where we use the equations that $||\mathbf{U}\mathbf{X}|| = ||\mathbf{U}\mathbf{X}^\top|| = ||\mathbf{X}||$ for any orthogonal matrix $\mathbf{U}$ and $\mathbf{S}\mathbf{\Lambda}_{K_i}\mathbf{S} = \mathbf{S}^2\mathbf{\Lambda}_{K_i} = \mathbf{\Lambda}_{K_i}$ since $\mathbf{S}$ and $\mathbf{\Lambda}_{K_i}$ are both orthogonal matrices and $\mathbf{S}^2 = \mathbf{I}$. Moreover, it is clear that

$$\mathrm{tr}(\mathbf{P}^\top\mathbf{U}_{M_{(s,t)}}\mathbf{S}\mathbf{U}_{K_i}^\top) \leq \mathrm{tr}\left(\mathbf{P}^\top|\mathbf{U}_{M_{(s,t)}}||\mathbf{U}_{K_i}^\top|\right) \tag{11}$$

because of the elements in $\mathbf{S}$ are either -1 or $+1$. Also, since each row vector of $\mathbf{U}_{M_{(s,t)}}$ and $\mathbf{U}_{K_i}$ is unit vector, thus we can have

$$\mathrm{tr}(\mathbf{P}^\top|\mathbf{U}_{M_{(s,t)}}||\mathbf{U}_{K_i}^\top|) \leq n \tag{12}$$

If there exists a perfect permutaion matrix, $\mathbf{P}^*$, then there exists such $\mathbf{S}^*$, s.t.

$$\mathrm{tr}(\mathbf{P}^{*\top}\mathbf{U}_{M_{(s,t)}}\mathbf{S}^*\mathbf{U}_{K_i}^\top) = \mathrm{tr}(\mathbf{P}^{*\top}\mathbf{P}^*) = n \tag{13}$$

Thus, based on Equation (11), Equation (12) and Equation (13), we can get

$$\mathrm{tr}(\mathbf{P}^{*\top}|\mathbf{U}_{M_{(s,t)}}||\mathbf{U}_{K_i}^\top|) \leq n. \tag{14}$$

This means that $\mathbf{P}$ maximizes $\mathrm{tr}(\mathbf{P}^\top|\mathbf{U}_{M_{(s,t)}}||\mathbf{U}_{K_i}^\top|)$. Therefore, when $\mathbf{K}_i$ and $\mathbf{M}_{(s,t)}$ are isomorphic, the optimal permutation matrix can be obtained as a permutation matrix $\mathbf{P}^*$ which maximizes the trace in Eq.14. Therefore, we take $\mathbf{P}^* = |\mathbf{U}_{\mathbf{M}_{(s,t)}}||\mathbf{U}_{K_i}^\top|$ directly as the approximated optimal permutation matrix instead, which together with the corresponding optimal feature $z_{i,(s,t)}$ can be denoted as follows:

$$\mathbf{P}^* = |\mathbf{U}_{\mathbf{M}_{(s,t)}}||\mathbf{U}_{K_i}^\top| \quad \text{and} \quad z_{i,(s,t)} = ||\mathbf{P}^*\mathbf{K}(\mathbf{P}^*)^\top - \mathbf{M}_{(s,t)}||^2, \tag{15}$$

where $|\cdot|$ denotes the absolute value operator and $|\mathbf{U}_{\mathbf{M}_{(s,t)}}||\mathbf{U}_{K_i}^\top| \geq \mathbf{U}_{M_{(s,t)}}\mathbf{S}\mathbf{U}_{K_i}^\top$ hold for $\forall\mathbf{S} \in \mathcal{S}$.

By replacing Equations (2)-(3) with Equation (15), we can compute the optimal graph isomorphic feature for the kernel $\mathbf{K}_i$ and input regional sub-matrix $\mathbf{M}_{(s,t)}$ with a much lower time cost. Furthermore, since the eigendecomposition time complexity of a $k \times k$ matrix is $\mathcal{O}(k^3)$, based on the above theorem, we will be able to lower down the total time cost in graph isomorphic feature extraction to $\mathcal{O}(ck^3n^2)$, which can be optimized with the method introduced in the following subsection.

2) Deep Graph Isomorphic Feature Extraction: Since graph isomorphic layer is the main functional layer, we simply use a multi-layer for short to denote the multiple graph isomorphic feature extraction components (*i.e.*, deep model). To ensure the deep model can still work well, we add residual module into the deep model [19]. Here, we will illustrate the advantages of deep ISONN model with small-sized kernels compared against shallow ISONN model with large kernels. In Figure 2, we provide an example two ISONN models with different model architectures

- the left model has one single layer and 6 kernels, where the kernel size $k = 12$;
- the right model has two layers: layer 1 involves 2 kernels of size 3, and layer 2 involves 3 kernels of size 4.

By comparing these two different models, we observe that they have identical representation learning capacity. However, the time cost in feature extraction introduced by the left model is much higher than that introduced by the right model, which can be denoted as $\mathcal{O}(6(12^3)n^2)$ and $\mathcal{O}(2(3^3)n^2 + 3(4^3)n^2)$, respectively.

Therefore, for the ISONN model, we tend to use small-sized kernels. Formally, according to the fast method provided in the previous part, given a 1-layer ISONN model with c large kernels of size k, its graph isomorphic feature extraction time complexity can be denoted as $\mathcal{O}(ck^3n^2)$. Inspired by Figure 2, without affecting the representation capacity, such a model can be replaced by a $\max\{\lceil\log_2^c\rceil, \left\lceil\log_3^k\right\rceil\}$-layers deep ISONN model instead, where each layer involves 2 kernels of size 3. The graph isomorphic feature extraction time complexity of the deep model will be $\mathcal{O}\big((\max\{\lceil\log_2^c\rceil, \left\lceil\log_3^k\right\rceil\}) \cdot 2 \cdot 3^3n^2\big)$ (or $\mathcal{O}\left((\max\{\lceil\log^c\rceil, \left\lceil\log^k\right\rceil\}) \cdot n^2\right)$ for simplicity).

IV. EXPERIMENTS

To evaluate the performance of ISONN, we will talk about the experimental settings as well as four brain graph datasets. Then, we will discuss the experimental results with parameter analyses on kernel size , channel number and layer number. Finally, we will study the time complexity and provide a case study for breaking the node order.

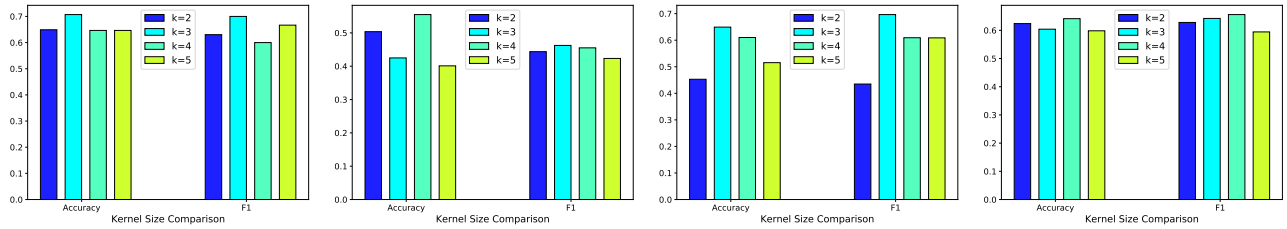
A. Experimental Settings

In this subsection, we will use four real-world benchmark datasets: HIV-fMRI [20], HIV-DTI [20], BP-fMRI [21], and ADHD-fMRI[1]. Both HIV-fMRI and HIV-DTI have 56 positive instances and 21 negative instances. Also, graph instances in both of them are represented as 90×90 matrices [20]. BP-fMRI has 52 positive and 45 negative instances and each instance is presented by an 82×82 matrix [21]. ADHD-fMRI dataset is a public dataset with size 116×116 per brain

[1]http://neurobureau.projects.nitrc.org/ADHD200

TABLE I: Classification Results of the Comparison Methods.

Dataset	Metric	Methods								
		Freq	AE	CNN	SDBN	WL	GCN	GIN	IsoNN-fast	IsoNN
HIV-fMRI	Accuracy	54.3	46.9	59.3	66.5	44.2	58.3	52.5	73.9	**76.5**
	F1	58.2	35.5	66.3	66.7	27.2	56.4	35.6	70.7	**75.9**
HIV-DTI	Accuracy	64.6	62.4	54.3	65.9	47.1	57.7	55.1	60.1	**67.5**
	F1	63.9	0.0	55.7	65.6	48.4	54.4	53.6	61.9	**72.3**
BP-fMRI	Accuracy	56.8	53.6	54.6	64.8	56.2	60.7	45.4	62.3	**64.9**
	F1	57.6	69.5	52.8	63.7	58.8	61.2	42.3	63.2	**69.7**
ADHD-fMRI	Accuracy	55.1	49.6	56.9	50.3	52.1	54.8	53.0	62.4	**64.1**
	F1	55.4	50.5	57.2	**66.8**	45.2	59.8	55.7	62.3	65.6



(a) HIV-fMRI (b) HIV-DTI (c) BP-fMRI (d) ADHD-fMRI

Fig. 3: Effectiveness of Different k

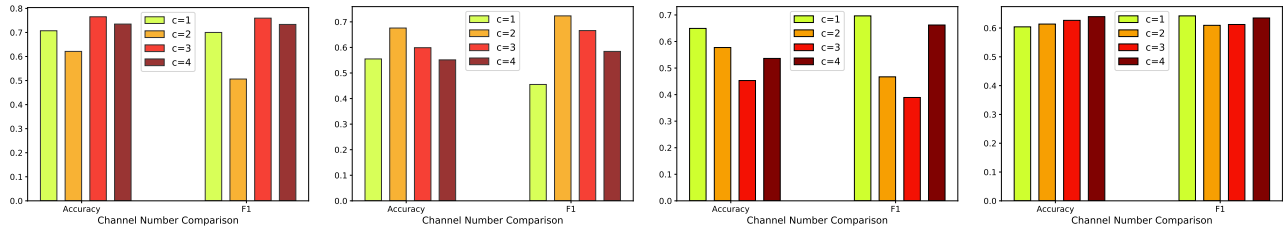
graph and has 776 brain graphs in total and in which 491 are negative. With three small datasets and one large dataset, we first introduce the comparison methods used in this paper and then talk about the experimental setups and the adopted evaluation metrics in detail.

1) Comparison Methods:

- **IsoNN & IsoNN-fast** : The proposed method IsoNN uses a set of template variables as well as the permutation matrices to extract the isomorphic features and feed these features to the classification component. The variant model named IsoNN-fast uses the Equation (15) to compute the optimal permutation matrices and other settings remain unchanged.
- **Freq**: The method uses the top-k frequent subgraphs as its features. This is also an unsupervised feature selection method based on frequency.
- **AE**: We use the autoencoder model (AE) [22] to get the features of graphs without label information. It is an unsupervised learning method, which learns the latent representations of connections in the brain graphs without considering the structural information.
- **CNN:** It is the convolutional model [23] learns the structural information within small regions of the whole image. While the graph is different from the image due to its 'node-orderless' property.
- **SDBN**: A model proposed in [12], which reorders the nodes in the brain graph first and then feeds the reordered graph into an augmented CNN. In this way, it not only learns the structural information but also tries to minimize the effect of the order constraint.
- **WL:** WL [15] is a classic algorithm to do the graph isomorphism test. Since brain graphs do not have node or edge labels, we simply assign each node different number as their node labels.
- **GCN[2]:** GCN is a classic graph neural network model proposed in [9]. It uses the adjacent matrix to learn the implicit structure information in graphs. However, GCN learns node embeddings instead of graph embedding. Here, to obtain the graph representation, we simply concatenate all node features.
- **GIN[2]:** GIN is proposed in [10] recently. GIN updates the node features via neighboring nodes and aggregate the node features as the graph features.

2) Experimental Setup and Evaluation Metrics: In our experiments, to make the results more reliable, we partition the datasets into 3 folds and then set the ratio of train/test according to $2:1$, where two folds are treated as the training data and the remaining one is the testing data. Here, for unsupervised methods, we use SVM classifier. For supervised models, we will use MLP classifier with three fully connected layers with 1024, 128, 2 neurons, respectively. We select top-100 features for Freq as stated in [12]. For Auto-encoder, we apply the two-layer encoder and two-layer decoder with 1024 and 128 neurons, respectively. For the CNN, we apply the one convolutional layer with the size $5\times5\times50$, a max-pooling layer with kernel size 2×2, one gating relu layer as an activation layer. For the SDBN, we set the architecture as follows: we use two layers of the "convolution layer + max-pooling layer + activation layer " and concatenate a fully connected layer with 100 neurons as well as an activation layer, where the parameters are the same as those in CNN. We also set the dropout rate in SDBN being 0.5 to avoid overfitting. For WL kernel, we first compute the pairwise similarity matrices for both training and testing sets and then use feed the training

[2] To make fair comparison, we use the adjacency matrix as the feature matrix, where each row is the feature vector for corresponding node.



(a) HIV-fMRI (b) HIV-DTI (c) BP-fMRI (d) ADHD-fMRI

Fig. 4: Effectiveness of Different c

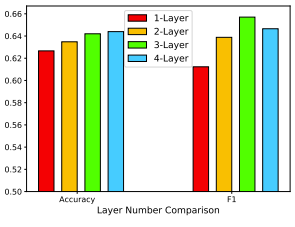


Fig. 5: Study on Layers

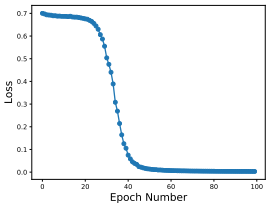


Fig. 6: Convergence

similarity matrix to train the SVM classifier. We use one graph convolutional layer to learn the node features and concatenate the node features as graph features. We follow the setting in [10] to do GIN-0. Here, to make a fair comparison, we will use the adjacency matrices as features (*i.e.*, no node label information) for WL, GCN, and GIN. In the experiments, we set the kernel size k in the isomorphic layer for three datasets as 3, 4, 3, 3, respectively, and then set the parameters in classification component the same as those in MLP classifier. To evaluate the performance, we use accuracy and F1 as the metrics. In this experiment, we adopt Adam optimizer and the set the learning rate $\eta = 0.001$, and run the model on Dell PowerEdge T630 Server with 2 20-core Intel CPUs and 256GB memory. Finally we will report the average results on balanced datasets.

B. Experimental Results

In this section, we investigate the effectiveness of the learned subgraph-based graph feature representations for graphs. We adopt one isomorphic layer where the kernel size $k = 3$ and channel number $c = 3$ for HIV-fMRI, one isomorphic layer with $(k = 4, c = 2)$, $(k = 3, c = 1)$, $(k = 3, c = 4)$ for the HIV-DTI, BP-fMRI and ADHD-fMRI respectively. The results are shown in Table I. From that table, we can observe that ISONN outperforms all other baseline methods on these all datasets. Compared with Freq, the proposed method achieves a better performance without searching for all possible subgraphs manually. AE has almost the worst performance among all comparison methods. This is because the features learned from AE do not contain any structural information. For HIV-DTI, AE gets 0 in F1. This is because the dataset contains too many zeros, which makes the AE learn trivial features. CNN performs better than AE but still worse than ISONN. The reason can be that it learns some structural information but fails to break the node order within subgraphs. One possible reason for WL gets bad results is the brain graphs do not have a useful pattern according to node labels. GCN performs better than GIN but worse than ISONN, showing that GCN can learn some structural information without node labels and node features. Comparing ISONN with AE, ISONN achieves better results. This means the structural information is more important than only connectivity information for the classification problem. If compared with CNN, the results also show the contribution of breaking the node-order in learning the subgraph templates. Similar to SDBN, ISONN also finds the features from subgraphs, but ISONN gets better performance with more concise architecture. While the SDBN has better F1 on ADHD-fMRI, its accuracy is low, thus the high F1 cannot be treated as a reliable result. Contrasting with GCN and GIN, ISONN can maintain the explicit subgraph structures in graph representations, while the GCN and GIN using the aggregation of the neighboring node features only learn the implicit structural infomation. Since the approximation on $\mathbf{P}$ impairs the performances, we choose different parameters for HIV-fMRI and HIV-DTI, which are $(k = 4, c = 4)$ and $(k = 4, c = 3)$, respectively. Generally, the performances of ISONN-fast are not the best for four datasets, but the performances are close to ISONN.

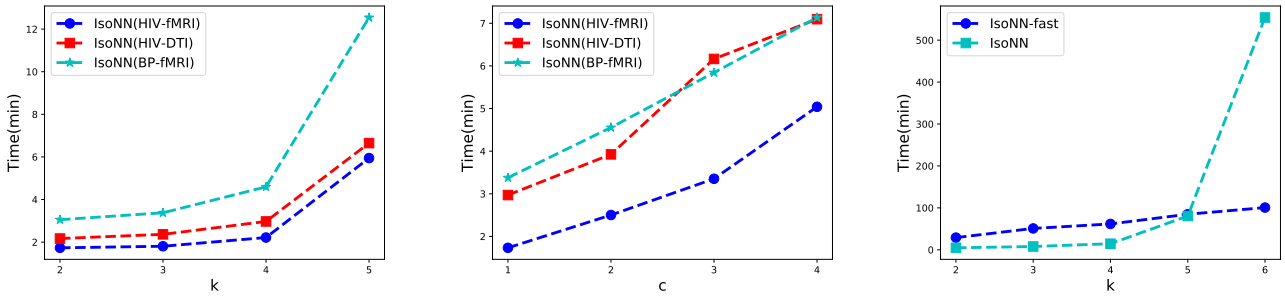
C. Convergence Analysis

The Figure 6 shows the convergence of ISONN, where the x-axis denotes the epoch number and the y-axis is the training loss, respectively. It illustrates that the proposed method can achieve a stable optimal solution within 50 iterations, which shows our method can converge with few epochs.

D. Parameter Analysis

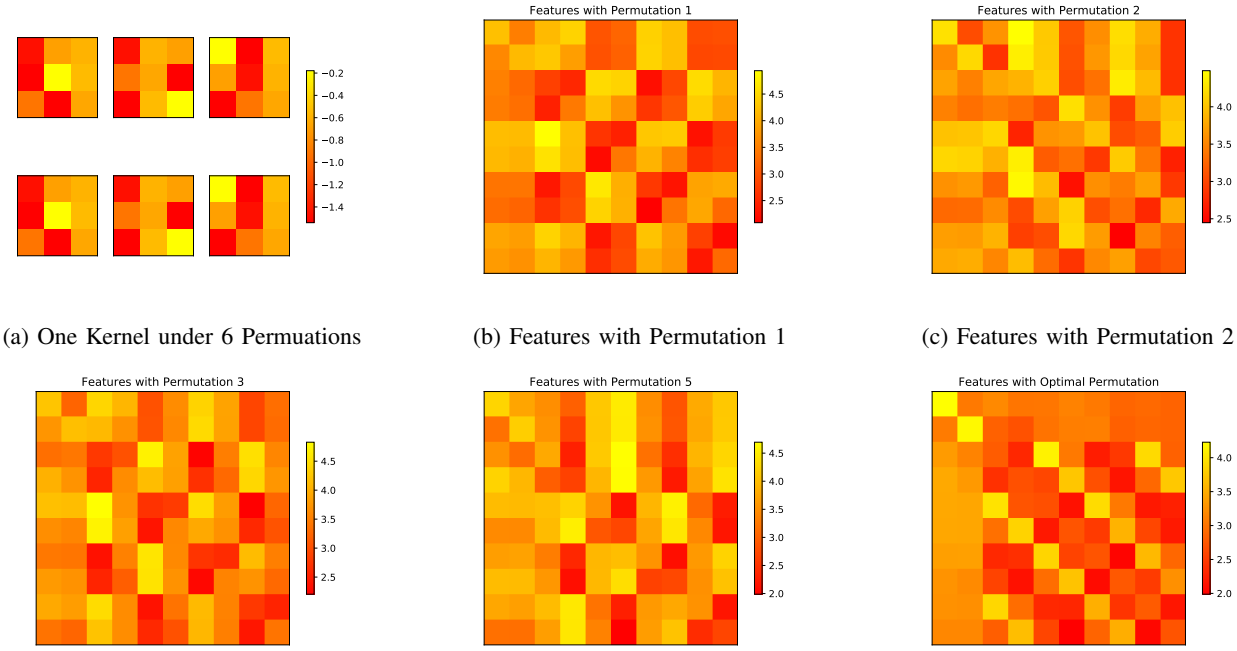
To further study the proposed method, we will discuss the effects of different kernel size and channel number in ISONN.

- **Kernel Size**: We show the effectiveness of different k in Figure 3. Based on the previous statement, parameter k can affect the final results since it controls the size of learned subgraph templates. To investigate the best kernel size for each dataset, we fix the channel number $c = 1$. As Figure 3 shows, different datasets have different appropriate kernel sizes. The best kernel sizes are 3, 4, 4, 3, for the three datasets, respectively.
- **Channel Number**: We also study the effectiveness of multiple channels (i.e., multiple templates in one layer). To discuss how the channel number influences the results,



(a) Different k (b) Different c (c) IsoNN & IsoNN-fast

Fig. 7: Time Complexity Study



(a) One Kernel under 6 Permuations (b) Features with Permutation 1 (c) Features with Permutation 2

(d) Features with Permutation 3 (e) Features with Permutation 5 (f) Features with Optimal Permutation

Fig. 8: Case Study on Breaking the Node Order. (Figure 8b-8e shows learned features under 4 permutations of one kernel in Figure 8a and the 4 features learned under the 4 permutations are totally different. Compared with features obtained by the optimal permutation shown in Figure 8f, features learned by a fixed permutation (*i.e.*, node order) cannot discover the same subgraphs in a real graph.)

we choose the kernel size for each dataset (i.e., 3, 4, 3, 3 respectively). From all sub-figures in Figure 4, we can see that the differences among the different channel numbers by using only one isomorphic layer. As shown in Figure 4, IsoNN achieves the best results by $c = 3, 2, 1, 4$, respectively, which means the increase of the channel number can improve the performance, but more channels do not necessarily lead to better results. The reason could be the more templates we use, the more complex our model would be, thus it loses the model generalization.

- **Layer Number**: Figure 5 shows the performance with different layers on ADHD-fMRI dataset. From the figure, we can see that when we apply more layers, the accuracy and F1 slightly increase as the number of layers goes up to 3. However, the performance of 4 layers goes down. It indicates the big subgraph can be beneficial but also has the probability of being overfitting since the big subgraph captures abstract global patterns instead of local patterns.

E. Time Complexity Study

To study the efficiency of IsoNN and IsoNN-fast, we collect the actual running time on the training model, which is shown in Figure 7. In both Figures 7a and 7b [3], the x-

[3]Since the ADHD-fMRI is a relatively large dataset compared with the others, its running time is in different scale compared with the other datasets, which makes the time growth curve of other datasets not obvious. Thus, we don't show the results on ADHD-fMRI.

axis denotes its value for k or c and the y-axis denotes the time cost with different parameters. From Figure 7a, three lines show the same pattern. When the k increases, the time cost grows exponentially. This pattern can be directly explained by the size of the permutation matrix set. When we increase the kernel size by one, the number of corresponding permutation matrices grows exponentially. While changing c, shown in Figure 7b, it is easy to observe that those curves are basically linear with different slopes. This is also natural since whenever we add one channel, we only need to add a constant number of the permutation matrices. To study the efficiency of ISONN-fast, Figure 7c shows the running times of ISONN and ISONN-fast on ADHD-fMRI. As it shows, ISONN-fast uses less time when the kernel size greater than 5, otherwise ISONN and ISONN will have little difference since the eigendecomposition has nearly the same time complexity as calculating all possible node permutations.

F. Case Study on Breaking the Node Order

To study the node order broken by ISONN, we provide a case study. Here, Figure 8 shows the one kernel and features obtained by setting the $k = 3, c = 4$ for one-layer ISONN on ADHD-fMRI dataset. To better illustrate the idea, we only use part of the whole brain graph and the window size is 10×10. Figure 8a shows one of four kernels learned by ISONN under all 6 permutations. From it, we can see that one kernel template can be presented into 6 different matrices, which also shows the different node-order results in different matrix representations. By moving each kernel matrices on the small window of the original brain graph, we compute the distances between subgraphs and the kernel under all permutations as well as maintain the minimal distances with optimal permutations. Due to the space limit, we only show four out of six permutations, which are illustrated in Figure 8b, 8c, 8d, 8e. From Figure 8b-8e, we can see that the learned features under different permutation of the same kernel are totally different, which shows different node orders on small kernels can lead to big differences on the whole brain graph. Figure 8f shows the features under the optimal permutation for subgraphs. The colors of the grids on left side of Figure 8f is almost the same, it indicates the subgraphs should be the same subgraph, however, in Figure 8b-8e, the same grids have different colors, showing that the node order existed in matrices cannot discover the same subgraphs in a real graph.

V. RELATED WORK

Our work relates to subgraph mining and graph kernel methods, graph neural networks, network embedding as well as brain graph analysis.

Subgraph Mining and Graph Kernel Methods: Mining subgraph features from graph data have been studied for many years. The aim is to extract useful subgraph features from a set of graphs by adopting some specific criteria. One classic unsupervised method (i.e., without label information) is gSpan [24], which built a lexicographic order among graphs and map each graph to a unique minimum DFS code as its canonical label. For the supervised model (i.e., with label information), CORK utilized labels to guide the feature selection, where the features were generated by gSpan [25]. Due to the mature development of the sub-graph mining field, subgraph mining methods have also been adopted in life sciences [26]. On the other hand, graph kernel methods are also applied to discover the subgraph structures [14], [15]. Among them, most existing works focus on the graph with node labels and the kernels need to be predefined. Yet, in this paper, we are handling the graph without node labels. Moreover, we can not only compute the similarity between pairwise graphs but also learn subgraph templates, which can be further analyzed.

Graph Neural Network and Graph Classification: Graph Neural Networks [9], [28]–[30] have been studied in recent years because of the prosperity of deep learning. Traditional deep models cannot be directly applied to graphs due to the special data structure. The GCN proposed in [9] utilized the normalized adjacency matrix to learn the node features for node classification. However, these existing works based on graph neural networks all fail to investigate the node-orderless property of the graph data and to maintain the explicit structural information. Another important topic related to this paper is graph classification. Initially, researchers mine the subgraphs by DFS or BFS [4], [31], and use them as the features. With the rapid development of deep learning (DL), many works are done based on DL methods. GAM builds the model by RNN with self-attention mechanism [32]. DCNN extend CNN to general graph-structured data by introducing a 'diffusion-convolution' operation [28]. Moreover, many works are proposed based on GNNs, too, where they often add an additional graph-pooling layer to get the graph representation [10], [33]–[35]. For example, GIN [10] proposes a general pooling function that simply takes the sum of the embeddings of all nodes. SAGPool [35] suggests the nodes have varying importance towards graph embeddings. However, these methods still neglect the sturcture naturally since they are often built with node features.

Brain Graph Analysis: Due to the development of brain-based graph theory, the brain can be analyzed from the graph perspective. Jie *et al.* studied both local connectivity and glocal topology of fMRI brain graphs by graph kernels [36]. Kong *et al.* proposed a discriminative criterion to select subgraphs [4]. Wang *et al.* first reordered the graph and used three ways of the decoder to predict brain-related diseases [12]. One of the early works applied GCN to predict Autism Spectrum Disorder and Alzheimer's disease [2]. InceptionGCN [37] defined a geometric 'inception modules' that capture intra- and inter-graph structural heterogeneity. Li *et al.* used mutual information to improve the performance of GNN [1]. Other works also focus on the interpretability of brain graphs. GroupINN [3] learned te node grouping and extract features jointly and the grouped nodes can provide good interpretability. Yang *et al.* proposed an edge-weighted graph attention network to better understand the roots of the bipolar disorder [38].

VI. CONCLUSION

In this paper, we proposed a novel graph neural network named ISONN to solve the brain graph classification problem.

ISONN consists of two components: (1) isomorphic component, where a set of permutation matrices is used to break the randomness order posed by matrix representation for a bunch of templates and one min-pooling layer and one softmax layer are used to get the best isomorphic features, and (2) classification component, which contains three fully-connected layers. We further discuss the two efficient variants of ISONN to accelerate the model. Next, we perform the experiments on four real-world brain datasets. The experimental results show the proposed method outperforms all comparison methods, which demonstrates the superiority of our proposed method. The experimental analysis on time complexity illustrates the efficiency of the ISONN-fast.

REFERENCES

[1] X. Li, N. C. Dvornek, J. Zhuang, P. Ventola, and J. Duncana, "Graph embedding using infomax for asd classification and brain functional difference detection," *arXiv preprint arXiv:1908.04769*, 2019.

[2] S. Parisot, S. I. Ktena, E. Ferrante, M. Lee, R. Guerrero, B. Glocker, and D. Rueckert, "Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer's disease," *Medical image analysis*, vol. 48, pp. 117–130, 2018.

[3] Y. Yan, J. Zhu, M. Duda, E. Solarz, C. Sripada, and D. Koutra, "Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 772–782.

[4] X. Kong, P. S. Yu, X. Wang, and A. B. Ragin, "Discriminative feature selection for uncertain graph classification," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 82–93.

[5] X. Kong and P. S. Yu, "Semi-supervised feature selection for graph classification," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 793–802.

[6] N. Jin, C. Young, and W. Wang, "Graph classification based on pattern co-occurrence," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 573–582.

[7] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE transactions on cybernetics*, vol. 45, no. 3, pp. 416–429, 2014.

[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[12] S. Wang, L. He, B. Cao, C.-T. Lu, P. S. Yu, and A. B. Ragin, "Structural deep brain network mining," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 475–484.

[13] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[14] B. Gaüzere, L. Brun, and D. Villemin, "Two new graphs kernels in chemoinformatics," *Pattern Recognition Letters*, vol. 33, no. 15, pp. 2038–2047, 2012.

[15] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[17] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[18] Y.-A. Lai, C.-C. Hsu, W. H. Chen, M.-Y. Yeh, and S.-D. Lin, "Prune: Preserving proximity and global ranking for network embedding," in *Advances in neural information processing systems*, 2017, pp. 5257–5266.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] B. Cao, X. Kong, J. Zhang, S. Y. Philip, and A. B. Ragin, "Identifying hiv-induced subgraph patterns in brain networks with side information," *Brain informatics*, vol. 2, no. 4, pp. 211–223, 2015.

[21] B. Cao, L. Zhan, X. Kong, S. Y. Philip, N. Vizueta, L. L. Altshuler, and A. D. Leow, "Identification of discriminative subgraph patterns in fmri brain networks in bipolar affective disorder," in *International Conference on Brain Informatics and Health*. Springer, 2015, pp. 105–114.

[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[24] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE, 2002, pp. 721–724.

[25] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 2009, pp. 1076–1087.

[26] A. Mrzic, P. Meysman, W. Bittremieux, P. Moris, B. Cule, B. Goethals, and K. Laukens, "Grasping frequent subgraph mining for bioinformatics applications," *BioData mining*, vol. 11, no. 1, p. 20, 2018.

[27] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[28] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[29] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.

[30] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[31] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gboost: a mathematical programming approach to graph classification and regression," *Machine Learning*, vol. 75, no. 1, pp. 69–89, 2009.

[32] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1666–1674.

[33] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *International conference on machine learning*. PMLR, 2020, pp. 874–883.

[34] H. Yuan and S. Ji, "Structpool: Structured graph pooling via conditional random fields," in *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[35] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*. PMLR, 2019, pp. 3734–3743.

[36] B. Jie, D. Zhang, W. Gao, Q. Wang, C.-Y. Wee, and D. Shen, "Integration of network topological and connectivity properties for neuroimaging classification," *IEEE transactions on biomedical engineering*, vol. 61, no. 2, pp. 576–589, 2013.

[37] A. Kazi, S. Shekarforoush, S. A. Krishna, H. Burwinkel, G. Vivar, K. Kortüm, S.-A. Ahmadi, S. Albarqouni, and N. Navab, "Inceptiongcn: receptive field aware graph convolutional network for disease prediction," in *International Conference on Information Processing in Medical Imaging*. Springer, 2019, pp. 73–85.

[38] H. Yang, X. Li, Y. Wu, S. Li, S. Lu, J. S. Duncan, J. C. Gee, and S. Gu, "Interpretable multimodality embedding of cerebral cortex using attention graph network for identifying bipolar disorder," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 799–807.