Graph Harmony: Denoising and Nuclear-Norm Wasserstein Adaptation for Enhanced Domain Transfer in Graph-Structured Data

Anonymous authors
Paper under double-blind review

Abstract

Graph-structured data can be found in numerous domains, yet the scarcity of labeled instances hinders its effective utilization of deep learning in many scenarios. Traditional unsupervised domain adaptation (UDA) strategies for graphs primarily hinge on adversarial learning and pseudo-labeling. These approaches fail to effectively leverage graph discriminative features, leading to class mismatching and unreliable label quality. To address these obstacles, we develop the Denoising and Nuclear-Norm Wasserstein Adaptation Network (DNAN). DNAN employs the Nuclear-norm Wasserstein discrepancy (NWD), which can simultaneously achieve domain alignment and class distinguishment. It also integrates a denoising mechanism via a Variational Graph Autoencoder. This denoising mechanism helps capture essential features of both source and target domains, improving the robustness of the domain adaptation process. Our comprehensive experiments demonstrate that DNAN outperforms state-of-the-art methods on standard UDA benchmarks for graph classification.

1 Introduction

While deep learning has made substantial progress in handling graph-structured data, it shares a drawback with other methods in the same category — a heavy reliance on labeled data. This requirement presents a significant obstacle in real-world applications, where the gathering and annotating of graph-structured data come with a steep price tag, both in terms of time and resources. Obtaining detailed labels for graph-structured data, e.g., chemical molecules, is a considerable challenge because chemical molecules are incredibly complex, comprising a large number of atoms connected in various ways through different kinds of bonds. Collecting annotated graph-structured data like social networks is also challenging due to the need to protect personal and sensitive information and the continual changes in network relationships. The label scarcity makes it difficult to derive meaningful insights and hinders the development of strategies and solutions based on deep learning. Therefore, it is highly desirable to relax the need for extensive graph-structured data annotation to replicate the success of deep learning in applications.

To navigate the challenge of label scarcity, Unsupervised Domain Adaptation (UDA) (Ganin & Lempitsky, 2015) has emerged as a promising frontier, with the aim of leveraging labeled data from a related source domain to inform an unlabeled target domain. The principle of UDA is to align the data distributions between the two domains within a common embedding space, enabling a classifier trained on the source domain to perform competently on the target domain. While UDA has been extensively applied to array-structured data (Long et al., 2016; Kang et al., 2019), its translation to graph-structured data remains under-explored. Pioneering methods, such as DANE (Zhang et al., 2019a), integrate generative adversarial networks (GANs) with graph convolutional networks (GCNs) to align the domains. Others, like the approach by Wu et al. (2020), introduce attention mechanisms to reconcile global and local consistencies, again employing GANs for cross-domain node embedding extraction. However, these GAN-based methods have the drawback of class mismatching, lacking clear separability between features from different classes, as they align target and source domain features irrespective of their classes. In addition, these methods are designed for node classification. The UDA strategy for graph classification has not been well-explored.

This paper focuses on the UDA setting for graph classification. We propose the Denoising and Nuclear-Norm Wasserstein Adaptation Network (DNAN) and address the problems in the previous GAN-based methods. Our DNAN benefits from the denoising mechanism with a Variational Graph Autoencoder (VGAE) and the Nuclear-Norm Wasserstein Discrepancy. By leveraging the Nuclear-Norm Wasserstein Discrepancy, it tackles the class mismatch issue in existing graph-based UDA methods. Unlike the previous GAN-based methods, DNAN performs a refined, class-specific alignment of source and target domain distributions within a shared embedding space, preserving the distinct separability of features across classes. The inclusion of the denoising mechanism is also crucial as it enhances feature representation for transferability. By using these two components, DNAN performs competitively and achieves state-of-the-art performance on major UDA benchmarks for graph classification.

2 Related Work

Unsupervised Domain Adaptation A foundational approach within UDA is to reduce the discrepancy between the source and target domain distributions using adversarial learning. A representative method in this space, the Domain Adversarial Neural Network (DANN) (Ganin & Lempitsky, 2015), employs an adversarial training framework to align domain representations by confusing a domain classifier in a shared embedding space. This strategy is adapted from generative adversarial networks (GANs) (Goodfellow et al., 2020), tailored for domain adaptation purposes. Expanding on this adversarial methodology, the FGDA technique (Gao et al., 2021) used a discriminator to discern the gradient distribution of features, thereby achieving better performance in mitigating domain discrepancy. Furthermore, DADA (Tang & Jia, 2020) proposed an innovative strategy by integrating the domain-specific classifier with the domain discriminator to align the joint distributions of two domains more effectively.

Adversarial approaches are complemented by statistical discrepancy measures like Maximum Mean Discrepancy (MMD), utilized in the Joint Distribution Optimal Transport (JDOT) model (Courty et al., 2017b). Although MMD effectively measures distributional divergence, it may not capture higher statistical moments, an area where the Wasserstein distance (WD) (Villani, 2008) excels. WD has been leveraged for distribution alignment in UDA methods (Courty et al., 2017a; Damodaran et al., 2018), with Redko et al. (2017) providing theoretical foundations for model generalization on the target domain when employing WD. However, the practical application of WD is computationally intensive due to the absence of a closed-form solution. The Sliced Wasserstein Distance (SWD) (Rabin et al., 2011; Bonneel et al., 2015) offers a computationally feasible alternative. Reconstruction-based objectives constitute another research direction, enforcing feature invariance across domains by reconstructing source domain data from target domain features, as in the work by Ghifary et al. (2016). Additionally, the application of Variational Autoencoders (VAEs) Kingma & Welling (2013) to UDA, such as in the Variational Fair Autoencoder (Louizos et al., 2015), showcases the capabilities of probabilistic generative models in domain-invariant feature learning. Our proposed method draws inspiration from the Variational Autoencoder's framework. Other notable approaches like ToAlign (Wei et al., 2021), SDAT (Rangwani et al., 2022), and BIWAA (Westfechtel et al., 2023), mark the recent advancement in UDA, surpassing previous models in performance. These three approaches are detailed in the experiment sections as our references for current state-of-the-art methods. However, extending these existing methods to graph-structured data is often non-trivial.

Graph Representation Learning Graph representation learning (GRL) has emerged as an important approach in machine learning, tasked with distilling complex graph-structured data into a tractable, low-dimensional vector space to enable the use of architectures developed for array-structured data. Previously, spectral methods laid the foundation, leveraging graph Laplacians to capture topological structures of graphs despite the limitations in scalability for larger graphs (Belkin & Niyogi, 2003; Chung, 1997). The field then evolved with algorithms such as DeepWalk (Perozzi et al., 2014) and Node2Vec (Grover & Leskovec, 2016), which utilized random walks to encode local neighborhood structures into node embeddings, balancing the preservation of local and global graph characteristics. The introduction of Graph Neural Networks (GNNs) marked a significant advancement in GRL. GNNs, specifically Graph Convolutional Networks (GCNs), offer a way to generalize neural network approaches to graph data, integrating neighborhood information into node embeddings (Kipf & Welling, 2016a). This was further refined by GraphSAGE, which scaled GNNs by learning a function to sample and aggregate local neighborhood features (Hamilton et al., 2017b;a). Moreover,

Graph Attention Networks (GATs) introduced an attention mechanism, enabling the model to adaptively prioritize information from different parts of a node's neighborhood, thus enhancing the expressiveness of the embeddings (Velickovic et al., 2017). These advances, along with the development of graph autoencoders like VGAEs that focused on graph reconstruction from embeddings, have broadened the applications of GRL and continue to shape its trajectory (Kipf & Welling, 2016b). For a fair comparison, when we compare with methods originally not proposed for graph domain adaptation, we replace their feature extraction backbones with GAT. This is because GAT is the graph encoder we use in our approach.

3 Problem Description

We operate under the assumption that there is a source domain with labeled data and a target domain with unlabeled data. In both domains, each input instance is a graph-structured data sample. Our primary objective is to develop a predictive model for the target domain by transferring knowledge from the source domain.

Graph Classification We focus on a graph classification task, where a graph sample can be represented as $G = (X, A)$. $X \in \mathbb{R}^{n \times K}$, where n represents the number of nodes in G and K represents the dimension of the features for each node. Specifically, $x_i \in X$ corresponds to the feature associated with a node v_i. Let $A \in \mathbb{R}^{n \times n}$ denote the adjacency matrix. The matrix A encapsulates the topological structure of G. Each graph is associated with a class, and we use y to denote the ground truth label of the graph sample G. The goal is to train a model capable of classifying graphs effectively and accurately.

Source Domain Dataset We consider a fully labeled source domain dataset as $(D_s, Y_s) = (\{G_s^k\}, \{y_s^k\})$, where G_s^k is the k^{th} graph sample in D_s and y_s^k is the ground truth label of G_s^k.

Target Domain Dataset We consider that only an unlabeled target domain dataset $D_t = \{G_t^k\}$ is accessible, where G_t^k is the k^{th} graph sample in the target dataset D_t.

UDA for Graph Classification Given an unlabeled target domain dataset D_t and a fully labeled source domain dataset D_s, our goal is to develop an effective approach to transfer knowledge from the source domain graph samples to the target domain graph samples, enabling accurate classification in the target domain.

4 Proposed Method

Figure 1 visualizes a high-level description of our proposed pipeline. Our algorithm benefits from a denoising mechanism via Variational Graph Autoencoder (VGAE) and the Nuclear-norm Wasserstein discrepancy for distribution alignment. In a nutshell, our method embeds the graph-structured data from both domains into a shared feature space via VGAE with a denoising mechanism. Then, we align the distributions of both domains in this shared feature space by minimizing the Nuclear-norm Wasserstein discrepancy.

4.1 Latent Variables Construction with Denoising Mechanism

Our model adopts encoding and decoding structures from the Variational Graph Autoencoders (Kipf & Welling, 2016b), an unsupervised learning model for graph-structured data based on Variational Autoencoder concepts. Given a graph sample $G = (X, A)$ with n nodes, the graph encoder in VGAE generates a corresponding latent variable Z. $q_\phi(Z|A, X)$ is used to denote the graph encoder, characterized by the parameter ϕ. q_ϕ aims to approximate the real posterior distribution $p(Z|A, X)$. The graph decoder of a standard VGAE is represented as $P_\theta(A|Z)$, defined by parameters θ. The prior distribution is denoted by $P(Z)$. The prior distribution is assumed to be a normal distribution, specifically $P(Z) \sim \mathcal{N}(0, I)$. The evidence lower bound (ELBO) for standard VGAE is given as:

$$\mathcal{L}_{\text{ELBO}}(\phi, \theta) = E_{q_\phi(Z|A,X)}[\log P_\theta(A|Z)] - D_{KL}(q_\phi(Z|A, X)||P(Z)) \tag{1}$$

where $D_{KL}(q(\cdot)||P(\cdot))$ represents the Kullback-Leibler divergence between distributions $q(\cdot)$ and $P(\cdot)$. The standard VGAE is trained through maximizing the $\mathcal{L}_{\text{ELBO}}$. Similar to the training process of VGAE, however, we have some variations. Given a graph sample $G = (X, A)$, we train the VGAE on both $G = (X, A)$ and

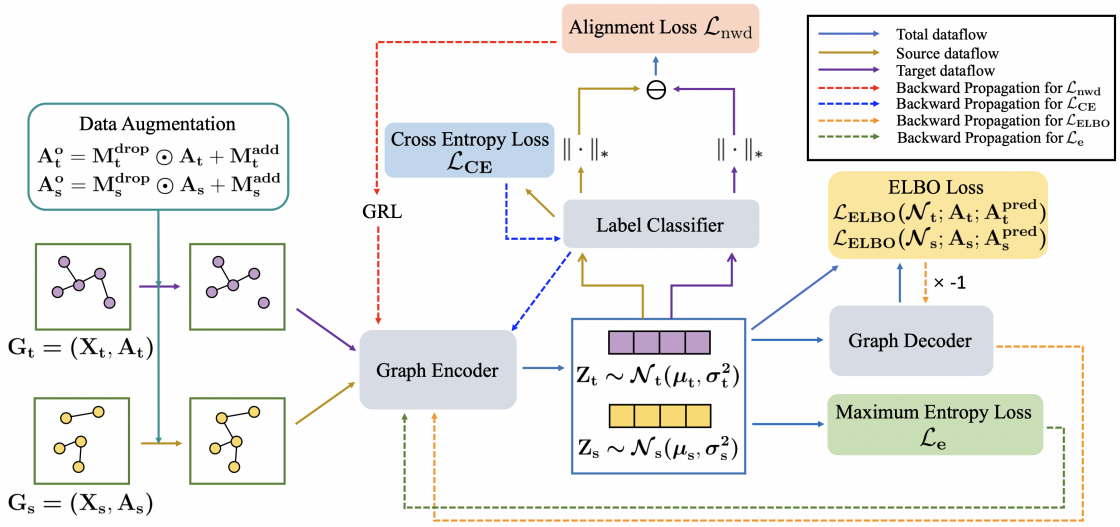

Figure 1: The block-diagram visualization of DNAN: We first add noise to the graph samples of the source and target domains by applying data augmentation to their adjacency matrices A_s and A_t, using masks M_t^{drop}, M_s^{drop}, M_s^{add}, and M_t^{add}. Then, the graph encoder of the VGAE produces the latent variable Z_s and Z_t from node features X_s, X_t and augmented adjacency matrices A_s^o, A_t^o. We train a label classifier using a cross-entropy loss $\mathcal{L}_{\text{CE}}$ between the output of the label classifier and the ground-truth label. To align the latent variables of both domains, we compute a Nuclear-norm Wasserstein discrepancy (NWD) using Z_s, Z_t, and the label classifier. The graph decoder of VGAE reconstructs the original adjacency matrices A_s, A_t from Z_s, Z_t. Then, the Evidence Lower Bound (ELBO) loss is computed based on the outputs of the graph encoder and original and reconstructed adjacency matrices. Lastly, the model applies maximum entropy regularization $\mathcal{L}_e$ to the latent variables Z_t, Z_s.

$G_o(X, A^o)$, where A^o is the adjacency matrix with noise. We implement data augmentation to add noise to the adjacency matrices. Specifically, we benefit from a random manipulation-based approach (Cai et al., 2021). To this end, edges are dropped and added randomly by modifying the values in the adjacency matrix A of the original graph. A^o is constructed as follows:

$$A^o = M^{\text{drop}} \odot A + M^{\text{add}}, \qquad m_{ij}^{\text{add}} \sim \text{Bernoulli}(p^{\text{add}} \cdot p^{\text{edge}}), \qquad m_{ij}^{\text{drop}} \sim \text{Bernoulli}(p^{\text{drop}}) \tag{2}$$

where p^{add}, p^{edge}, and p^{drop} denote the edge addition rate, the sparsity of the adjacency matrix A, and the edge dropping rate. $\odot$ represents the element-wise multiplication between two matrices. M^{drop} and M^{add} represent mask matrices with the same dimensions as A. For each element $m_{ij}^{\text{add}} \in M^{\text{add}}$ or $m_{ij}^{\text{drop}} \in M^{\text{drop}}$, we sample its value from a Bernoulli distribution.

In a standard VGAE setup, if the input is A^o, then the VGAE reconstructs A^o. However, in our approach, we aim to reconstruct the original adjacency matrix before adding noise. In other words, we reconstruct A given either $G = (X, A)$ or $G^o = (X, A^o)$. This way, we train the VGAE to function as a sophisticated filter purifying the data. This purification is crucial for aligning the source and target domain distributions. The process can be likened to enhancing one's vision in dense fog: the graph encoder, through training, becomes proficient at distinguishing unwanted random information and can reliably identify and extract the core features fundamental to source and target domains. These core features, distilled by the denoising process, are then used to align the distributions of the source and target domains. By focusing on these essential features, we hypothesize that the distribution alignment can be based on the intrinsic similarities of the source and target domains rather than on noisy information not representative of the actual distributions. Following this idea, we modify the evidence lower bound when the input is A^o to be:

$$\mathcal{L}_{\text{ELBO}} = E_{q_\phi(Z|A^o,X)}[\log P_\theta(A|Z)] - D_{KL}(q_\phi(Z|A^o, X)||P(Z)) \tag{3}$$

by replacing $P_\theta(A^o|Z)$ with $P_\theta(A|Z)$. We utilize Graph Attention Networks (GAT) as the graph encoder q_ϕ of the VGAE. The encoding equations when the input is A^o are given as follows:

$$\begin{aligned} \mu &= \mathrm{GAT}_\mu(A^o, X) \\ \log\sigma &= \mathrm{GAT}_\sigma(A^o, X) \\ z_i &= \mu_i + \varepsilon_i \cdot \sigma_i,\ \varepsilon_i \sim \mathcal{N}(0,1) \\ q_\phi(z_i|A^o, X) &= \mathcal{N}(z_i|\mu_i, \mathrm{diag}(\sigma_i^2)) \\ q_\phi(Z|A^o, X) &= \prod_{i=1}^{n} q_\phi(z_i|A^o, X) \end{aligned} \tag{4}$$

The element z_i corresponds to the i^{th} row of Z. This same row-wise correspondence applies to μ_i and $\log\sigma_i$ as well. Using the reparameterization trick, we transform the generated μ_i and σ_i into latent variable z_i. To achieve a cleaner construction of the latent variable, we apply an element-wise maximum entropy loss, $\mathcal{L}_e$, as a regularization term. This maximum entropy loss removes irrelevant information from the latent variable, enhancing its clarity and effectiveness. The specifics of the maximum entropy loss are described below:

$$\begin{aligned} \mathcal{L}_\mathrm{e} &= \frac{1}{N_s + N_t} \sum_{G^k \in (D_s, D_t)} \mathrm{ME}(Z^k) \\ \mathrm{ME}(Z^k) &= \frac{1}{n_k \times F} \sum_{i=1}^{n_k} \sum_{j=1}^{F} \sigma(z_{ij}) \log\sigma(z_{ij}) \end{aligned} \tag{5}$$

where n_k is the number of nodes in the graph sample G^k and F denotes the dimension of G^k's latent variable Z^k. After obtaining the latent variable Z, an inner product decoder $P_\theta(A|Z)$ is applied to Z to reconstruct the adjacency matrix before data augmentation. This decoder translates each pair of node representations into a binary value, indicating whether an edge exists in the reconstructed adjacency matrix A'. Specifically, we first use an MLP (multilayer perceptron) described by parameters $\{W_0, W_1\}$ to improve the expressive capacity of the latent variable Z. Then, we compute the dot product for each node representation pair as:

$$\begin{aligned} H &= \mathrm{ReLU}((Z \cdot W_0) \cdot W_1) \\ p(A'_{ij} = 1|h_i, h_j) &= \sigma(h_i^T h_j) \\ p(A'|Z) &= \prod_{i=1}^{n} \prod_{j=1}^{n} p(A'_{ij}|h_i, h_j) \end{aligned} \tag{6}$$

where h_i represents the i^{th} row of H and A'_{ij} is an element of reconstructed adjacency matrix A'. The parameter θ describes the graph decoder includes $\{W_0, W_1\}$.

4.2 Distribution Alignment

By introducing the new evidence lower bound loss, the graph encoder of VGAE is better equipped to grasp the essential features of both domains. However, we still face a crucial challenge: addressing the performance degradation that occurs when a model trained on data from a source domain is applied to a target domain with a different data distribution. As mentioned in the previous sections, traditional approaches in unsupervised domain adaptation often use a domain discriminator that engages in a min-max game with a feature extractor to produce domain-invariant features. However, these methods primarily focus on confusing features at the domain level, which might negatively impact class-level information and lead to the mode collapse problem (Kurmi & Namboodiri, 2019; Tang & Jia, 2020). To address these challenges, our approach integrates the Nuclear-norm Wasserstein discrepancy (NWD) (Chen et al., 2022) to effectively align the source and target domains' feature representations while maintaining class-level discrimination by considering it as a loss function. Our method utilizes a Variational Graph Autoencoder described in the previous section and a classifier C with parameter θ_c. We construct C with two fully connected layers. The classifier C serves two purposes. As a classifier, it distinguishes between categories, and as a discriminator,

it aligns features to close the domain gap. The empirical NWD loss is defined as:

$$\mathcal{L}_{\text{nwd}} = \frac{1}{N_s^{\text{train}}} \sum_{k=1}^{N_s^{\text{tran}}} \|C(Z_s^k)\|_* - \frac{1}{N_t^{\text{train}}} \sum_{k=1}^{N_t^{\text{train}}} \|C(Z_t^k)\|_* \tag{7}$$

where Z_s^k represents the latent variable for the graph sample G_s^k and Z_t^k represents the latent variable for the graph sample G_t^k. $\|\cdot\|_*$ denotes the Nuclear norm. N_s^{train} is the number of training samples in the source dataset, and N_t^{train} is the number of training samples in the target dataset. To avoid complex alternating updates, we employ a Gradient Reverse Layer (GRL) (Ganin et al., 2016), which allows for updating in a single backpropagation step. The distribution alignment is achieved through a min-max game, optimized as:

$$\min_{\phi} \max_{\theta_c} \mathcal{L}_{\text{nwd}} \tag{8}$$

Algorithm 1 DNAN Method

Input: $(D_s, Y_s), D_t$
Parameters: VGAE parameters $\{\phi$ (Graph Encoder), θ (Graph Decoder)$\}$, Classifier parameter $\{\theta_c\}$
Output: Trained Parameters ϕ, θ, θ_c

1: Randomly sample a batch of $\{(G_s^k, y_s^k)\}$
2: Randomly sample a batch of $\{G_k^t\}$
3: Forward Propagation
4: Update ϕ, θ, θ_c based on Equation (10)
5: Add noise to $\{G_s^k\}, \{G_t^k\}$ based on Equation (2)
6: Forward Propagation
7: Update ϕ, θ, θ_c based on Equation (10)
8: **return** ϕ, θ, θ_c

4.3 Algorithm Summary

In addition to distribution alignment, to ensure accurate classification, we optimize the graph encoder in VGAE and the classifier C using a supervised classification loss $\mathcal{L}_{\text{cls}}$ for the source domain:

$$\mathcal{L}_{\text{cls}} = \frac{1}{N_s^{\text{train}}} \sum_{j=1}^{N_s^{\text{train}}} \mathcal{L}_{\text{CE}}(C(Z_s^k, y_s^k)) \tag{9}$$

Then, by combining all the loss described in the previous sections, our total optimization object is formulated as follows:

$$\min_{\phi,\theta,\theta_c} \left\{\mathcal{L}_{\text{cls}} - \mathcal{L}_{\text{ELBO}} + \lambda_e \mathcal{L}_{\text{e}}\right\} + \min_{\phi} \max_{\theta_c} \mathcal{L}_{\text{nwd}}, \tag{10}$$

where ϕ is the parameter of the graph encoder of the VGAE, θ is the parameter of the graph decoder of the VGAE, θ_c is the parameter of the classifier and λ_e is a hyperparameter that weighs the maximum entropy loss $\mathcal{L}_{\text{e}}$. It is worth noting that we balance the supervised classification loss and the NWD loss equally. In this case, our model effectively learns transferable and distinct features, leading to accurate and diverse predictions in the target domain. The complete procedures of our UDA approach for graph-structured data are summarized in Algorithm 1.

5 Experimental Validation

We validate our algorithm using two graph classification benchmarks. Our code is provided as a supplement.

5.1 Experimental Setup

Datasets We use the IMDB&Reddit Dataset (Yanardag & Vishwanathan, 2015) and the Ego-network Dataset (Qiu et al., 2018) in our experiments. Following the previous works, we include Coreness (Batagelj & Zaversnik, 2003), Pagerank (Page et al., 1999), Eigenvector Centrality (Bonacich, 1987), Clustering Coefficient (Watts & Strogatz, 1998), and Degree/Rarity (Adamic & Adar, 2003) as the node features for both datasets.

IMDB&Reddit Dataset IMDB&Reddit consists of the IMDB-Binary (1000 samples) and Reddit-Binary (2000 samples) datasets, each denoting a single domain.

- **IMDB-BINARY** Each graph in this dataset represents an ego network for an actor/actress. Nodes correspond to actors/actresses. There will be an edge between two actors/actresses who appear in the same movie. A graph is generated from either romance or action movies. The task is to classify the graph into romance or action genres.
- **REDDIT-BINARY** Each graph represents an online discussion thread. Nodes correspond to users. If one of the users has responded to another's comments, then an edge exists between them. The discussion threads are drawn from four communities: AskReddit and IAmA are question/answer-based communities. Atheism and TrollXChromosomes are discussion-based communities. The binary classification task is to classify a graph into discussion-based or question/answer-based communities.

Ego-network Dataset Ego-network consists of data from four social network platforms, Digg, OAG, Twitter, and Weibo, each representing a domain. Each network is modeled as a graph. Each graph has 50 nodes, and nodes in the graphs represent users. Every graph has an ego user. An edge is drawn between two nodes if a social connection occurs between two users. The definitions of social connection of these four social network platforms are different. We extract the descriptions of social connections and social actions of each social network according to Qiu et al. (2018):

- **Digg** allows users to vote for web content such as stories and news (up or down). The social connection is users' friendship, and the social action is voting for the content.
- **OAG** is generated from AMiner and Microsoft Academic Graph. The social connection is represented as the co-authorship of users, and the social action is the citation behavior.
- **Twitter** currently known as X, the social connection on Twitter represents users' friendship, and the social action is posting tweets related to the Higgs boson, a particle discovered in 2012.
- **Weibo** is a social platform similar to Twitter. The Weibo dataset includes posting logs between September 28th, 2012, and October 29th, 2012, among 1,776,950 users. The social connection is defined as users' friendship, and the social action is re-posting messages on Weibo.

All the graphs in the four domains are labeled as active or inactive, the ego user's action status. If the user makes the social action, then the user is active. The task is to identify whether the ego users are active or inactive. In addition to previously referenced node features, Ego-nework dataset also contains DeepWalk embeddings for each node (Perozzi et al., 2014), the number/ratio of active neighbors (Backstrom et al., 2006), the density of subnetwork induced by active neighbors (Ugander et al., 2012), and the number of connected components formed by active neighbors (Ugander et al., 2012).

Baselines for Comparison There is a limited number of UDA algorithms specifically designed for graph classification tasks. As a result, we conduct a comparative analysis between our proposed method and updated versions of several representative methods (DANN, MDD, DIVA) and current state-of-the-art UDA methods (SDAT, BIWAA, ToAlign) for array-structured data. To facilitate the adaptation of these algorithms to graph-structured data and consistent with the structure of our VGAE's graph encoder, we replace the feature extraction backbones originally designed for array-structured data with GATs. These methods are explained below:

- **Sources:** Plain GATs trained without domain adaptation techniques.
- **DANN:** Domain Adversarial Neural Network (DANN) (Ganin et al., 2016) adopts an adversarial learning strategy. It contains a domain classifier. The domain classifier tries to distinguish the samples from which domain and the feature extractor aims to confuse the domain classifier.
- **MDD:** Margin Disparity Discrepancy (MDD) (Zhang et al., 2019b) is first proposed for computer vision tasks. It measures the distribution discrepancy and is tailored to the minimax optimization for training.
- **DIVA:** Domain Invariant Variational Autoencoders (DIVA) (Ilse et al., 2020) disentangles the inputs into three latent variables, domain latent variables, semantic latent variables, and residual variations latent variables. It is proposed to solve problems in fields such as medical imaging.
- **SDAT:** Smooth Domain Adversarial Training (SDAT) (Rangwani et al., 2022) focuses on achieving smooth minima with respect to classification loss, which stabilizes adversarial training and improves the performance on the target domain.
- **BIWAA:** Backprop Induced Feature Weighting for Adversarial Domain Adaptation with Iterative Label Distribution Alignment (BIWAA) (Westfechtel et al., 2023) employs a classifier-based backprop-induced weighting of the feature space, allowing the domain classifier to concentrate on features that are important for classification and coupling the classification and adversarial branch more closely.
- **ToAlign:** Task-oriented Alignment for Unsupervised Domain Adaptation (ToAlign) (Wei et al., 2021) decomposes features in the source domain into classification task-related and classification task-irrelevant parts under the guidance of classification meta-knowledge, ensuring that the domain adaptation is beneficial for the performance on the classification task.

Evaluation Metrics Following the literature, we use F1-Score to account for imbalance in the datasets.

Training Scheme In our evaluation, we rigorously train five models for each baseline method by employing five distinct random seeds for parameter initialization and dropping or adding edges during the data augmentation phase. We report both the average performance and standard deviation of the obtained F1-score. To ensure a fair comparison across all methods, we maintain the same seed for data shuffling. The optimization process uses the Adam (Kingma & Ba, 2014) optimizer. Please refer to the Appendix for a comprehensive description of our training scheme.

5.2 Performance Results and Comparison

Tables 1 and 2 present our performance results. The bold font denotes the highest performance in each column.

Table 1: Performance results on Ego-network Dataset

Method	O→T	O→W	O→D	T→O	T→W	T→D	W→O	W→T	W→D	D→O	D→T	D→W	Avg
Source	$40.0_{\pm 0.0}$	$40.4_{\pm 0.3}$	$43.8_{\pm 2.4}$	$40.2_{\pm 0.0}$	**48.0**$_{\pm 1.2}$	$41.3_{\pm 0.0}$	$40.2_{\pm 0.0}$	$46.6_{\pm 0.9}$	$41.3_{\pm 0.1}$	$40.2_{\pm 0.0}$	$40.0_{\pm 0.0}$	$39.8_{\pm 0.0}$	41.8
DANN	$42.0_{\pm 0.6}$	$41.7_{\pm 0.6}$	$51.3_{\pm 0.8}$	$40.7_{\pm 0.7}$	$42.0_{\pm 0.9}$	$49.9_{\pm 1.7}$	$40.3_{\pm 0.1}$	$41.3_{\pm 0.6}$	$50.9_{\pm 0.4}$	$40.3_{\pm 0.0}$	$40.4_{\pm 0.3}$	$42.5_{\pm 0.7}$	43.6
MMD	$40.2_{\pm 0.1}$	$41.2_{\pm 1.1}$	$48.0_{\pm 3.2}$	$40.2_{\pm 0.0}$	$45.5_{\pm 1.5}$	$41.3_{\pm 0.0}$	$40.2_{\pm 0.0}$	$46.2_{\pm 2.9}$	$41.9_{\pm 1.3}$	$40.2_{\pm 0.0}$	$40.1_{\pm 0.1}$	$40.0_{\pm 0.2}$	42.1
DIVA	$42.1_{\pm 0.5}$	$42.4_{\pm 1.4}$	$48.9_{\pm 0.7}$	$40.3_{\pm 0.2}$	$42.0_{\pm 0.4}$	$50.3_{\pm 0.5}$	$40.4_{\pm 0.3}$	$41.2_{\pm 0.3}$	$48.7_{\pm 0.9}$	**40.6**$_{\pm 0.5}$	$41.3_{\pm 0.4}$	$42.5_{\pm 0.5}$	43.4
SDAT	$40.2_{\pm 0.1}$	$40.1_{\pm 0.5}$	$42.2_{\pm 1.6}$	$40.2_{\pm 0.0}$	$41.6_{\pm 1.3}$	$42.9_{\pm 3.2}$	$40.3_{\pm 0.2}$	$41.1_{\pm 0.7}$	$43.1_{\pm 2.8}$	$40.2_{\pm 0.0}$	$40.1_{\pm 0.1}$	$39.9_{\pm 0.1}$	41.0
BIWAA	$40.1_{\pm 0.2}$	$41.5_{\pm 0.2}$	$45.1_{\pm 1.3}$	$40.3_{\pm 0.2}$	**48.0**$_{\pm 2.9}$	$43.3_{\pm 3.0}$	$40.2_{\pm 0.0}$	**50.4**$_{\pm 0.5}$	$45.7_{\pm 3.8}$	$40.3_{\pm 0.1}$	$41.0_{\pm 0.9}$	**43.1**$_{\pm 0.3}$	43.2
ToAlign	$36.5_{\pm 13.3}$	$43.0_{\pm 0.9}$	$49.1_{\pm 1.4}$	**40.8**$_{\pm 0.3}$	$43.1_{\pm 1.1}$	$50.2_{\pm 0.5}$	$40.7_{\pm 0.4}$	$42.8_{\pm 1.3}$	$48.4_{\pm 3.2}$	$40.5_{\pm 0.2}$	$43.4_{\pm 0.9}$	$42.6_{\pm 1.6}$	43.4
DNAN	**42.9**$_{\pm 1.6}$	**43.4**$_{\pm 1.2}$	**53.7**$_{\pm 0.6}$	**40.8**$_{\pm 0.2}$	$45.3_{\pm 3.0}$	**53.9**$_{\pm 1.0}$	**40.8**$_{\pm 0.4}$	$48.6_{\pm 0.8}$	**53.4**$_{\pm 2.9}$	**40.6**$_{\pm 0.2}$	**44.1**$_{\pm 1.5}$	$42.8_{\pm 0.9}$	**45.9**

Table 2: Perofmrance results on IMDB&Reddit Dataset

Task	Source	DANN	MMD	DIVA	SDAT	BIWAA	ToAlign	DNAN
I→R	$63.4_{\pm 0.2}$	$63.9_{\pm 0.8}$	$63.7_{\pm 0.4}$	$63.6_{\pm 0.5}$	$63.6_{\pm 0.6}$	$64.0_{\pm 0.8}$	$63.3_{\pm 0.2}$	$\mathbf{64.2}_{\pm 0.6}$
R→I	$72.3_{\pm 1.7}$	$72.0_{\pm 1.7}$	$73.6_{\pm 1.7}$	$71.1_{\pm 0.3}$	$74.1_{\pm 2.0}$	$71.4_{\pm 1.0}$	$73.4_{\pm 0.8}$	$\mathbf{74.9}_{\pm 2.0}$
Avg	67.8	68.0	67.3	68.0	68.8	67.7	68.3	**69.6**

Ego-network Results Results for this dataset are presented in Table 1. In this benchmark, twelve UDA tasks can be defined by pairing the four domains. Our experimental results indicate that the DNAN performs the best on average and achieves state-of-the-art performance on nine tasks: O to T, O to W, O to D, T to O, T to D, W to O, W to D, D to O, and D to T. DNAN has good performances on T to W and W to T, and achieve SOTA performance on D to T and T to D tasks, showing that DNAN can successfully handle similar domains, as Digg, Twitter, and Weibo are similar content-sharing platforms. Notably, it exceeds the second-best methods by about 4% on T to D and about 3% on W to D. In addition, DNAN can also achieve SOTA performance when there is a large distribution gap between domains, such as on tasks between OAG and Twitter or OAG and Weibo. It is important to underscore that no single method can achieve the best performance on all tasks, likely due to the diverse range of domain gaps.

IMDB&Reddit Results Results for this dataset are presented in Table 2. We note that the experiments demonstrate that all UDA methods perform better on the Reddit to IMDB task (R to I) than on the IMDB to Reddit task (I to R), indicating that the two tasks are not equally challenging. We hypothesize that the smaller size of IMDB-Binary compared to Reddit-Binary may result in performance degradation when testing on the larger Reddit-Binary dataset, as less knowledge can be transferred to the target domain. Our experiment results show that DNAN outperforms all other methods on both the R to I and I to R tasks, leading to SOTA results on average. On the I to R task, we observe that all methods perform similarly. Though DNAN does not outperform other methods by a large margin on the I to R task, the results still indicate that DNAN has a competitive performance compared to other methods. It is worth noting that the performance of a UDA algorithm may vary to some extent based on hyperparameter tuning. Therefore, when comparing two UDA algorithms with similar performance, they should be considered equally competitive. Based on this consideration, we can conclude that our proposed method performs competitively on all the UDA tasks and outperforms other UDA methods on average. These findings suggest that DNAN can serve as a robust UDA algorithm.

5.3 Analytic and Ablative and Experiments

We first perform analytic experiments to offer a deeper insight into our approach. We then performed an ablative experiment to demonstrate that all components in our algorithm are important to achieve optimal performance.

The effect of DNAN on data representations in the output space of the classifier To evaluate the effectiveness of our proposed approach, we analyze how DNAN influences the target domain's distribution in the classifier's output space on the Oag to Weibo task (O to W). We picked this task to demonstrate the effect of our model because it is challenging as Weibo and Oag are very dissimilar platforms; one is connected by co-authorship, and the other is friendship. We utilize the UMAP (McInnes et al., 2018) visualization tool and compare the representations of the source domain's test data, the target domain's test data before using DNAN, and the target domain's test data after applying DNAN. In Figure 2, each point represents a single data point in the output space of the classifier before the softmax activation. Blue and red colors denote the two classes. In Figure 2, the middle plot shows that the classifier doesn't work well with the target domain data before adaptation. It's hard to distinguish the class boundary as red dots are mixed with blue ones. However, after applying DNAN, the class boundary becomes clearer, and the data representation distribution of the target domain matches well with the source domain. This is evident in the left and right plots of Figure 2, where the patterns of dots are consistent. These visualization results demonstrate that DNAN successfully mitigates the performance degradation caused by the domain shift.

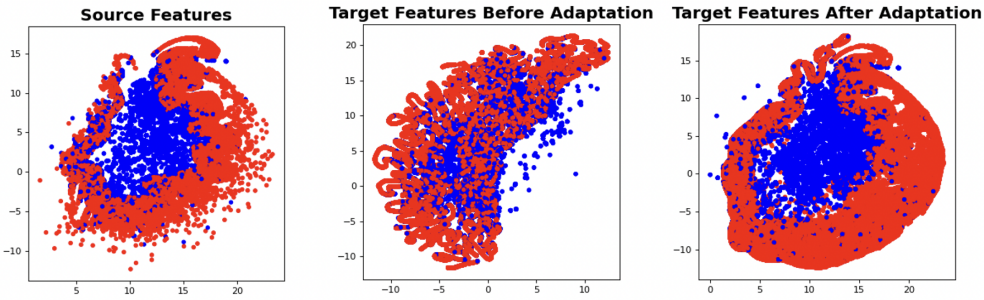

Figure 2: UMAP visualizations showing the test data representations before softmax activation on the Oag to Weibo task. Blue and red points denote the different classes. The middle plot displays the target domain data representations obtained from a model trained on the source dataset before adaptation. The left and right plots show the source and target domain data representations after adaptation using DNAN.

Table 3: Ablation Study Results on Ego-network Dataset

Method	O→T	O→W	O→D	T→O	T→W	T→D	W→O	W→T	W→D	D→O	D→T	D→W	Avg
DNAN-D	$42.8_{\pm 1.3}$	$42.5_{\pm 1.7}$	$52.3_{\pm 2.6}$	$40.5_{\pm 0.2}$	**46.9**$_{\pm 2.0}$	$50.0_{\pm 3.5}$	$40.4_{\pm 0.2}$	**50.1**$_{\pm 0.6}$	$52.8_{\pm 2.4}$	**40.6**$_{\pm 0.2}$	$42.1_{\pm 1.3}$	$42.5_{\pm 1.8}$	45.4
DNAN-N	**44.4**$_{\pm 2.2}$	$43.1_{\pm 0.7}$	$52.6_{\pm 1.0}$	**41.0**$_{\pm 0.3}$	$45.0_{\pm 2.4}$	$52.7_{\pm 2.7}$	$40.7_{\pm 0.3}$	$46.9_{\pm 2.2}$	$53.3_{\pm 2.5}$	$40.5_{\pm 0.2}$	$43.0_{\pm 1.1}$	**43.6**$_{\pm 0.9}$	45.6
DNAN	$42.9_{\pm 1.6}$	**43.4**$_{\pm 1.2}$	**53.7**$_{\pm 0.6}$	$40.8_{\pm 0.2}$	$45.3_{\pm 3.0}$	**53.9**$_{\pm 1.0}$	**40.8**$_{\pm 0.4}$	$48.6_{\pm 0.8}$	**53.4**$_{\pm 2.9}$	**40.6**$_{\pm 0.2}$	**44.1**$_{\pm 1.5}$	$42.8_{\pm 0.9}$	**45.9**

Table 4: Ablation Study Results on IMDB&Reddit Dataset

Task	DNAN-D	DNAN-N	DNAN
I to R	$63.8_{\pm 0.4}$	$64.0_{\pm 0.5}$	**64.2**$_{\pm 0.6}$
R to I	$72.3_{\pm 2.4}$	$74.2_{\pm 1.8}$	**74.9**$_{\pm 2.0}$
Avg	68.0	69.0	**69.6**

Ablative study The ablation experiments are conducted to demonstrate the effectiveness of the two ideas we benefit from to develop DNAN. To this end, we remove one of the two components at a time and report our performance. We denote the ablated versions of DNAN as: (i) **DNAN-D**: We exclude the denoising mechanism and only apply NWD loss. and (ii) **DNAN-N**: We exclude NWD loss and only apply the denoising mechanism.

Our ablation study results for the Ego-network and the IMDB&Reddit datasets are illustrated in Table 3 and 4, respectively. The Ego-network dataset results reveal that the integration of both NWD loss and the denoising mechanism (DNAN) yields the highest average performance at 45.9%. The DNAN-D configuration, lacking the denoising mechanism, shows competitive performance with an average of 45.4%. However, the DNAN-N configuration, which excludes NWD loss, displays an even smaller decrease in performance, with an average of 45.6%. For the IMDB&Reddit dataset, the full DNAN model again demonstrates superior performance with an average score of 69.6%. Interestingly, the DNAN-N variant outperforms DNAN-D with averages of 69.0% and 68.0%, respectively. This observation indicates that the denoising mechanism is more critical in this context. The ablation study highlights the importance of both the denoising mechanism and NWD loss in our proposed method. While the NWD loss and the denoising techniques contribute more evenly to the Ego-network dataset, the denoising mechanism is more beneficial for the IMDB&Reddit dataset. This suggests that the effectiveness of each component is context-dependent, and future work may explore this dependency in greater depth. The cooperating effect of combining both techniques confirms the robustness of our DNAN model, as it consistently outperforms its counterparts with either component excluded.

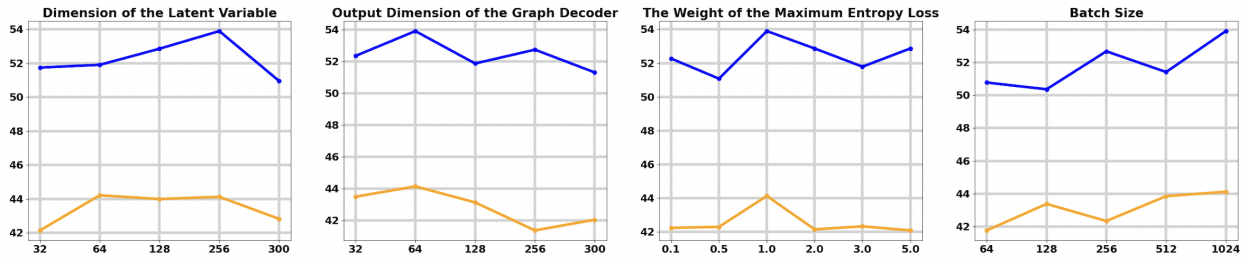

Figure 3: The performance of DNAN with different hyperparameter settings on Twitter to Digg (Blue lines) and Digg to Twitter (Yellow lines) tasks.

5.4 Hyperparameters Sensitivity Analysis

An important concern for most algorithms is tuning the hyperparameters and measuring the performance sensitivity with respect to them. We evaluate the sensitivity of DNAN with respect to various hyperparameters on two tasks: Twitter to Digg (T to D) and Digg to Twitter (D to T). We varied the dimension of the latent embedding space, the output dimension of the graph decoder, the weight of the maximum entropy loss ($\mathcal{L}_e$), and the batch size. We present the F1-scores of the DNAN model as a linear function of these hyperparameters in Figure 3, with the blue lines representing the T to D task and the yellow lines representing D to T task. Through inspecting this figure, we deduce:

- **Dimension of the Latent Embedding Space:** We test the performance of DNAN on five different dimension sizes for the embedding space: 32, 64, 128, 256, and 300. The performance of DNAN peaks at a latent variable size of 256 for the T to D task and a less pronounced peak on the D to T task, indicating that a moderately large value for the dimension of the latent variable is beneficial for capturing the salient features of the data. Performance declines when the dimension is too small to capture the complexity or too large, potentially introducing noise or overfitting. We note, however, that the result indicates that the performance remains relatively decent for a wide range of embedding sizes.
- **Output Dimension of the Graph Decoder:** Similar to the experiments on the dimension of the latent variable, we test the performance of DNAN on five dimension sizes: 32, 64, 128, 256, and 300. The output dimension of the graph decoder shows a performance peak at 64 for the T to D task and the D to T task. This observation suggests that a moderately small representation capacity in the graph decoder is more beneficial. Compared with performances on the D to T task, the T to D task is less sensitive to this hyperparameter.
- **Weight of the $\mathcal{L}_e$:** We test the DNAN on six weights: 0.1,0.5,1.0,2.0,3.0,5.0. The weight of the maximum entropy loss presents a clear peak at 1.0 for both the T to D and D to T tasks, suggesting that a balanced contribution of the entropy loss is critical for performance.
- **Batch Size:** We test five batch sizes: 64, 128, 256, 512, 1024. For batch size, there is a trend of increasing performance as the size grows, with a notable peak at a batch size of 1024 for the T to D and D to T tasks. This implies that the performance of DNAN benefits from larger batch sizes, possibly due to more stable gradient estimates. Compared with the T to D task, the D to T task is less affected by batch size variations.

The sensitivity analysis of hyperparameters for the DNAN model on the T to D and D to T tasks demonstrates the stability of DNAN models when using different hyperparameter values, as there is a moderate fluctuation around $\pm 3\%$. However, fine-tuning hyperparameters to the specific characteristics of the task and the dataset is beneficial. Although optimal performance is achieved with a latent variable dimension of 256, a decoder output dimension of 64, an entropy loss weight of 1.0, and a batch size of 1024, tuning the hyperparameter is not essential to achieve performance in the competitive range.

Table 5: Training time for IMDB&Reddit Dataset

Task	DANN	MMD	DIVA	SDAT	BIWAA	ToAlign	DNAN
I→R	10	260	14	12	899	10	10
R→I	3	2	5	6	34	3	5

Table 6: Model complexity on IMDB&Reddit-Binary dataset. K represents the input feature dimension, M represents the hidden dimension, and D represents the output dimension of the decoder.

DANN	MMD	DIVA	SDAT	BIWAA	ToAlign	DNAN
(K+3M+3)M	(K+4M+4)M	(K+11M+3D+3)M	(K+3M+4)M	(K+3M+3)M	(K+2M+11)M	(K+4M+D+2)M

5.5 Time Complexity and Model Complexity Analysis

The analysis of the training time (in minutes) and model complexity for various domain adaptation methods on the IMDB&Reddit dataset is presented in Table 5 and Table 6.

- **Training Time:** The training times reported in Table 5 illustrate the efficiency of the DNAN model relative to its counterparts. For the I to R task, DNAN required 10 minutes, positioning it the fastest in terms of training time, together with DANN and ToAlign, compared to other methods. In the R to I task, DNAN again demonstrated moderate efficiency with 5 minutes, with MMD being the fastest at 2 minutes and BIWAA the slowest at 34 minutes. These results suggest that DNAN provides a balanced trade-off between model performance and training efficiency without adding a significant computational overload.
- **Model Complexity:** The model complexity, as shown in Table 6, is assessed based on the number of parameters in the models, which is a function of the input feature dimension (K), hidden dimension (M), and the output dimension of the Decoder (D). Compared to other methods like DANN and SDAT, which have similar forms, DNAN introduces additional complexity due to the parameters in the graph decoder. However, it remains less complex than DIVA, which includes an extra (7M+2D+1)M term.

We conclude that the DNAN model shows competitive training time that is significantly lower than the most time-consuming method (BIWAA) while maintaining comparable or better performance. Model complexity analysis reveals that DNAN, while not the simplest, avoids the higher complexity seen in more complex methods such as DIVA. DNAN balances the computational cost with the capacity to learn and transfer knowledge effectively for better UDA performance. This observation is important because, in certain applications, it is crucial to perform UDA quickly. This is due to the constant changes in the input distribution and the limited time available to update the model.

6 Conclusions

We developed a new UDA method, which is specifically designed for graph-structured data. Our proposed method includes denoising and using the Nuclear-Norm Wasserstein discrepancy for domain alignment in a shared embedding space. The experiments demonstrate our approach to be a promising method. By innovatively combining domain alignment through Nuclear-norm Wasserstein discrepancy with a denoising mechanism via a Variational Graph Autoencoder, DNAN has outperformed state-of-the-art methods across two major benchmarks without adding significant computational overload. The ability of our method to handle subtle and significant domain differences showcases its versatility and robustness. From ablative studies, the two ideas that DNAN benefits from are proven to be crucial for optimal performance. Future work can explore extending our approach to partial domain adaptation scenarios or situations where the source domain data can not be directly accessible.

References

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 44–54, 2006.

Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Ruichu Cai, Fengzhu Wu, Zijian Li, Pengfei Wei, Lingling Yi, and Kun Zhang. Graph domain adaptation: A generative view. *arXiv preprint arXiv:2106.07482*, 2021.

Lin Chen, Huaian Chen, Zhixiang Wei, Xin Jin, Xiao Tan, Yi Jin, and Enhong Chen. Reusing the task-specific classifier as a discriminator: Discriminator-free adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7181–7190, 2022.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE TPAMI*, 39(9):1853–1865, 2017a.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *Advances in neural information processing systems*, 30, 2017b.

Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3941–3950, 2020.

Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Fast batch nuclear-norm maximization and minimization for robust domain adaptation. *arXiv preprint arXiv:2107.06154*, 2021.

B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. *arXiv preprint arXiv:1803.10081*, 2018.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Zhiqiang Gao, Shufei Zhang, Kaizhu Huang, Qiufeng Wang, and Chaoliang Zhong. Gradient distribution alignment certificates better adversarial domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8937–8946, 2021.

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pp. 597–613. Springer, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017a.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pp. 322–348. PMLR, 2020.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4893–4902, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Vinod Kumar Kurmi and Vinay P Namboodiri. Looking back at labels: A class based domain adaptation technique. In *2019 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *Advances in neural information processing systems*, 29, 2016.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

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

Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2110–2119, 2018.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 435–446. Springer, 2011.

Harsh Rangwani, Sumukh K Aithal, Mayank Mishra, Arihant Jain, and R. Venkatesh Babu. A closer look at smoothness in domain adversarial training. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical analysis of domain adaptation with optimal transport. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part II 10*, pp. 737–753. Springer, 2017.

Hui Tang and Kui Jia. Discriminative adversarial domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5940–5947, 2020.

Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. Structural diversity in social contagion. *Proceedings of the national academy of sciences*, 109(16):5962–5966, 2012.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684): 440–442, 1998.

Guoqiang Wei, Cuiling Lan, Wenjun Zeng, Zhizheng Zhang, and Zhibo Chen. Toalign: task-oriented alignment for unsupervised domain adaptation. *Advances in Neural Information Processing Systems*, 34:13834–13846, 2021.

Thomas Westfechtel, Hao-Wei Yeh, Qier Meng, Yusuke Mukuta, and Tatsuya Harada. Backprop induced feature weighting for adversarial domain adaptation with iterative label distribution alignment. *Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pp. 1457–1467, 2020.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. Dane: Domain adaptive network embedding. *arXiv preprint arXiv:1906.00684*, 2019a.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pp. 7404–7413. PMLR, 2019b.

A Nuclear-norm Wasserstein Discrepancy

From Intra-class and Inter-class Correlations to Domain Discrepancy Consider a prediction matrix $P \in \mathbb{R}^{b\times k}$ predicted by classifier C, where b represents the number of samples and k represent the number of classes. P has the following properties:

$$\sum_{j=1}^{k} P_{ij} = 1,\ P_{ij} \geq 0,\ \forall i \in 1, 2, ...b \tag{11}$$

The self-correlation matrix $R \in \mathbb{R}^{k\times k}$ then can be computed by $R = Z^T Z$. The intra-class correlation I_a is defined as the sum of the main diagonal elements in R, and the inter-class correlation I_e is defined as the sum of the off-diagonal elements in R:

$$I_a = \sum_{i,j=1}^{k} R_{ij},\ I_e = \sum_{i\neq j}^{k} R_{ij} \tag{12}$$

The I_a and I_e are very different for source and target domains. For the source domain, the I_a is large while the I_e is relatively small, as we train with labels available so that most samples are correctly classified. For the target domain, the I_a is small while the I_e is relatively large due to the lack of supervised training. Based on linear algebra, we can represent $I_a = \|P\|_F$, the Frobenius norm of P, and

$$I_a - I_e = 2\|P\|_F - b \tag{13}$$

For the source domain, $I_a - I_e$ will be large; for the target domain, $I_a - I_e$ will be small. Therefore, $I_a - I_e$ can represent the discrepancy between two domains. Since the prediction matrix P is generated by the classifier C, we can rewrite $P = C(Z)$, where Z is the feature representation of the sample from either the source or the target domain. With inspiration from WGAN (Arjovsky et al., 2017) and 1-Wasserstein distance, the domain discrepancy can be formally formulated as

$$W_F(D_s, D_t) = \sup_{\|\|C\|_F\|_L \leq K} \mathbb{E}_{Z_s\sim D_s}[\|C(Z_s)\|_F] - \mathbb{E}_{Z_t\sim D_t}[\|C(Z_t)\|_F] \tag{14}$$

We call $W_F(D_s, D_t)$ the Frobenius norm-based 1-Wasserstein distance, where D_s denotes the source domain, D_t denotes the target domain, $\|\cdot\|_L$ denotes the Lipschitz semi-norm (Villani et al., 2009), and K denotes the Lipschitz constant.

From Frobenius Norm to Nuclear Norm From the domain discrepancy formulated above, we can see that the classifier C works like a discriminator in GAN. Therefore, we can perform adversarial training to train the feature generator via $W_F(D_s, D_t)$. However, adversarial training with $W_F(D_s, D_t)$ limits the diversity of predictions. This is because it tends to push the samples in a class with fewer samples near the decision boundary closer to a neighboring class with a significantly larger number of samples far from the decision boundary (Cui et al., 2021). To address this limitation, the author proposes to use the nuclear norm instead of the Frobenius norm. The nuclear norm has been shown to be bound by the Frobenius norm (Chen et al., 2022). In addition, maximizing the nuclear norm maximizes the rank of the prediction matrix P when $\|\cdot\|_F$ is nearby $\sqrt{b}$ (Cui et al., 2020; 2021). In consequence, the diversity of predictions will be enhanced. Thus, the domain discrepancy can be improved to be

$$W_N(D_s, D_t) = \sup_{\|\|C\|_*\|_L \leq K} \mathbb{E}_{Z_s\sim D_s}[\|C(Z_s)\|_*] - \mathbb{E}_{Z_t\sim D_t}[\|C(Z_t)\|_*] \tag{15}$$

$W_N(D_s, D_t)$ is called the Nuclear-norm 1-Wasserstein discrepancy (NWD). To integrate NWD into implementation, we can approximate the empirical NWD $\bar{W}_N$ by maximizing $\mathcal{L}_{\text{nwd}}$ that is defined below

$$\mathcal{L}_{\text{nwd}} = \frac{1}{N_s}\sum_{k=1}^{N_s} \|C(Z_s^k)\|_* - \frac{1}{N_t}\sum_{k=1}^{N_t} \|C(Z_t^k)\|_*,\ \bar{W}_N(D_s, D_t) \approx \max \mathcal{L}_{\text{nwd}} \tag{16}$$

where Z_s^k is the feature representation of the kth sample in the source domain dataset and Z_t^k is the feature representation of the kth sample in the target domain dataset. N_s and N_t represent the number of samples in the source and target domain, respectively.

B Implementation Details of DNAN

In this section, we present our implementations of DNAN. Our codes are in Python, mainly with PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019) libraries. We train five models for every baseline using five random seeds for parameter initialization. The five random seeds are 27, 28, 29, 30, and 31. We also conducted a hyperparameter search, described in the hyperparameter sensitivity section in the paper, to find suitable hyperparameters for optimal performance. The hyperparameters we use to achieve the results listed in the main paper are presented in Table 7.

Parameter	Ego-network	IMDB&Reddit
Batch size	1024	64
Learning rate	0.01	0.001
Dropout rate	0.5	0.2
Encoder hidden size	256	128
Decoder output size	64	128
Learning decay rate	0.75	0.75
Entropy weight	1.0	1.0
Weight decay	0.0005	0.0005
p_{add}	0.1	0.1
p_{drop}	0.1	0.1

Table 7: Hyper-parameters of DNAN