*mathematics*

MDPI

*Article*

# Improving Unsupervised Network Alignment with Matched Neighborhood Consistency

**Yan Li [1,2], Lei Zhang [1,2] and Feng Qian [1,2,*]**

[1] School of Mathematics and Computer Science, Tongling University, Tongling 244061, China; 242120@tlu.edu.cn (Y.L.); 267522@tlu.edu.cn (L.Z.)
[2] Anhui Engineering Research Center of Intelligent Manufacturing of Copper-Based Materials, Tongling 244061, China
* Correspondence: 020515@tlu.edu.cn

**Abstract:** Network alignment is an important technique with applications in diverse domains, such as social network analysis, bioinformatics, and knowledge graph construction. Many of the alignment methods rely on predefined anchor nodes, which are often unavailable in real-world scenarios. To overcome this limitation, we propose MANNA (MAtched Neighbor consistency for Network Alignment), an unsupervised approach to network alignment that exploits the concept of Matched Neighborhood Consistency (MNC). The hypothesis of MANNA is that nodes with higher similarity within their local neighborhood structures are more likely to be aligned across different networks. To learn the structural and attribute features of networks, MANNA uses a Graph Neural Network (GNN). It extracts multi-order node embeddings to capture multi-scale neighborhood features, which are then used to construct similarity matrices for the alignment process. MANNA introduces a key innovation by using pseudo-anchor nodes identified by the MNC strategy to provide self-supervised learning signals in the absence of real anchor nodes. This approach enhances the model's ability to learn accurate network representations and improve alignment accuracy. Alignment results are iteratively refined by applying the MNC strategy, which strengthens the consistency of neighborhood structures between matched nodes. Extensive experiments on three public datasets show that MANNA outperforms existing network alignment methods.

**Keywords:** network alignment; graph neural network; matched neighborhood consistency; pseudo-anchor nodes; self-supervised learning

**MSC:** 68T01



**Citation:** Li, Y.; Zhang, L.; Qian, F. Improving Unsupervised Network Alignment with Matched Neighborhood Consistency. *Mathematics* **2024**, *12*, 1211. https://doi.org/10.3390/math12081211

Academic Editor: Paolo Crippa

Received: 26 February 2024
Revised: 2 April 2024
Accepted: 16 April 2024
Published: 17 April 2024



## 1. Introduction

Network alignment, also known as graph alignment, is a technique used to identify and match nodes that represent the same entity or role in different networks [1]. This technique has various applications in multiple domains. For instance, in social network analysis, network alignment can identify the accounts of the same user on different platforms and enable friend recommendation [2]. In bioinformatics, identifying and comparing similar biological functions among different species can promote the transfer of functional knowledge across species [3]. Aligning the gene co-expression network of patients provides a scientific basis for precise treatment [4]. Furthermore, network alignment can facilitate the construction and integration of knowledge graphs by combining knowledge from different sources to enhance the completeness of knowledge graphs [5,6].

Most traditional network alignment algorithms rely on heuristic methods or optimization strategies and perform well on small to medium-sized networks [1,7]. However, as networks become larger and more structurally complex, these conventional methods face significant challenges in terms of computational efficiency and scalability. To overcome these challenges, embedding-based network alignment techniques have attracted much

attention from the academic community in recent years. These techniques can effectively handle more complex network structures and large datasets. Their main advantage is that they map network nodes into a low-dimensional vector space, and these representations (node embeddings) aim to preserve the topological structure and node attributes of the network [8]. Through this mapping, nodes representing the same entity can have similar representations even in different networks, thus simplifying the task of node correspondence identification. Embedding-based network alignment methods are mainly divided into two categories: one is to learn the vector representations of nodes by random walk or by directly exploiting the network structure, such as REGAL [9], NAWAL [10], PALE [11], and so on. These methods typically adopt a shallow neural network model and optimize the loss function to ensure that the nodes with similar structures in the network also maintain consistency in the vector representation. The other type of method captures the high-level structure and semantic information of the network more deeply through a Graph Neural Network (GNN), such as GAlign [12], WAlign [13], and CCNE [14]. These models use a multi-layer neural network structure, where each layer can effectively aggregate information from neighboring nodes to learn more robust node embeddings. This not only improves the accuracy of the alignment, but also increases the adaptability of the model to the complexity of the network structure.

Although network embedding technology has remarkable advantages in feature extraction and representation, which have greatly advanced research in network alignment, this field still faces several limitations. First, the sensitivity of network alignment methods to structural noise is a critical issue. Since the node representation is highly dependent on the network topology, noise or abnormal connections in the network may be erroneously incorporated during the model learning process and amplify these effects, which in turn negatively affects the accuracy of the alignment results. Second, some models may require known node correspondences (anchor nodes) to aid the learning process during the training phase. However, in real-world applications, it is often difficult to collect a large number of anchor nodes, and this reliance on prior knowledge can limit the applicability of the model.

The concept of Matched Neighborhood Consistency (MNC) [7,15] has gained attention as a key principle in enhancing network alignment. MNC focuses on the similarity between the neighborhoods of nodes in different networks, positing that nodes with similar local structures are more likely to represent the same entity. This approach aims to improve network alignment by considering the similarity of local structures rather than global network properties. Although MNC has potential, its application in unsupervised network alignment remains an open research question. Most existing methods do not fully exploit the power of MNC to enhance alignment accuracy, particularly in the absence of anchor nodes.

This paper proposes an unsupervised network alignment method called MANNA (MAtched Neighbor consistency for Network Alignment) to address the above challenges. Our approach employs a GNN model to capture network structure and attributes, and trains the GNN in an unsupervised manner. To ensure consistency in node embeddings between the source and target networks, we use a weight-sharing mechanism during training. Furthermore, to improve the model's robustness to noise, a random perturbation strategy is implemented. The GNN produces multi-order node embeddings from different layers, which can capture the nodes' multi-scale features. These multi-order node embeddings are used to construct the alignment matrix. High-quality pseudo-anchor nodes are identified based on MNC using the alignment matrix. These anchor nodes are used to refine the GNN model and optimize the network representation. In the final step, we refine the alignment matrix through a fine-tuning process, which reinforces the consistency of the neighborhood structures between corresponding nodes and improves alignment accuracy.

The main contributions of our study are as follows:

(1) We propose an unsupervised network alignment method that significantly reduces the model's sensitivity to network noise by introducing a random perturbation strategy. In addition, we use a self-supervised learning approach to optimize network

representations, thereby mitigating the excessive dependence on anchor nodes in existing methods;

(2) Following the MNC principle, we adopt two optimization strategies. First, we carefully select high-confidence pseudo-anchor nodes by analyzing the neighborhood similarity of cross-network node matches, which are used in subsequent self-supervised learning processes. Second, we apply an iterative update method to refine the alignment matrix, increasing the similarity between nodes with similar neighborhood structures across the two networks, thereby further improving the accuracy of network alignment;

(3) We performed experimental evaluations on three publicly available datasets. The results show that the proposed method is clearly superior to the most advanced comparison method, and its validity is effectively verified;

The remainder of this manuscript is structured as follows: Section 2 introduces several current embedding-based network alignment methods; Section 3 discusses the MANNA in more detail; in Section 4, experiments are performed to prove the effectiveness of the MANNA; finally, the research content of the manuscript is summarized.

## 2. Related Work

Embedding-based network alignment methods typically consist of the following steps:

Network Embedding: Network embedding is the process of mapping each node in the network to a low-dimensional vector space using a network-embedding algorithm such as Node2Vec [16] or LINE [17], or a graph neural network such as GCN [18] or GAT [19]. The objective of this process is to capture the structural information of the nodes. If the network contains node attribute information, these attributes are also considered during the embedding process to generate richer node embeddings.

Embedding Space Construction: After the embedding process, each node in the network is assigned a vector representation. Nodes with similar embeddings are positioned closer to each other in this embedding space. However, direct node matching can lead to incorrect matches if the embedding spaces of different networks are learned independently. This is because the embedding spaces of different networks may have different distributions, which can cause the positions of similar nodes in the embedding space to not align. In this case, it is necessary to align the embedding spaces to ensure that matching nodes from different networks maintain similar feature representations in the final embedding space.

Node Matching: In the embedding space, the similarity between nodes can be evaluated by calculating the similarity between node embedding vectors (e.g., cosine similarity). This allows for the identification of corresponding node pairs in two networks. This process typically involves finding the most similar pair of nodes in the embedding space or employing more sophisticated matching strategies.

Some works use anchor nodes to align embedding spaces. PALE [11] proposes a two-stage framework that consists of embedding and matching. In the embedding phase, PALE maps the networks to a low-dimensional space. In the matching phase, PALE ensures that anchor nodes remain close in the embedding space by learning a mapping function. To optimize the mapping process, PALE employs a loss function based on anchor nodes to minimize the difference in node embeddings before and after mapping. PALE provides two options for mapping functions: linear mapping and multi-layer perceptron (MLP). RAN [20] is similar to PALE as it also uses anchor nodes to reconcile the embedding spaces of the source and target networks. To enhance node representation, RAN generates a two-layer network consisting of a structure layer and an attribute layer. Biased random walk and language modelling techniques are then used to learn the node embedding. In contrast, CCNE [14] uses a collaborative optimization strategy instead of relying on complex mapping functions. CCNE employs two separate encoders to obtain node embeddings and reconstruct the original networks using a shared decoder. The objective function comprises intra-network and inter-network losses. Technical term abbreviations are explained when first used. The intra-network loss aims to minimize the reconstruction error to preserve the

internal structure of the network. Simultaneously, the inter-network loss aims to minimize the distance between anchor nodes to preserve the inter-network structure. The encoder and decoder parameters are updated through back-propagation during training. The two losses work together to gradually bring the embedding space closer. Additionally, CCNE employs a hard negative sampling strategy to maintain distance between anchor nodes and non-anchor nodes (hard negative samples) in the latent space.

Obtaining anchor nodes in the real world can be difficult, making it challenging to align the embedding space under unsupervised conditions. NAWAL [10] uses Generative Adversarial Networks (GAN) [21] to learn the correspondence between networks adaptively. After embedding learning, NAWAL employs GAN to initialize the mapping function, aligning the embedding space of the source network with that of the target network. The aligner generates mappings to match nodes, while the discriminator distinguishes real from generated embeddings. NAWAL selects the most similar pair of nodes as the initial anchor and iteratively optimizes the mapping function, minimizing the embedding difference using the Procrustes solution until the optimal mapping is found. Similarly, WAlign [13] optimizes node correspondences through self-supervised learning without anchors. This transforms the graph alignment problem into minimizing the Wasserstein distance in the embedding space. The Wasserstein distance discriminator is a neural network that optimizes embeddings through an adversarial training framework. Its goal is to maximize the similarity of pairs of similar nodes by generating pseudo-anchors. These pseudo-anchors are defined automatically based on the Wasserstein distance principle and updated iteratively to reflect the best node correspondences. REGAL [9] uses implicit matrix factorization to learn node embeddings without relying directly on the network adjacency structure. It generates identity information for each node by constructing a similarity matrix that combines structure and attribute information. The embedding representation is learned by computing its similarity with randomly selected landmark nodes. This approach provides a network alignment solution without explicit anchors.

Parameter sharing is a crucial technique in neural networks that enables the model to reuse the same weights for different inputs, rather than learning a new set of weights for each input. This strategy is especially significant in network alignment, as it guarantees that corresponding nodes in the source and target networks can share a consistent embedding representation. By utilizing parameter sharing, GAlign [12] is able to effectively map the nodes of both the source and target networks into a shared embedding space without the need for labeled data. To achieve this, GAlign employs the multi-order characteristics of the Graph Convolutional Network (GCN) [18] to capture node information across different neighborhood structures at varying levels of embedding. This method of multi-order embedding enables the model to comprehend the global network structure while retaining the local structure information, resulting in a more profound comprehension of network alignment. GAlign enhances the resilience of alignment outcomes by simulating noise and inconsistency in actual networks through data augmentation and noise adaptation loss functions. GATAL [22] also utilizes the parameter-sharing strategy. Unlike GAlign, GATAL adopts the multi-layer architecture of the Graph Attention Network (GAT) [19]. GAT assigns different weights to each node through the attention mechanism, enabling GATAL to capture the complex relationships between nodes more effectively.

Inferring node correspondences solely based on network structure and attribute information is a challenging task when explicit anchor links are absent. To enhance alignment accuracy, researchers have investigated various strategies. For instance, Grad-Align [23] utilizes GNN to acquire low-dimensional representations of nodes, capturing both structural and attribute information from the networks. The approach adopts a gradual alignment strategy to iteratively identify and match node pairs across networks, leveraging information from previously aligned nodes to guide subsequent matches. In addition, Grad-Align utilizes the Tversky similarity measure to align networks with different scales, thereby improving the accuracy of the alignment process. Grad-Align+ [24] also introduces node feature enhancement technology to enrich the node feature representation by calculating

the centrality of nodes. SANA [25], on the other hand, continuously optimizes the similarity matrix by using the topological structure similarity of the source and target networks through an iterative refinement process to improve the alignment quality. Other methods, such as NAME [26], capture the first-order proximity, high-order proximity and global network structure information of nodes by constructing shallow, deep and community-based embedding, and fuse these embeddings to improve alignment accuracy. By defining high-order topological consistency based on edge orbits, HTC [27] not only considers direct neighbors, but also pays attention to the structural role of nodes in the network, and integrates this consistency into the information aggregation process of GCN to identify and align corresponding nodes more accurately. MINING [28] adopts the multi-granularity alignment framework and contrastive learning strategy, decomposes the large network into sub-networks of different granularities, and combines the graph SAGE encoder to learn discriminative node features to achieve more accurate alignment at both coarse and fine granularities. Together, these methods have advanced the state of the art in network alignment techniques, enabling models to learn important information from unlabeled data to aid alignment.

Although GNN has made remarkable progress in the field of network alignment, it still faces some challenges, including sensitivity to noise and dependence on anchor nodes. To overcome these problems, this paper proposes an unsupervised network alignment method combining GNN and self-supervised learning, which aims to improve the accuracy and robustness of the alignment, especially when noisy data and anchor nodes are scarce. The core innovation of this method is the use of pseudo-anchor links for self-supervised learning. This method allows the model to learn effectively in the absence of real anchor nodes by introducing pseudo-anchor nodes in the source and target networks. As a regularization factor for model training, the pseudo-anchor nodes encourage the model to learn a better network representation and ensure that the embedding of the pseudo-anchor nodes in the source and target networks is as similar as possible.

## 3. Proposed Method

This chapter provides a detailed description of the MANNA framework, as shown in Figure 1. The framework includes the following main sequential steps. First, a GNN is initialized to capture the structural and attribute features of the networks. To enhance the model's robustness to noise, the algorithm introduces controlled structural and attribute perturbations during the training phase, ensuring that the learned representations are not overly sensitive to network imperfections. The GNN is used to generate multi-order node embeddings that capture the multi-scale neighborhood characteristics of each node. This provides a comprehensive representation of space for the alignment process. The algorithm applies the MNC principle to identify pseudo-anchor nodes from the similarity matrices derived. These pseudo-anchor nodes act as self-supervised guidance, enabling the model to learn more accurate network representations. The algorithm improves the alignment matrix by iteratively applying the MNC strategy. This strategy adjusts the similarity scores between pairs of nodes to strengthen the consistency of the neighborhood structures between aligned nodes. The iterative process is governed by a termination condition to optimize the alignment quality without over-fitting to noise.

### *3.1. Notation and Problem Definition*

Consider two networks, $G_s = (V_s, E_s, \mathbf{X}_s)$ and $G_t = (V_t, E_t, \mathbf{X}_t)$, where $G_s$ is the source network and $G_t$ is the target network. To simplify the presentation, we use an asterisk (*) as a wildcard to represent either the source or the target network, without specifying which one. Let $V_*$ and $E_*$ denote the sets of nodes and edges, respectively, within the network. The attribute matrix $\mathbf{X}_* \in \mathbb{R}^{|V_*| \times f}$ provides the initial feature for each node, where $f$ is the dimension of the feature. In this paper, we assume that both networks are undirected and unweighted, and we use the adjacency matrix $\mathbf{A}_* \in \mathbb{R}^{|V_*| \times |V_*|}$ to represent the connectivity relationships between nodes. In the adjacency matrix, the element $A_*(i, j)$ describes the

relationship between nodes $i$ and $j$. Specifically, if there is a direct edge between nodes $i$ and $j$, then $A_*(i,j) = A_*(j,i) = 1$; otherwise, $A_*(i,j) = A_*(j,i) = 0$. We denote by $N(i)$ the set of first-order neighbors of node $i$ in the adjacency matrix $\mathbf{A}_*$ of the network.

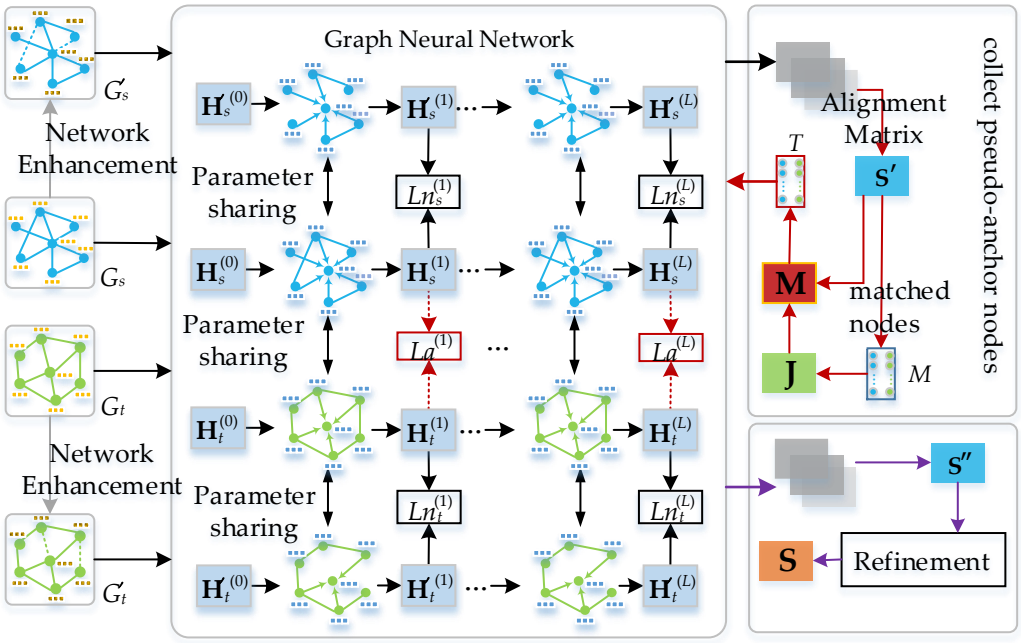


**Figure 1.** Overview of MANNA framework.

The objective of network alignment is to obtain a mapping matrix (alignment matrix) $\mathbf{S} \in \mathbb{R}^{|V_s| \times |V_t|}$, where the element $S(i,j)$ represents the similarity between node $i$ in the source network and node $j$ in the target network. The values of $S(i,j)$ typically satisfy the following criteria:

Non-negativity: $S(i,j) \geq 0$, since similarity scores are inherently non-negative.

Normalization: For each row $i$, the sum of all elements equals 1, i.e., $\sum_{j=1}^{|V_t|} S(i,j) = 1$, indicating that each node in the source network is aligned with, at most, one node in the target network.

Matched Neighborhood Consistency (MNC) [7,15,25] is a key concept in network alignment that measures the similarity between the sets of neighbors of two nodes that correspond to the same entity in different networks. This similarity can be quantified by calculating the similarity of the neighbors of the nodes.

In this paper, we use the node neighborhood similarity matrix $\mathbf{J} \in \mathbb{R}^{|V_s| \times |V_t|}$ to measure the similarity between pairs of nodes in two networks. Here, $J(u,v)$ denotes the neighborhood similarity between node $u$ in $G_s$ and node $v$ in $G_t$. Given a set of node mappings $M = \{(u,v) | u \in V_s, v \in V_t\}$, the formula for computing $J(u,v)$ is as follows:

$$J(u,v) = \frac{|N(u) \cap N'(v)|}{|N(u) \cup N'(v)|} \tag{1}$$

$$N'(v) = \{M(v') | v' \in N(v) \wedge v = M(u)\} \tag{2}$$

Figure 2 displays two distinct node mappings. For instance, we use the node pair $(a, 1)$ to demonstrate how MNC is employed to assess the alignment quality of the node pair. In this scenario, $N(a)$ of node $a$ in the source network is $\{b, c, d, e, f, g\}$, and $N(1)$ of node 1 in the target network is $\{2, 4, 5, 7\}$. In Figure 2a, $M = \{a:1, b:2, c:3, d:4, e:5, f:6, g:7\}$, $N'(a)$ of node 1 is $\{b, d, e, g\}$. Consequently, the value of $J(a,1) = 4/6 \approx 0.67$. In Figure 2b, $M = \{a:2, b:1, c:3, d:4, e:5, f:6, g:7\}$, the mapped neighborhood set $N'(a)$ for node 1 is $\{a, d, e, g\}$, and $J(a,1) = 3/6 = 0.5$. It is clear that a higher score indicates a more similar

neighborhood structure between node $a$ in the source network and node 1 in the target network under different mappings, which typically indicates better alignment quality.

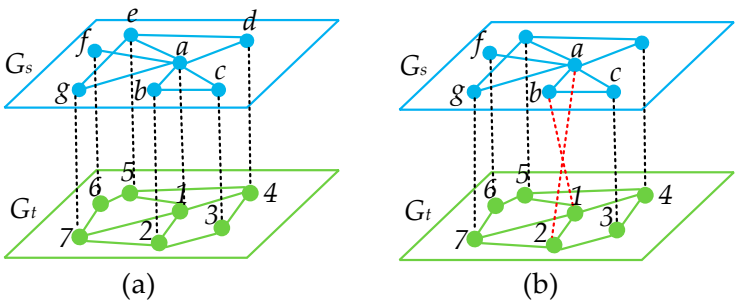


**Figure 2.** An illustration of two different node mappings in two networks. (**a**) The nodes of the different networks are correctly matched. (**b**) Some nodes are incorrectly matched.

We can calculate the MNC score for all pairs of mappings in $M$ using the following formula:

$$MNC(M) = \frac{1}{|M|} \sum_{(u,v \in M)} J(u,v) \tag{3}$$

where, $|M|$ is the number of mapping pairs. The formula calculates the average neighborhood similarity across all mapping pairs, which serves as a metric for the quality of network alignment. Higher MNC values indicate that the node pairs in the two networks are more aligned, suggesting that their neighborhood structures are more similar.

### *3.2. Generation Nodes Embedding Using Graph Augmentation*

A Graph Neural Network (GNN) consists of multiple convolutional layers, with each layer updating the node embeddings based on the previous layer's representations. However, as the number of layers increases, over-smoothing can occur. This phenomenon causes the node representations to lose their diversity and distinctiveness, which reduces their discriminative power and can lead to matching ambiguity. Moreover, noise in the network can be amplified by propagation and aggregation across multiple layers, affecting matching accuracy.

To mitigate the problem of over-smoothing, we obtain multi-order node embeddings by extracting outputs from the different layers of the GNN model. The multi-order embedding captures the node feature at different neighborhood scales. In our study, we use the GNN architecture proposed in [29]. Let $L$ denote the depth of the GNN model. For each node $v$ in a network, its multi-order node embeddings are denoted as $h_*^{(0)}(v)$, $h_*^{(1)}(v)$, ..., $h_*^{(\mathrm{L})}(v)$, where $h_*^{(0)}(v) = X_*(v)$ and $h_*^{(1)}(v)$ is the embedding of node $v$ at layer l. At the $k$-th layer of the GNN, the feature vector $h_*^{(k)}(v)$ for node $v$ is computed as follows:

$$h_*^{(k)}(v) = \sigma(h_*^{(k-1)}(v) \cdot W_1^{(k)} + \sum_{u \in N_*(v)} h_*^{(k-1)}(u) \cdot W_2^{(k)}), k = 1, 2, \ldots, L \tag{4}$$

where $W_1^{(k)}$ and $W_2^{(k)}$ is the weight matrices in the $k$-th layer of the GNN that are shared between the source and target networks. Weight sharing avoids the complex process involved in traditional methods that first learn the node embeddings and then perform spatial alignment. $\sigma(\cdot)$ is the activation function. In our study, we choose theTanh function.

To increase the robustness of the model to noise, we introduce two types of noise, structural and attribute noise, into the model training process so that our model can cope better with noisy or incomplete network data. For structural noise, we simulate structural changes in the network by randomly removing a certain percentage of edges. For attribute noise, we perturb the features of the nodes by adding Gaussian noise to the attribute matrix. Specifically, we generate a Gaussian noise matrix $\mathbf{N}_* \in \mathbb{R}^{|V_*| \times f}$ whose elements $N_*(i,j)$

follow a normal distribution with mean 0 and standard deviation 0.1. We then multiply this noise matrix by the original feature matrix $\mathbf{X}_*$ to obtain the final noisy feature matrix $\mathbf{X}_*^{'}$:

$$\mathbf{X}_*^{'} = \mathbf{X}_* \times \mathbf{N} \tag{5}$$

The GNN model is trained using an unsupervised learning strategy to discover and understand the inherent structure and features of the data without relying on labelled data. The training process includes two loss functions: structural-aware loss (*Ls*) and noise-aware loss (*Ln*). The structural-aware loss ensures that the model accurately captures and preserves the topological properties of the network data. The noise aware loss function aims to enhance the model's robustness to data noise, enabling steady learning even in its presence. These two loss functions work together to guide the model in effectively learning deep representations of network data under unsupervised conditions.

The learning process of node embedding is guided by the loss function *Ls*, which measures the discrepancy between the predicted adjacency matrix, based on the node features output by different network layers, and the given multi-order adjacency matrix. The formula for calculating this function is as follows:

$$Ls_*^{(k)} = \frac{1}{|V_*|}\sum_{i=1}^{|V_*|}\sum_{j=1}^{|V_*|}(P_{ij}^{(k)} - \overline{A}_{ij}^{(k)})^2, k = 1, 2, \ldots, L \tag{6}$$

$$\mathbf{P}_*^{(k)} = \text{normalize}(\mathbf{H}_*^{(k)}) \cdot (\text{normalize}(\mathbf{H}_*^{(k)}))^{\mathrm{T}} \tag{7}$$

$$Ls = \sum_{k=1}^{L}(Ls_s^{(k)} + Ls_t^{(k)}) \tag{8}$$

where $\mathbf{P}_*^{(k)}$ is the predicted adjacency matrix, and $\overset{-(k)}{\mathbf{A}}_*$ is the matrix of order $k$, which captures more distant neighbor information by considering $k$-hop relationships between nodes, constructed from the adjacency matrix. The multi-order matrix is constructed using the following method:

$$\overset{-(k)}{\mathbf{A}}_* = \mathbf{D}_*^{(k)} \cdot (\mathbf{A}_*^{(k-1)}) \cdot \mathbf{D}_*^{(k)}, k = 1, 2, \ldots, L \tag{9}$$

$$\mathrm{A}_*^{(k)} = \mathbf{A}_*^{(k-1)} + \mathbf{A}_*^{(k-1)} \cdot \mathbf{A}_*^{(k-1)} \tag{10}$$

where $\mathrm{A}_*^{(0)} = \mathbf{A}_* + \mathbf{I}$, $\overset{-(1)}{\mathbf{A}}_* = \mathbf{D}_*^{(1)} \cdot (\mathbf{A}_*^{(0)}) \cdot \mathbf{D}_*^{(1)}$, and, as $k$ increases, $\mathbf{A}_*^{(k)}$ encapsulates information over additional hops, thereby facilitating the model's ability to capture long-range dependencies within the network. This approach allows each node to learn not only from its immediate neighbors, but also from more distant neighbors, thereby improving the model's understanding of the structure and features of the network. The diagonal matrix $\mathbf{D}_*^{(k)}$ is computed from $\mathbf{A}_*^{(k-1)}$ as follows:

$$(D_*^{(k)})_{ii} = (\sum_j (A_*^{(k-1)})_{ij})^{-0.5}, k = 1, \ldots, L \tag{11}$$

The noise-aware loss function aims to guide the GNN model in generating representations that are consistent with the original data-embedding vectors, even when faced with augmented data. This loss function targets embedding vectors that deviate from the original embedding vectors within a pre-set threshold and encourages the model to be insensitive to these variations. This increases the model's robustness to noise in the input data. The model can maintain important structural information while allowing for some

level of noise with the use of the noise-aware loss function. The formula for calculating this function is as follows:

$$Ln_*^{(k)} = \frac{1}{|V_*|}\sum_{i=1}^{|V_*|}\left(\left\|h_*^{(k)}[i] - h_*^{\prime(k)}[i]\right\|_2^2 \cdot I\left(\left\|h_*^{(k)}[i] - h_*^{\prime(k)}[i]\right\|_2 > \theta\right)\right), k = 1, 2, \ldots, L \quad (12)$$

$$Ln = \sum_{k=1}^{L}\left(Ln_s^{(k)} + Ln_t^{(k)}\right) \quad (13)$$

where $h_*^{(k)}[i]$ is the $k$-th order embedding of the $i$-th node in either the source or target network; $h_*^{\prime(k)}[i]$ is the $k$-th order embedding of the $i$-th node in the perturbed source or target network; $\|\cdot\|_2$ signifies the Euclidean norm, commonly referred to as the length of a vector; $\theta$ is a predefined threshold; and $\mathrm{I}(\cdot)$ is an indicator function that assumes a value of 1 when its internal condition is true, and 0 otherwise.

### *3.3. Pseudo-Anchor Nodes Collection and Self-Supervised Learning*

Anchor nodes can act as a direct supervisory signal to guide the model towards learning more accurate network representations, thereby enhancing the performance of network alignment tasks. Furthermore, anchor nodes can serve as a regularization mechanism, helping the model avoid over-fitting and improving its generalization capabilities. However, obtaining anchor nodes can be challenging in real-world applications, especially within large-scale networks. In response, researchers have introduced pseudo-anchor nodes to optimize network alignment performance. However, if the selection of pseudo-anchor nodes is biased, it may lead the model to learn incorrect mapping relationships, subsequently compromising the alignment outcome. In this work, we employ the MNC strategy to collect high-quality pseudo-anchor nodes.

Specifically, we calculate the similarity between multi-order embeddings to obtain several similarity matrices. These matrices are then combined to create a composite embedding similarity matrix (alignment matrix), denoted $\mathrm{S}' \in \mathbb{R}^{|V_s|\times|V_t|}$, using the following formula:

$$\mathbf{S}^{'} = \sum_{k=0}^{L}\frac{1}{L+1}\mathbf{S}^{'(k)} \quad (14)$$

$$\mathbf{S}^{'(k)} = \mathrm{normalize}(\mathbf{H}_s^{(k)}) \cdot (\mathrm{normalize}(\mathbf{H}_t^{(k)}))^{\mathrm{T}} \quad (15)$$

After obtaining the alignment matrix $\mathbf{S}^{'}$, we create an initial node mapping set $M$ using a greedy matching algorithm [30]. Following this, the neighborhood similarity matrix $\mathbf{J}$ is calculated for all node pairs using Equation (1). A fixed ratio of node pairs is then selected as pseudo-anchor nodes, based on both the neighborhood similarity matrix and the alignment matrix.

The execution process of the pseudo-anchor nodes generation is summarized in Algorithm 1.

**Algorithm 1** Collect Pseudo-Anchor Nodes

**Input:** source network $G_s$, target network $G_t$, alignment matrix $\mathbf{S}'$, ratio of pseudo-anchor nodes $r$.
**Output:** pseudo-anchor nodes set $T = \{(i,j)|i \in V_s, j \in V_t\}$.

1. Calculate the number of the pseudo-anchor nodes, $TrainNum = |V_s| * r$.
2. Obtain the mapping set $M$ using a greedy matching algorithm and the alignment matrix $\mathbf{S}'$.
3. Compute the neighborhood similarity matrix $\mathbf{J}$ for the nodes of the two graphs.
4. Compute the weighted similarity matrix $\mathbf{M} = \mathbf{S}' * \mathbf{J}$.
5. For each row of matrix $\mathbf{M}$, identify the *index* of the maximum value and the corresponding *value*.
6. *Items* = [(*node1*, *node2*, *similarity*) | $node1 \in V_s$, $node2 \in V_t$, $similarity \in Value$]
7. Sort the *Items* list in descending order based on *similarity* values.
8. $T$ = ItemsSorted[:*TrainNum*]
9. **Return** $T$

During the second phase of model training, we utilize the pseudo-anchor node pairs within set $T$ to provide enhanced supervisory signals. These pseudo-anchor node pairs are instrumental in enabling the model to learn correspondences between nodes in the two networks with greater accuracy. Specifically, for each node pair $(i, j)$ in $T$, we employ the loss function $La$ to measure the discrepancy between their embedding vectors. The design principle of the loss function $La$ is to minimize the disparity between the embedding vectors of matched node pairs, thereby ensuring that similar nodes in the source and target graphs are assigned similar embedding representations. The calculation of the loss function $La$ is as follows:

$$La^{(k)} = \frac{1}{|T|} \sum_{(i,j \in T)} \left\| h_i^{(k)} - h_j^{(k)} \right\|^2, k = 1, 2, \ldots, L \tag{16}$$

$$La = \sum_{k=1}^{L} (La_s^{(k)}) \tag{17}$$

### *3.4. Alignment and Refinement*

After completing the second phase of model training, we obtain an updated alignment matrix $\mathbf{S}''$. Following this, a refinement process is carried out to optimize the alignment matrix and improve alignment accuracy. The process involves iteratively adjusting the similarity scores between matched node pairs, thereby strengthening the consistency of the neighborhood structure among matched nodes. This strategy uses the topological information of the network to refine the matching quality between nodes iteratively.

During each iteration, the model updates the alignment matrix using the adjacency matrices of the source network and the target network. The iterative update rule is as follows:

$$\mathbf{S}''' = \mathbf{S}'' \circ \mathbf{A}_s * (\mathbf{S}'' * \mathbf{A}_t) + \varepsilon \tag{18}$$

where symbol $\circ$ denotes the Hadamard product, and $\varepsilon$ is a small positive scalar. The use of $\varepsilon$ introduces randomness, which can help to reduce the risk of over-fitting in the model.

To prevent a potential decrease in alignment accuracy caused by too many iterations, the module includes a termination mechanism. The iteration is automatically stopped if the consistency score of matched neighbors starts to deteriorate during the iterative process. This ensures that the final alignment score is maximized.

The process of executing MANNA is summarized in Algorithm 2.

**Algorithm 2** MANNA

**Input:** source network $G_s$, target network $G_t$, hyper-parameters *epochs*, $\alpha$, $\beta$, $\varphi$, $T$.
**Output:** final alignment matrix **S**.

Obtain the enhanced graphs $G_s'$ and $G_t'$.
Model←Initialize the GNN model.
**for** some *epochs* **do**
  $H_s^{(0)}, \ldots, H_s^{(L)} = \text{Model}(A_s, X_s)$, $H_t^{(0)}, \ldots, H_t^{(L)} = \text{Model}(A_t, X_t)$.
Obtain structural aware loss *Ls* using (6), (7) and (8).
  $H_s'^{(0)}, \ldots, H_s'^{(L)} = \text{Model}(A_s', X_s')$, $H_t'^{(0)}, \ldots, H_t'^{(L)} = \text{Model}(A_t', X_t')$.
Obtain structural aware loss $Ls'$ using (6), (7) and (8).
Obtain noise aware loss *Ln* using (12) and (13).
  $L1 = \alpha \times Ls + (1 - \alpha) \times Ls'$
  $L2 = \beta \times Ls + (1 - \beta) \times Ln$
Back-propagation and optimization of the model parameters.
Calculate the similarity matrix $\mathbf{S}^{'}$ using (14) and (15).
$T$←Obtain pseudo-anchor nodes use Algorithm 1.
**for** some *epochs* **do**
  $H_s^{(0)}, \ldots, H_s^{(L)} = \text{Model}(A_s, X_s)$, $H_t^{(0)}, \ldots, H_t^{(L)} = \text{Model}(A_t, X_t)$.
Obtain structural aware loss *Ls* using (6), (7) and (8).
Obtain anchors aware loss *La* using (16) and (17).
  $H_s'^{(0)}, \ldots, H_s'^{(L)} = \text{Model}(A_s', X_s')$, $H_t'^{(0)}, \ldots, H_t'^{(L)} = \text{Model}(A_t', X_t')$.
Obtain structural aware loss $Ls'$ using (6), (7) and (8).
Obtain noise aware loss *Ln* using (12) and (13).
  $L1 = \alpha \times Ls + (1 - \alpha) \times Ls'$
  $L2 = \beta \times Ls + (1 - \beta) \times Ln$
  $L = \varphi \times L2 + (1 - \varphi) \times La$
Back-propagation and optimization of the model parameters.
Calculate the similarity matrix $\mathbf{S}^{''}$ using (14) and (15).
**for** *iteration* in 1, 2, ..., $T$ **do**
Calculate the update similarity matrix $\mathbf{S}^{'''}$ using (18).
**if** *termination_condition*($\mathbf{S}^{'''}$, $\mathbf{A}_s$, $\mathbf{A}_t$) **then**
    **break**
$\mathbf{S} \leftarrow \mathbf{S}^{'''}$.
**Return S**

The time complexity of the MANNA algorithm is influenced by several components that work in tandem to provide an efficient and accurate network alignment. These components include the Network Enhancement Module, the GNN Module, the Self-Supervised Learning Module, and the Alignment Refinement Module. Below is a more detailed analysis of how each of these components contributes to the overall time complexity.

Network Enhancement Module: This component introduces controlled structural and attribute perturbations to the network to enhance the model's robustness to noise. The complexity of generating structural noise by randomly removing edges is linear with respect to the number of edges, $O(m)$, where $m$ is the number of edges in the network. For attribute noise, if each node has $f$ features, the complexity of adding Gaussian noise to the feature matrix is $O(n \cdot f)$, where $n$ is the number of nodes.

GNN Module: The GNN module is at the core of the MANNA algorithm, responsible for learning the structural and attribute features of the networks. The complexity of this module is primarily determined by the depth of the GNN ($L$ layers) and the size of the networks. Each layer processes the nodes and their connections, leading to a time complexity of $O(L \cdot (n + m) \cdot d)$, where $d$ is the dimensionality of the node features.

Self-Supervised Learning Module: This module identifies high-quality pseudo-anchor nodes based on the MNC principle. The process involves calculating the similarity ma-

trix and selecting pseudo-anchor nodes, which can be computationally intensive. The complexity of this module can be approximated as $O(n^2 \cdot k)$, where $k$ is the number of pseudo-anchor nodes selected.

Alignment Refinement Module: The alignment refinement module iteratively updates the alignment matrix to strengthen the consistency of neighborhood structures between matched nodes. Each iteration involves recalculating the alignment matrix based on the current node embeddings, with a complexity of $O(n^2)$. Over $T$ iterations, the total complexity for this module is $O(T \cdot n^2)$.

## 4. Evaluation

### *4.1. Datasets*

We use three openly available datasets: douban online/offline [12,22,23], allmovie/imdb [12,22,23], and flickr/myspace [12,27]. The douban dataset comprises two subsets, douban online and douban offline, obtained from Douban, a Chinese social application. These subsets were created by analyzing users' social interactions. The application douban online reflects users' online social networks, while douban offline tracks their real-world social activities, such as attending events in the physical world. The allmovie/imdb dataset consists of two subnetworks that are based on the relationships between actors in films. Each node in this dataset represents a movie, and each edge indicates a connection between two movies that feature at least one common actor. The flickr/myspace dataset combines networks from the separate social media platforms Flickr and Myspace. Nodes represent individual users, while edges represent the friendships between them. Table 1 provides a detailed description of these datasets.

**Table 1.** Details of the datasets.

| Dataset | Nodes | Edges | Attributes | Anchor Nodes | Average Degree | Max Degree | Clustering Coefficient |
|---|---|---|---|---|---|---|---|
| douban online/offline | 3906/1118 | 8164/1511 | 538 | 1118 | 4.18/2.70 | 97/26 | 0.0404/0.0866 |
| allmovie/imdb | 6011/5713 | 124,709/119,073 | 14 | 5175 | 41.49/41.68 | 159/157 | 0.3813/0.3833 |
| flickr/myspace | 6714/10,733 | 7333/11,081 | 3 | 267 | 2.18/2.06 | 639/164 | 0.0014/0.0 |

### *4.2. Baselines*

In order to evaluate the performance of the proposed model, the following six models are compared:

REGAL [9]: REGAL combines the topological structure and node attributes to generate a matrix that represents the similarity of different network nodes. The Nyström [31] method is then used to decompose the similarity matrix and obtain node embeddings, which are later used for node matching.

GAlign [12]: GAlign is a network alignment method that is based on a graph convolutional network. The model achieves node matching between different networks by learning the embedding of nodes and optimizing using the consistency loss function and adaptive loss function.

GATAL [22]: GATAL is an unsupervised network alignment model that utilizes graph attention networks. It is comprised of three parts: an enhancement learning process, multi-level GAT embedding, and alignment computation. The model enhances adaptability by simulating noise in the real world and employs multi-level GAT to learn the latent features of the nodes.

WAlign [13]: WAlign is a network alignment method that uses node embedding learning and a Wasserstein distance discriminator to achieve efficient and accurate network alignment. The method minimizes the distance between the embedding distributions of the source and target networks.

Grad-Align [23]: Grad-Align is a method that uses a stepwise matching strategy to achieve network alignment. This is done through multi-layer embedding similarity calculation and Tversky similarity coefficient, resulting in improved accuracy and efficiency of alignment.

Grad-Align+ [24]: Grad-Align+ is an enhanced version of Grad-Align that uses centrality-based node feature augmentation (CNFA) to improve the performance of GNN in network alignment tasks. Additionally, it employs a gradual alignment strategy to accurately identify correspondences between nodes across networks, even without additional information.

### *4.3. Parameter Setting and Evaluation Metrics*

The experimental procedures were conducted on a personal computer running Windows 10, equipped with an Intel® i7-4790 processor clocked at 3.60 GHz and 16 GB of RAM. During the MANNA training phase, we employed the Adam optimizer with an initial learning rate of 0.005. The graph neural network architecture consisted of 2 convolutional layers, and we iterated the training for 30 epochs. We set the sampling rate for pseudo-anchor nodes to 0.1, and the embedding dimension to 100. The benchmark algorithm parameters were adjusted according to the recommendations of the original papers, with the exception of the embedding dimensionality, number of convolutional layers, and training epochs, which remained consistent with MANNA. The optimal performance under these parameter configurations was recorded.

Building on previous research, we used two evaluation metrics—*MAP* (Mean Average Precision) [12,22] and *Precision@k* [12,22–24]—to compare the alignment effectiveness of different methods.

The *MAP* metric was used to evaluate the performance of algorithms in identifying corresponding nodes (anchor nodes) in the target network for each node in the source network. This metric considers alignment results for all nodes in the source network and calculates the average of the inverse ranks. For each node $i$ in the source network, all nodes in the target network are ranked based on the alignment scores (similarity) in the alignment matrix. The position of the true anchor node for node $i$ in the ranking is determined, and the inverse of its rank is computed. Subsequently, the inverses of the ranks of the true anchor nodes for all nodes in the source network are averaged to obtain the *MAP* value. The formula is as follows:

$$MAP = \frac{1}{|V_s|} \sum_{i \in V_s} \frac{1}{r_i} \tag{19}$$

where $r_i$ denotes the rank of the true anchor node for the node $i$ in the source network within the alignment results.

*Precision@k* measures the proportion of correctly identified true anchor nodes among the top $k$ candidate anchor nodes for each node. To achieve this, we first identify the top $k$ target network nodes with the highest alignment scores to node $i$ in the alignment matrix and place these target nodes in the set $Q_k(i)$ for each node $i$ in the source network. We then check whether the node pair $(i, j)$ appears in the set of true anchor node pairs $T$ for each node $j$ in the set $Q_k(i)$. If the node pair $(i, j)$ is present in the set $T$, it indicates a correct match and is counted as a successful match. The *Precision@k* value is calculated by dividing the number of successful matches by the total number of nodes in the source network as follows:

$$Precision@k = \frac{\sum_{i \in V_s} \sum_{j \in V_t, j \in Q_k(i)} \mathrm{I}((i, j) \in T)}{|V_s|} \tag{20}$$

where the indicator function $\mathrm{I}(\cdot)$ is a function that takes the value of 1 when the condition is true and 0 otherwise.

### *4.4. Model Performance Comparison*

To evaluate the performance of the method proposed in this paper, we compared it with six baselines. Table 2 presents the experimental results of the proposed method and the other six baselines on three datasets. The optimal results of each column are in bold, and the best results of the baseline are underlined.

**Table 2.** Network alignment comparison of three datasets.

| Dataset | Metrics | REGAL | GAlign | GATAL | WAlign | Grad-Align | Grad-Align+ | MANNA |
|---|---|---|---|---|---|---|---|---|
| douban online/offline | *MAP* | 0.1005 | 0.5215 | 0.5926 | 0.3207 | 0.5920 | 0.6807 | **0.7223** |
| | *Precision@1* | 0.0456 | 0.4034 | 0.4839 | 0.2120 | 0.4928 | 0.5903 | **0.6273** |
| | *Precision@5* | 0.1360 | 0.6601 | 0.7272 | 0.4186 | 0.6914 | 0.7907 | **0.8371** |
| | *Precision@10* | 0.2030 | 0.7639 | 0.8122 | 0.5340 | 0.7862 | 0.8515 | **0.8980** |
| allmovie/imdb | *MAP* | 0.1888 | 0.7133 | 0.7331 | 0.6501 | 0.7023 | 0.8915 | **0.9375** |
| | *Precision@1* | 0.0953 | 0.6521 | 0.6637 | 0.5715 | 0.6303 | 0.8460 | **0.9160** |
| | *Precision@5* | 0.2687 | 0.7810 | 0.8158 | 0.7370 | 0.7831 | 0.9416 | **0.9648** |
| | *Precision@10* | 0.3869 | 0.8257 | 0.8635 | 0.7907 | 0.8375 | 0.9528 | **0.9700** |
| flickr/myspace | *MAP* | 0.0227 | 0.0200 | 0.0212 | 0.0190 | 0.0310 | 0.0173 | **0.0350** |
| | *Precision@1* | 0.0075 | 0.0075 | 0.0075 | 0.0037 | 0.0187 | 0.0000 | **0.0262** |
| | *Precision@5* | 0.0262 | 0.0262 | 0.0225 | 0.0262 | 0.0412 | 0.0112 | **0.0449** |
| | *Precision@10* | 0.0412 | 0.0449 | 0.0375 | 0.0375 | 0.0599 | 0.0337 | **0.0712** |

The experimental results show that there are significant performance differences among various baselines on different datasets. REGAL uses xNetMF, a low-rank approximation and implicit matrix decomposition technique, to learn node representations in different networks. In contrast, other baselines use Graph Neural Networks (GNNs) to learn node representations from graph data. GNNs can effectively capture both the local and global neighborhood information of nodes by transferring and aggregating information on graphs. This makes them more efficient in expressing complex dependencies between nodes. However, xNetMF mainly relies on node degree distribution and attribute information, which may limit its flexibility in capturing complex interactions and dependencies between nodes. Among all baseline algorithms, REGAL has the worst performance. Of the GNN-based baselines, WAlign performs the worst. This could be attributed to the weight-sharing mechanism used by other algorithms. The weight-sharing mechanism allows the model to learn common, meaningful representations from both networks, which is crucial for the network alignment task. Additionally, both Grad-Align and Grad-Align+ employ a stepwise network alignment strategy to gradually discover node pairs through an iterative process. The importance of the structural positions of the nodes in their respective networks is emphasized by measuring the similarity of the aligned cross-network node pairs, thus optimizing the alignment results. Grad-Align+ outperformed other baselines on the douban online/offline and allmovie/imdb datasets, while Grad-Align outperformed other baselines on the flickr/myspace dataset.

MANNA consistently outperformed the baselines in all three datasets, demonstrating its superior performance. Compared to the highest-performing baseline, Grad-Align+, on the douban dataset, MANNA showed a significant improvement in performance metrics: approximately 15.3% in *MAP*, 27.3% in *Precision@1*, 20.6% in *Precision@5*, and 14.2% in *Precision@10*. MANNA outperformed Grad-Align+ on the almovie/imdb dataset with a remarkable enhancement of around 33.5% in *MAP*, 45.9% in *Precision@1*, 24.0% in *Precision@5*, and 15.5% in *Precision@10*. Similarly, on the flickr/myspace dataset, MANNA

surpassed Grad-Align with significant gains of approximately 12.9% in *MAP*, 39.6% in *Precision@1*, 11.4% in *Precision@5*, and 20.3% in *Precision@10*.

### *4.5. Ablation Experiment*

In this section, we conducted experiments on three datasets to investigate the contributions of various key modules within our model to the overall performance. To evaluate the efficacy of the different components of the model more clearly, we removed some key modules from the original model and monitored the resulting performance changes.

Base Model (BM): This version uses only unsupervised learning techniques to train the graph neural network for node representation. It removed all the modules designed to improve matching performance.

To evaluate the impact of individual components on model performance, we incorporated each of the following modules into the base model separately: the Network Enhancement (NE) module, the Self-Supervised Learning (SL) module, and the Alignment Refinement (AR) module. This enabled an evaluation of their respective contributions to the model's overall performance. The study analyzed the performance of three configurations: the model with only the NE module (only NE), the model with only the SL module (only SL), and the model with only the AR component (only AR).

Experiments were conducted to investigate the impact of the absence of each component on the model's performance by removing them individually. The configurations tested included the model without the NE component (w/o-NE), the model without the SL component (w/o-SL), and the model without the AR component (w/o-AR).

The experimental results are detailed in Table 3, where the optimal results in each column are highlighted in bold, and the suboptimal results are underlined.

**Table 3.** Network alignment comparison of four real-world datasets.

| Dataset | Metrics | BM | Only NE | Only SL | Only AR | w/o-NE | w/o-SL | w/o-AR | MANNA |
|---|---|---|---|---|---|---|---|---|---|
| Douban Online/Offline | *MAP* | 0.6288 | 0.6389 | 0.6382 | 0.7133 | 0.7217 | 0.7161 | 0.6606 | **0.7223** |
| | *Precision@1* | 0.5250 | 0.5313 | 0.5331 | 0.6190 | 0.6270 | 0.6234 | 0.5501 | **0.6273** |
| | *Precision@5* | 0.7496 | 0.7558 | 0.7603 | 0.8229 | 0.8363 | 0.8256 | 0.7800 | **0.8371** |
| | *Precision@10* | 0.8283 | 0.8354 | 0.8435 | 0.8792 | 0.8882 | 0.8882 | 0.8623 | **0.8980** |
| Allmovie/Imdb | *MAP* | 0.7772 | 0.7796 | 0.7846 | 0.9349 | 0.9374 | 0.9345 | 0.7899 | **0.9375** |
| | *Precision@1* | 0.7112 | 0.7149 | 0.7363 | 0.9117 | 0.9152 | 0.9124 | 0.7819 | **0.9160** |
| | *Precision@5* | 0.8545 | 0.8576 | 0.8646 | 0.9629 | 0.9635 | 0.9596 | 0.8815 | **0.9648** |
| | *Precision@10* | 0.8918 | 0.8933 | 0.8998 | 0.9681 | 0.9689 | 0.9650 | 0.9041 | **0.9700** |
| Flickr/Myspace | *MAP* | 0.0218 | 0.0222 | 0.0274 | 0.0283 | 0.0332 | 0.0249 | 0.0232 | **0.0350** |
| | *Precision@1* | 0.0075 | 0.0075 | 0.0112 | 0.0150 | 0.0150 | 0.0075 | 0.0075 | **0.0262** |
| | *Precision@5* | 0.0187 | 0.0187 | 0.0225 | 0.0225 | 0.0375 | 0.0262 | 0.0300 | **0.0449** |
| | *Precision@10* | 0.0262 | 0.0262 | 0.0412 | 0.0524 | 0.0562 | 0.0524 | 0.0337 | **0.0712** |

The experimental findings suggest that across the three datasets, the comprehensive model achieves the highest performance. Among all the variant models, the one lacking the Network Enhancement (NE) component (w/o-NE) demonstrates the most superior performance. Specifically, the integration of the NE component singularly enhances the model's efficacy on the douban online/offline and allmovie/imdb datasets. Conversely, on the flickr/myspace dataset, the inclusion of the NE component does not enhance accuracy, potentially due to the scarcity of attribute features in this dataset. The incorporation of the Self-Supervised Learning (SL) component universally improves the model's performance across all three datasets, indicating that modifying the distance between pseudo-anchor nodes aids in the extraction of node features that are more aligned with the network align-

ment task. The Alignment Refinement (AR) component contributes most significantly to the enhancement of alignment performance. For the douban dataset, the model comprising solely the AR component outperforms the Base Model (BM) with improvements of approximately 13.4% in *MAP*, 17.1% in *Precision@1*, 10.1% in *Precision@5*, and 6.0% in *Precision@10*. On the allmovie/imdb dataset, the only AR model surpasses the BM with enhancements of around 20.3% in *MAP*, 28.2% in *Precision@1*, 12.7% in *Precision@5*, and 8.6% in *Precision@10*. On the flickr/myspace dataset, the only AR model exceeds the BM with significant gains of approximately 29.8% in *MAP*, 46.7% in *Precision@1*, 19.8% in *Precision@5*, and 100% in *Precision@10*. The removal of the AR component in isolation results in a notable degradation of the model's performance. Overall, the AR component's iterative refinement of the similarity matrix, which reinforces the neighborhood consistency of matched node pairs, emerges as the most effective strategy for enhancing alignment accuracy. While the NE module and the SL module also contribute to performance improvement, their impact is relatively less pronounced compared to the AR component.

### *4.6. Hyper-Parameter Sensitivity Analysis*

This section analyzes the impact of three parameters on the experimental results: embedding dimensionality, number of epochs, and the ratio of pseudo-anchor nodes. The target parameter is changed while keeping all other parameters constant.

First, we analyzed the influence of different embedding dimensions (5, 10, 25, 50, 100, 150, 200) on model performance. The experimental results are presented in Figures 3 and 4. Figure 3 shows the trends in the *MAP* value as the embedding dimensionality increases. Similarly, Figure 4 displays the performance of the *Precision@10* value under the same conditions. According to the analysis, the model performs optimally with an embedding dimensionality of 150 for the douban online/offline and allmovie/imdb datasets. However, for the flickr/myspace dataset, the model's performance becomes unstable as the embedding dimensionality increases. The *MAP* value reaches its peak at an embedding dimensionality of 100, while the *Precision@10* value is best at 200. The fluctuation in performance can be attributed to the sparse network structure and insufficient node attribute information.

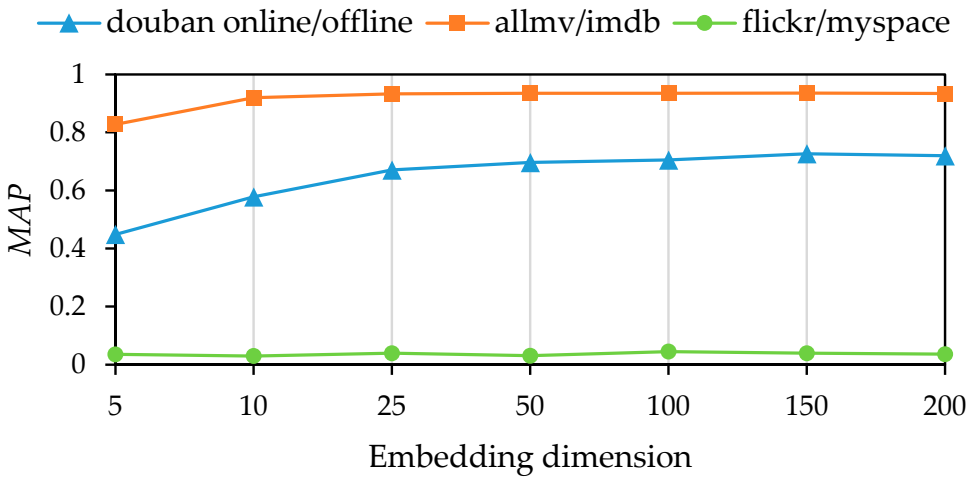


**Figure 3.** Effect of increasing embedding dimensions on *MAP* value.

Next, we analyzed the effects of varying the number of epochs for the graph neural network. The experimental setup involved setting the number of epochs to 10, 20, 30, 40, 50, 100, and 150. The results of this experiment are depicted in Figures 5 and 6. Figure 5 shows the *MAP* metric, while Figure 6 shows the *Precision@10* metric, as in the previous experiment. The results indicate that the model achieved satisfactory performance with a relatively low number of epochs across all three datasets. However, the optimal point varies for each dataset. The douban online/offline and allmovie/imdb datasets achieve their optimal performance at 40 epochs. After this point, performance slightly declines

with additional epochs. On the flickr/myspace dataset, the model's performance plateaus at 30 epochs, with only slight improvements observed thereafter.

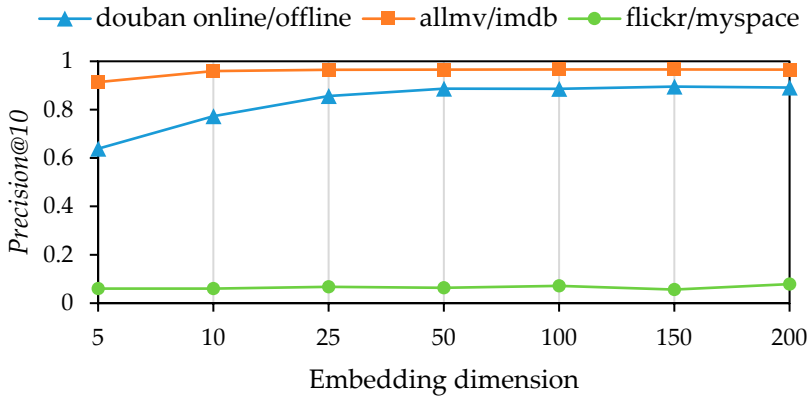


**Figure 4.** Effect of increasing embedding dimensions on *Precision@10* value.

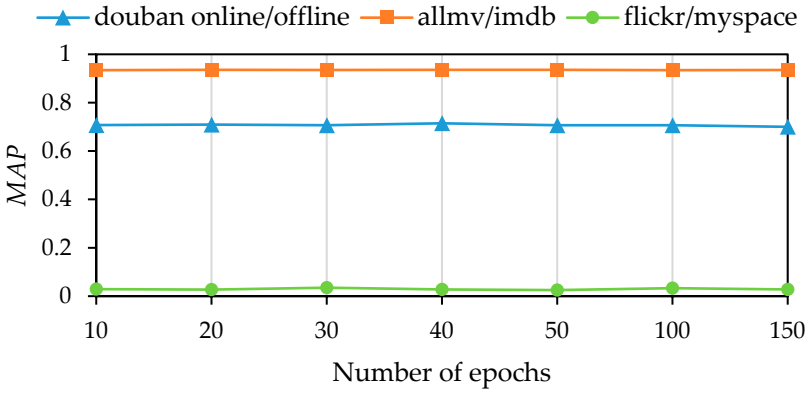


**Figure 5.** Effect of increasing the number of epochs on *MAP* value.

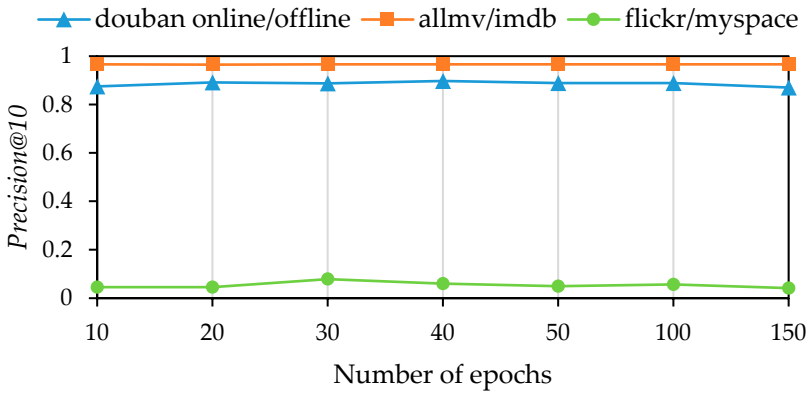


**Figure 6.** Effect of increasing the number of epochs on *Precision@10* value.

Finally, we analyzed the impact of the varying ratios of pseudo-anchor nodes on experimental outcomes. The experiment involved setting the ratio of pseudo-anchor nodes to 0.01, 0.05, 0.1, 0.2, 0.3, 0.4 and 0.6. The results of this investigation are visualized in Figures 7 and 8. As with the previous experiments, Figure 7 represents the *MAP*, while Figure 8 represents the *Precision@10*. The visualizations show that the model's performance improves with an increase in the proportion of pseudo-anchor nodes across all three datasets. The model's alignment performance, as measured by the *MAP*, is marginally enhanced with the increase in the proportion of pseudo-anchor nodes in the douban

online/offline and allmovie/imdb datasets, although the improvement is not significant. In the flickr/myspace dataset, the model's performance plateaus at a pseudo-anchor node proportion of 0.1. Further increases in this proportion lead to a decline in performance, as shown by *Precision@10*. This phenomenon may be attributed to the sparse and locally clustered network structures in the flickr/myspace datasets. Achieving matching neighbor consistency in network structures is challenging. An increase in the proportion of pseudo-anchor nodes may introduce excessive noise, potentially interfering with the alignment algorithm's performance.

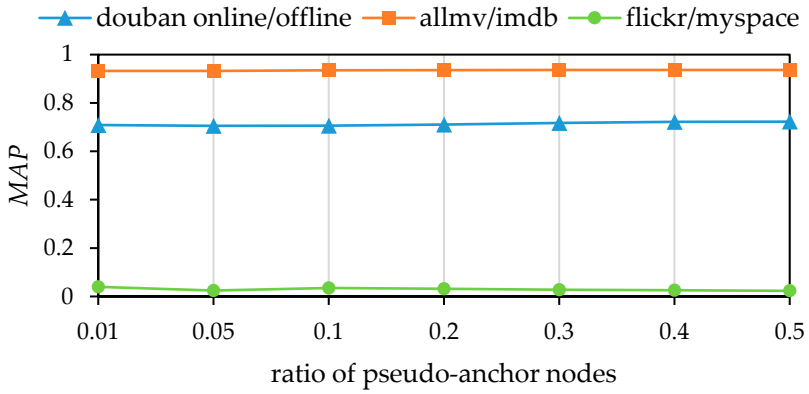


**Figure 7.** Effect of increasing the ratio of pseudo-anchor nodes on *MAP* value.

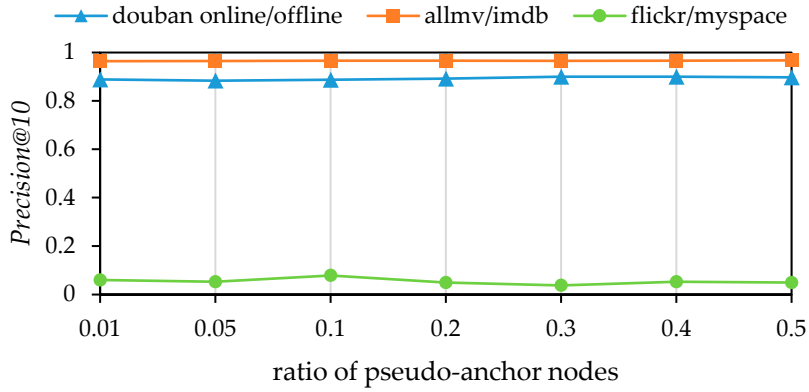


**Figure 8.** Effect of increasing the ratio of pseudo-anchor nodes on *Precision@10* value.

## 5. Conclusions

This paper proposes MANNA, an unsupervised network alignment framework that overcomes the limitations of anchor node unavailability and network noise sensitivity. MANNA aligns nodes across different networks without the need for predefined anchor nodes by leveraging the capabilities of the Graph Neural Network (GNN) and the Matched Neighborhood Consistency (MNC) principle. Our method has shown significant improvements in both alignment accuracy and robustness compared to state-of-the-art methods. This was validated through a comprehensive suite of experiments on various datasets.

The key innovation of MANNA is its ability to derive rich node representations through multi-order embeddings and iteratively refine alignment outcomes by consistently reinforcing neighborhood structural consistency. The use of pseudo-anchor nodes in self-supervised learning reduces reliance on anchor nodes and establishes a robust framework for model training under noisy conditions.

Although our approach is effective, there is still potential for further research. For instance, progress could be made by developing adaptive strategies to determine the optimal number of pseudo-anchor nodes across different network architectures. Integrating

attention mechanisms with node-embedding techniques could improve their descriptive abilities.

**Author Contributions:** Methodology, Y.L. and F.Q.; software, validation, Y.L. and F.Q.; data curation, F.Q.; writing—original draft preparation, Y.L. and L.Z.; writing—review and editing, Y.L. and F.Q. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the Key Research Projects of Natural Science in Colleges and Universities of Anhui Province (Grant No. 2022AH051753 and 2022AH051749), the Excellent Innovative Research Team of universities in Anhui Province (Grant No. 2023AH010056), the Collaborative Innovation Project for Universities in Anhui Province (Grant No. GXXT-2023-050); the Anhui Provincial Quality Engineering Demonstration Experimental Training Center Project for Higher Education Institutions (Grant No. 2022 sysx031); the Outstanding Young Backbone Talent Home and Abroad Visiting Study Program at University of Anhui Province (Grant No. gxgnfx2021148).

**Data Availability Statement:** The data presented in this study are available in https://github.com/thanhtrunghuynh93/networkAlignment (accessed on 1 October 2022).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Huynh, T.T.; Toan, N.T.; Tong, V.V.; Hoang, T.D.; Thang, D.C.; Hung, N.Q.V.; Sattar, A. A comparative study on network alignment techniques. *Expert Syst. Appl.* **2020**, *140*, 112883.
2. Huang, S.R.; Zhang, J.; Schonfeld, D.; Wang, L.; Hua, X.S. Two-stage friend recommendation based on network alignment and series expansion of probabilistic topic model. *IEEE Trans. Multim.* **2017**, *19*, 1314–1326. [CrossRef]
3. Li, L.C.; Dannenfelser, R.; Zhu, Y.; Hejduk, N.; Segarra, S.; Yao, V. Joint embedding of biological networks for cross-species functional alignment. *Bioinformatics* **2023**, *39*, btad529. [CrossRef] [PubMed]
4. Yang, J.; Lu, Z.S.; Chen, X.; Xu, D.L.; Ding, D.W.; Ding, Y.R. GCNA-Cluster: A gene co-expression network alignment to cluster cancer patients algorithm for identifying subtypes of pancreatic ductal adenocarcinoma. *IEEE ACM Trans. Comput. Biol. Bioinform.* **2023**, *20*, 3556–3566. [CrossRef] [PubMed]
5. Zhang, Y.H.; Lu, D.; Bu, C.Y.; Yu, K.; Wu, X.D. Cross-knowledge graph relation completion for non-isomorphic cross-lingual entity alignment. In Proceedings of the 29th International DMS Conference on Visualization and Visual Languages, KSIR Virtual Conference Center, San Francisco Bay, CA, USA, 29 June–3 July 2023; pp. 24–31.
6. Huang, Z.J.; Li, Z.; Jiang, H.M.; Cao, T.Y.; Lu, H.Q.; Yin, B.; Subbian, K.; Sun, Y.Z.; Wang, W. Multilingual knowledge graph completion with self-supervised adaptive graph alignment. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; pp. 474–485.
7. Skitsas, K.; Orlowski, K.; Hermanns, J.; Mottin, D.; Karras, P. Comprehensive evaluation of algorithms for unrestricted graph alignment. In Proceedings of the 26th International Conference on Extending Database Technology, Ioannina, Greece, 28–31 March 2023; pp. 260–272.
8. Liu, X.Y.; Tang, J. Network representation learning: A macro and micro view. *AI Open* **2021**, *2*, 43–64. [CrossRef]
9. Heimann, M.; Shen, H.M.; Safavi, T.; Koutra, D. REGAL: Representation learning-based graph alignment. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 117–126.
10. Nguyen, T.T.; Pham, M.T.; Nguyen, T.T.; Huynh, T.H.; Tong, V.V.; Nguyen, Q.V.H.; Quan, T.T. Structural representation learning for network alignment with self-supervised anchor links. *Expert Syst. Appl.* **2021**, *165*, 113857. [CrossRef]
11. Man, T.; Shen, H.W.; Liu, S.H.; Jin, X.L.; Cheng, X.Q. Predict anchor links across social networks via an embedding approach. In Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; pp. 1823–1829.
12. Huynh, T.T.; Tong, V.V.; Nguyen, T.T.; Yin, H.Z.; Weidlich, M.; Hung, N.Q.V. Adaptive network alignment with unsupervised and multi-order convolutional networks. In Proceedings of the 36th IEEE International Conference on Data Engineering, Dallas, TX, USA, 20–24 April 2020; pp. 85–96.
13. Gao, J.; Huang, X.; Li, J.D. Unsupervised graph alignment with Wasserstein distance discriminator. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, 14–18 August 2021; pp. 426–435.
14. Zhang, H.F.; Ren, G.; Ding, X.; Zhou, L.; Zhang, X. Collaborative cross-network embedding framework for network alignment. *IEEE Trans. Netw. Sci. Eng.* **2024**, 1–13. [CrossRef]
15. Heimann, M.; Chen, X.Y.; Vahedian, F.; Koutra, D. Refining network alignment to improve matched neighborhood consistency. In Proceedings of the 2021 SIAM International Conference on Data Mining, Virtual Event, 29 April–1 May 2021; pp. 172–180.
16. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
17. Tang, J.; Qu, M.; Wang, M.Z.; Zhang, M.; Yan, J.; Mei, Q.Z. LINE: Large-scale Information Network Embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077.

18. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
19. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
20. Huynh, T.T.; Tong, V.V.; Duong, C.T.; Thang, H.Q.; Nguyen, Q.V.H.; Sattar, A. Network alignment by representation learning on structure and attribute. In Proceedings of the 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, 26–30 August 2019; pp. 698–711.
21. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.C.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
22. Xiao, Y.L.; Hu, R.M.; Li, D.S.; Wu, J.H.; Zhen, Y.; Ren, L.F. Multi-level graph attention network based unsupervised network alignment. In Proceedings of the 46th IEEE Conference on Local Computer Networks, Edmonton, AB, Canada, 4–7 October 2021; pp. 217–224.
23. Park, J.D.; Tran, C.; Shin, W.Y.; Cao, X. On the power of gradual network alignment using dual-perception similarities. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 15292–15307. [CrossRef] [PubMed]
24. Park, J.D.; Tran, C.; Shin, W.Y.; Cao, X. Node feature augmentation vitaminizes network alignment. *arXiv* **2023**, arXiv:2304.12751.
25. Peng, J.K.; Xiong, F.; Pan, S.R.; Wang, L.; Xiong, X. Robust network alignment with the combination of structure and attribute embeddings. In Proceedings of the 2023 IEEE International Conference on Data Mining, Shanghai, China, 1–4 December 2023; pp. 498–507.
26. Huynh, T.T.; Duong, C.T.; Nguyen, T.T.; Vinh, T.V.; Sattar, A.; Yin, H.Z.; Nguyen, Q.V.H. Network alignment with holistic embeddings. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 1881–1894. [CrossRef]
27. Sun, Q.Q.; Lin, X.M.; Zhang, Y.; Zhang, W.J.; Chen, C.Q. Towards higher-order topological consistency for unsupervised network alignment. In Proceedings of the 39th IEEE International Conference on Data Engineering, Anaheim, CA, USA, 3–7 April 2023; pp. 177–190.
28. Zhang, Z.B.; Gao, S.; Su, S.; Sun, L.; Chen, R.Y. MINING: Multi-granularity network alignment based on contrastive learning. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12785–12798. [CrossRef]
29. Xu, K.; Hu, W.H.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
30. Kollias, G.; Mohammadi, S.; Grama, A. Network Similarity Decomposition (NSD): A fast and scalable approach to network alignment. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 2232–2243. [CrossRef]
31. Drineas, P.; Mahoney, M.W. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.* **2005**, *6*, 2153–2175.