SCIENCE CHINA
Information Sciences

• Supplementary File •

Node Quality-Driven Unsupervised Cross-Network Node Classification: A Reinforcement Transfer Learning Approach

Hongwei Yang[1], Meng Hao[1*], Jiaoxuan Lin[1], Hui He[1], Weizhe Zhang[1,2] & Wenqi Wang[1]

[1]*School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, Heilongjiang*
[2]*Pengcheng Laboratory, Shenzhen 518055, Guangdong*

Appendix A Related Work

In this section, we review the related literature about network embedding in Section Appendix A.1, graph transfer learning in Section Appendix A.2, and reinforcement learning in Section Appendix A.3.

Appendix A.1 Network Embedding

Network structured data, as a form of nonlinear data structure, is widely present across various domains, including social media networks [1], protein-protein interaction networks in genomics [2], brain networks in neuroimaging [3], molecular networks in drug discovery [4], and bank asset networks in finance [5]. Network embedding, often referred to as graph embedding, is a technique designed to map complex graph-structured data into low-dimensional vector spaces [6], facilitating more effective graph representation learning. These low-dimensional vectors are optimized to capture intricate relationships and features among nodes in the graph while preserving the graph's structural properties. The resulting graph embeddings are highly informative and applicable to a wide range of graph analysis tasks. Depending on the specific features and structure of the graph, neural network-based embedding methods can be effectively employed for both homogeneous and heterogeneous graphs.

Homogeneous networks, also known as homogeneous information networks, provide a simplified representation of real-world graph data, characterized by a single type of node and edge. The Graph Convolutional Network (GCN) [7], a widely utilized transductive algorithm, aggregates node feature information by performing convolutions on the adjacency matrix. However, GCN's uniform treatment of neighboring nodes during information propagation limits its ability to discern the varying importance of different nodes. Conversely, the Graph Attention Network (GAT) [8] enhances the modeling of node relationships by learning attention weights between each node and its neighbors, allowing for dynamic adjustment of neighbor contributions based on their significance within the graph. Hamilton *et al.* [9] proposed GraphSAGE, a representative inductive algorithm that updates node feature representations by randomly selecting and aggregating a subset of neighboring nodes. This approach is highly scalable for large graphs and is well-suited for more complex graph structures.

Appendix A.2 Graph Transfer Learning

Graph transfer learning (GTL) [10–12] aims to leverage the similarity between a source network and a target network to facilitate knowledge transfer. By integrating existing knowledge from the source network with the learning tasks of the target network, graph transfer learning can quickly and efficiently accomplish learning tasks on the target network [13], thereby improving learning performance on the target network.

For example, Zhang *et al.* [14] proposed DANE, an unsupervised domain-adaptive network embedding model that learns transferable node embeddings by combining a GCN network with shared weights and adversarial learning regularization. Shen *et al.* [15] proposed the CDNE algorithm, which reduces network discrepancies through the minimization of Maximum Mean Discrepancy (MMD) loss and applies the learned embeddings to cross-network node classification tasks. However, CDNE's reliance on preprocessing the PPMI matrix and its exclusion of network topology and node attribute information increase computational complexity.

To address these limitations, Dai *et al.* [16] proposed AdaGCN, which integrates semi-supervised learning with adversarial domain adaptation to learn cross-network node embeddings by employing GNN to capture node feature representations. Additionally, Shen *et al.* [17] proposed the ACDNE algorithm, which combines deep network embedding and adversarial domain adaptation to learn transferable node embeddings across networks.

* Corresponding author (email: haomeng@hit.edu.cn)

Most existing methods prioritize local consistency when learning node feature representations, often neglecting the impact of global consistency on embedding quality. To address this gap, Wu *et al.* [18] proposed the UDAGCN algorithm, which integrates both local and global consistency to enhance node embeddings and leverages a graph-based attention mechanism along with domain adaptation to mitigate cross-domain discrepancies in node classification. Yang *et al.* [19] proposed the MTGK algorithm, which identifies the most transferable networks from multiple sources using MMD, and learns domain-invariant and transferable node embeddings by incorporating both structural and label information of nodes.

Appendix A.3 Reinforcement Learning

Reinforcement learning (RL) [20, 21] is a machine learning paradigm that optimizes strategies through the iterative interaction between an agent and its environment. The agent executes a sequence of actions that influence the environment's state and, in return, receives rewards or penalties. The agent then adjusts its strategy to maximize cumulative long-term rewards based on this feedback.

Recent advancements have utilized RL to quantify data value, integrating the sample selection process with predictor model training to prioritize high-quality data from source domain datasets for subsequent learning tasks. For example, Yoon *et al.* [22] proposed a meta-learning framework, DVRL, where a data valuator estimates the utility of each data point for training a predictive model, thereby assessing the data's value within the training set. Swazinna *et al.* [23] proposed three metrics, i.e., ERI, EAS, and COI, for effective dataset selection. Abolfazli *et al.* [24] proposed DVORL, an offline RL-based data evaluation framework that identifies the most relevant source domain subset for a given target task, enabling the agent to learn strategies that generalize to the target domain.

The aforementioned data valuation methods are mainly applied to non-graph structured data, while research on node valuation methods for graph-structured data is still limited. Wu *et al.* [25] proposed the RSS-GNN model, which uses reinforcement learning to guide transfer learning. The model employs a Selection Distribution Generator (SDG) to generate probabilities for each graph and selects high-quality graphs to train the GNN, with a reward mechanism designed to assess the quality of the samples chosen by the SDG. Building on this, further research can focus on evaluating the data value of elements such as nodes and edges within graph-structured data, to select more valuable and higher-quality information to guide downstream tasks.

While these data valuation techniques are predominantly applied to structured data, research on node valuation within graph-structured data remains limited. Wu *et al.* [25] addressed this gap by introducing the RSS-GNN model, which employs reinforcement learning to guide transfer learning. The model uses a Selection Distribution Generator (SDG) to assign selection probabilities to graphs, focusing on high-quality graph selection for GNN training. The reward mechanism is designed to evaluate the quality of samples chosen by the SDG. Nevertheless, existing studies have yet to explore the evaluation of nodes within a source network in the context of GTL tasks, which could be instrumental in enhancing the classification performance of nodes in the target network.

Appendix B Problem Definition

Definition 1 (Node Quality-Driven Unsupervised Cross-Network Node Classification)**.** Given a fully labeled source network $\mathcal{G}_s = (V_s, E_s, X_s, A_s, Y_s)$, where V_s, E_s, X_s, A_s, and $Y_s \in \mathbb{R}^{n\times C}$ indicate the set of nodes, the set of edges, the attribute feature matrix, the topological structure feature matrix, and the label matrix associated with $\mathcal{G}_s$, respectively. $n = |V_s|$ is the number of nodes in $\mathcal{G}_s$, and C is the number of node classes. An unlabeled target network $\mathcal{G}_t = (V_t, E_t, X_t, A_t)$ with a set of nodes V_t, a set of edges E_t, the attribute feature matrix X_t, and the topological structure feature matrix A_t associated with $\mathcal{G}_t$, respectively. $m = |V_t|$ indicates the number of nodes in $\mathcal{G}_t$. The subscripts s and t indicate the source and target networks, respectively. The goal of NQDU is to select the highest quality nodes from the source network, and learn the most label-discriminative and network-invariant node vector representations, so as to effectively utilize the rich label information in the source network $\mathcal{G}_s$ to help classify unlabeled nodes in the target network $\mathcal{G}_t$.

Appendix C Theoretical Results

In this section, we propose the generalization error bound of the Non-IID network transfer learning for cross-network node classification.

Lemma C1 (Difference between Empirical and Theoretical Distances)**.** Let $\mathcal{H}^A_{\mathcal{D}_s}$ and $\mathcal{H}^A_{\mathcal{D}_t}$ be two hypothesis classes induced from a hypothesis space A, with VC dimensions d^A_s and d^A_t, respectively. Let G_s and G_t be two graphs with m_1 and m_2 unlabeled nodes drawn i.i.d. from $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively. Let $\hat{d}_{\mathcal{A}}(G_s, G_t)$ be the empirical $\mathcal{A}$-divergence between these graphs. Then, for any $\theta_1 \in (0, 1)$, with probability at least $1 - \theta_1$,

$$d_{\mathcal{A}}(\mathcal{D}_s, \mathcal{D}_t) \leqslant \hat{d}_{\mathcal{A}}(G_s, G_t) + 4\sqrt{\frac{\log[(2M)^{2d^A_s} + (2M)^{2d^A_t}] + \log\frac{1}{\theta_1}}{M}},$$

where $M = \min\{m_1, m_2\}$.

Proof. Based on Hoeffding's inequality, we have:

$$\begin{aligned}
&\mathrm{P}^{m_1+m_2}\left[|\varphi_{\mathcal{A}}(G_s,G_t)-\varphi_{\mathcal{A}}(\mathcal{D}_s,\mathcal{D}_t)|>\epsilon\right]\leqslant \mathrm{P}^{m_1+m_2}\left[|\varphi_{\mathcal{A}}(\mathcal{D}_s,G_s)-\varphi_{\mathcal{A}}(\mathcal{D}_t,G_t)|>\epsilon\right]\\
&\leqslant \mathrm{P}^{m_1+m_2}\left[2\max\{\varphi_{\mathcal{A}}(\mathcal{D}_s,G_s),\varphi_{\mathcal{A}}(\mathcal{D}_t,G_t)\}>\epsilon\right]=\mathrm{P}^{m_1+m_2}\left[\varphi_{\mathcal{A}}(\mathcal{D}^*,G^*)>\frac{\epsilon}{2}\right]\\
&\leqslant \mathrm{P}^{m_1}\left[\varphi_{\mathcal{A}}(\mathcal{D}_s,G_s)>\frac{\epsilon}{2}\right]+\mathrm{P}^{m_2}\left[\varphi_{\mathcal{A}}(\mathcal{D}_t,G_t)>\frac{\epsilon}{2}\right]\\
&\leqslant (2m_1)^{d_s^A}\exp\left\{\frac{-2(m_1\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m_1}(b_j-a_j)^2}\right\}+(2m_2)^{d_t^A}\exp\left\{\frac{-2(m_2\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m_2}(b_j-a_j)^2}\right\},
\end{aligned}$$

let $\exp\left\{\frac{-2(m\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m}(b_j-a_j)^2}\right\}=\max\left\{\exp\left\{\frac{-2(m_1\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m_1}(b_j-a_j)^2}\right\},\exp\left\{\frac{-2(m_2\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m_2}(b_j-a_j)^2}\right\}\right\}$, then, we have

$$\mathrm{P}^{m_1+m_2}\left[|\varphi_{\mathcal{A}}(G_s,G_t)-\varphi_{\mathcal{A}}(\mathcal{D}_s,\mathcal{D}_t)|>\epsilon\right]\leqslant\left[(2m_1)^{d_s^A}+(2m_2)^{d_t^A}\right]\exp\left\{\frac{-2(m\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m}(b_j-a_j)^2}\right\},$$

let $\left[(2m_1)^{d_s^A}+(2m_2)^{d_t^A}\right]\exp\left\{\frac{-2(m\epsilon-\sum_{j=1}^{k}(n_j-1)\delta d_j)^2}{\sum_{j=1}^{m}(b_j-a_j)^2}\right\}=\theta_1$, and suppose $m\epsilon\geqslant\sum_{j=1}^{k}(n_j-1)\delta d_j$, we have

$$d_{\mathcal{A}}(\mathcal{D}_s,\mathcal{D}_t)\leqslant\hat{d}_{\mathcal{A}}(G_s,G_t)+4\sqrt{\frac{\log[(2M)^{2d_s^A}+(2M)^{2d_t^A}]+\log\frac{1}{\theta_1}}{M}},$$

where $M=\min\{m_1,m_2\}$.

Lemma C2 (Classification Error Bounds in the Target Network)**.** Let $\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)$ denote the empirical $\mathcal{A}\triangle\mathcal{A}$ divergence. Then for any hypothesis, with probability at least $1-\theta_1$,

$$\epsilon_t(h)\leqslant\epsilon_s(h)+\frac{1}{2}\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)+4\sqrt{\frac{\log[(2M)^{2d_s^A}+(2M)^{2d_t^A}]+\log\frac{1}{\theta_1}}{M}}+\lambda,$$

where $M=\min\{m_1,m_2\}$.

Proof. For a given hypothesis space, the symmetric difference hypothesis space $\mathcal{A}\triangle\mathcal{A}$ can be detailed as:

$$g(x)\in\mathcal{A}\triangle\mathcal{A}\Leftrightarrow g(x)=(\mathcal{H}_s^A(x)\oplus\mathcal{H}_s^{A'}(x))\oplus(\mathcal{H}_t^A(x)\oplus\mathcal{H}_t^{A'}(x)),$$

where $A,A'\in\mathcal{A}$. Then, we have

$$\begin{aligned}
d_{\mathcal{A}\triangle\mathcal{A}}&=2\sup_{A,A'\in A}\left(\sup_{\substack{\mathcal{H}_s^A,\mathcal{H}_t^A\in A\\ \mathcal{H}_s^{A'},\mathcal{H}_t^{A'}\in A'}}\left(\sup_{\substack{h_s^A\in A,h_s^{A'}\in A'\\ h_t^A\in A,h_t^{A'}\in A'}}\left(\left|\Pr_{\mathcal{D}_s}\left[h_s^A(x)\neq h_s^{A'}(x)\right]-\Pr_{\mathcal{D}_t}\left[h_t^A(x)\neq h_t^{A'}(x)\right]\right|\right)\right)\right)\\
&=2\sup_{A,A'\in A}\left(\sup_{\substack{\mathcal{H}_s^A,\mathcal{H}_t^A\in A\\ \mathcal{H}_s^{A'},\mathcal{H}_t^{A'}\in A'}}\left(\sup_{\substack{h_s^A\in A,h_s^{A'}\in A'\\ h_t^A\in A,h_t^{A'}\in A'}}\left(\left|\epsilon_s(h_s^A,h_s^{A'})-\epsilon_s(h_t^A,h_t^{A'})\right|\right)\right)\right)\geqslant 2\left|\epsilon_s(h_s^A,h_s^{A'})-\epsilon_s(h_t^A,h_t^{A'})\right|,
\end{aligned}$$

then, we have

$$\left|\epsilon_s(h_s^A,h_s^{A'})-\epsilon_s(h_t^A,h_t^{A'})\right|\leqslant\frac{1}{2}d_{\mathcal{A}\triangle\mathcal{A}},$$

next, based on the triangle inequality, we have for any $\theta_1\in(0,1)$ and $h\in\mathcal{H}^A$, with probability at least $1-\theta_1$,

$$\epsilon_t(h_t^A,f_t)\leqslant\epsilon_s(h_s^A,f_s)+\frac{1}{2}\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)+4\sqrt{\frac{\log[(2M)^{2d_s^A}+(2M)^{2d_t^A}]+\log\frac{1}{\theta_1}}{M}}+\lambda,$$

where $M=\min\{m_1,m_2\}$.

Thus, Lemma C2 demonstrate that the source error and unlabeled $\mathcal{A}\triangle\mathcal{A}$-distance are important quantities for calculating the target error.

Theorem C1 (Generalization Bound for CNCC)**.** Let $\mathcal{H}_s^A$ and $\mathcal{H}_t^A$ denote hypothesis classes derived from a common hypothesis space A, corresponding to the source network G_s and the target network G_t, respectively. Let d_s^A and d_t^A be their representative VC-dimensions. Assume G_s and G_t consist of m_1 and m_2 unlabeled nodes sampled from distributions $\mathcal{D}_s$ and $\mathcal{D}_t$. A labeled set S of size m is constructed by sampling $\bar{\beta}m$ nodes from $\mathcal{D}_t$ and $(1-\bar{\beta})m$ nodes from $\mathcal{D}_s$, with labels given by f_s and f_t, respectively. Let $\hat{h}\in\mathcal{H}^A$ be the empirical minimizer of the convex combination of empirical

risks: $\hat{\epsilon}_{\bar{\alpha}}(h^A) = \bar{\alpha}\hat{\epsilon}_t(h_t^A) + (1-\bar{\alpha})\hat{\epsilon}_s(h_s^A)$ and let $h_t^* = \min_{h\in\mathcal{H}^A} \epsilon_t(h^A)$ be the optimal hypothesis minimizing the true risk on the target network. Then, with probability at least $1-\theta$, the following generalization bound holds:

$$\begin{aligned}\epsilon_t(\hat{h}_t^A) \leqslant \epsilon_t(h_t^*) &+ \frac{2\sum_{j=1}^{k_t}(n_j-1)\delta_t(\varphi_j^t+\tau_t) + 2\sum_{j=1}^{k_s}(n_j-1)\delta_s(\varphi_j^s+\tau_s)}{m} \\ &+ 4\sqrt{\frac{\bar{\alpha}^2}{\bar{\beta}} + \frac{(1-\bar{\alpha})^2}{1-\bar{\beta}}} \times \sqrt{\frac{2\log[(2(1-\bar{\beta})m)^{d_s^A} + (2\bar{\beta}m)^{d_t^A}] + 2\log\frac{8}{\theta}}{2m}} \\ &+ (1-\bar{\alpha})(\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)) + 4\sqrt{\frac{2d + \log[1+(2M)^{|2d_s^A - 2d_t^A|}] + \log\frac{1}{\theta_1}}{M}} + \lambda,\end{aligned}$$

where $\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)$ is the empirical $\mathcal{A}$-divergence between G_s and G_t, $M = \min\{\bar{\beta}m, (1-\bar{\beta})m\}$, $d = \min\{d_s^A, d_t^A\}$, λ is the joint optimal error of $\mathcal{H}^A$ on both domains, (δ, φ, τ) are correction terms arising from data selection and group imbalance.

Proof. Let $S = (S_t, S_s)$ be labeled nodes of size m generated by drawing $\bar{\beta}m$ nodes from $\mathcal{D}_t$ (i.e., S_t) and $(1-\bar{\beta}m)$ nodes from $\mathcal{D}_s$ (i.e., S_s). The goal of a learning machine is to find a hypothesis function with the least error, i.e., $\epsilon_t(h_t^A)$, in the target classification. Therefore, we choose to optimize the convex combination of empirical errors of the source and target networks, i.e., $\hat{\epsilon}_{\bar{\alpha}}(h^A) = \bar{\alpha}\hat{\epsilon}_t(h_t^A) + (1-\bar{\alpha})\hat{\epsilon}_s(h_s^A)$, where $\bar{\alpha} \in [0,1]$. Similarly, $\epsilon_{\bar{\alpha}}(h^A)$ is the true error of the source and target networks. Based on the triangle inequality, the relationship between the target error and the joint error can be detailed as:

$$|\epsilon_{\bar{\alpha}}(h^A) - \epsilon_t(h_t^A)| \leqslant (1-\bar{\alpha})(\frac{1}{2}d_{\mathcal{A}\triangle\mathcal{A}}(\mathcal{D}_s,\mathcal{D}_t) + \lambda)$$

Moreover, let $X_1, X_2, ..., X_{\bar{\beta}m} \in [0, \frac{\bar{\alpha}}{\bar{\beta}}]$ be $\bar{\beta}m$ node embeddings from $\mathcal{D}_t$, each value equals to $\frac{\bar{\alpha}}{\bar{\beta}}|h(x) - f_t(x)|$, let $X_{\bar{\beta}m+1}$, $X_{\bar{\beta}m+2}$, ..., $X_m \in [0, \frac{1-\bar{\alpha}}{1-\bar{\beta}}]$ be $(1-\bar{\beta})m$ node embeddings from $\mathcal{D}_s$, each value equals to $\frac{1-\bar{\alpha}}{1-\bar{\beta}}|h(x) - f_s(x)|$. Then, we have:

$$\begin{aligned}\bar{X} &= \frac{1}{m}\sum_{i=1}^{m} X_i = \frac{1}{m}\left(\frac{\bar{\alpha}}{\bar{\beta}}\sum_{x\in S_t}|h(x)-f_t(x)| + \frac{1-\bar{\alpha}}{1-\bar{\beta}}\sum_{x\in S_s}|h(x)-f_s(x)|\right) \\ &= \frac{\bar{\alpha}}{\bar{\beta}m}\sum_{x\in S_t}|h(x)-f_t(x)| + \frac{1-\bar{\alpha}}{(1-\bar{\beta})m}\sum_{x\in S_s}|h(x)-f_s(x)| = \bar{\alpha}\hat{\epsilon}_t(h_t^A) + (1-\bar{\alpha})\hat{\epsilon}_s(h_s^A) = \hat{\epsilon}_{\bar{\alpha}}(h^A),\end{aligned}$$

similarly,

$$\mathrm{E}(\bar{X}) = \mathrm{E}\left(\hat{\epsilon}_{\bar{\alpha}}(h^A)\right) = \frac{1}{m}\left(\bar{\beta}m\frac{\bar{\alpha}}{\bar{\beta}}\epsilon_t(h_t^A) + (1-\bar{\beta})m\frac{1-\bar{\alpha}}{1-\bar{\beta}}\epsilon_s(h_s^A)\right) = \bar{\alpha}\epsilon_t(h_t^A) + (1-\bar{\alpha})\epsilon_s(h_s^A) = \epsilon_{\bar{\alpha}}(h^A),$$

then, based on Hoeffding's inequality, we have:

$$\begin{aligned}\mathrm{P}\left(|\hat{\epsilon}_{\bar{\alpha}}(h^A) - \epsilon_{\bar{\alpha}}(h^A)| \geqslant \epsilon\right) &\leqslant \exp\left\{\frac{-2(m\epsilon - \sum_{j=1}^{k}(n_j-1)\delta\varphi_j)^2}{\sum_{j=1}^{m}(\mathrm{range}(X_j))^2}\right\} \\ &= 2\exp\left\{\frac{-2[m\epsilon - (\sum_{j=1}^{k_t}(n_j-1)\delta(\varphi_j^t+\tau_t) + \sum_{j=1}^{k_s}(n_j-1)\delta(\varphi_j^s+\tau_s))]^2}{\bar{\beta}m(\frac{\bar{\alpha}}{\bar{\beta}})^2 + (1-\bar{\beta})m(\frac{1-\bar{\alpha}}{1-\bar{\beta}})^2}\right\}.\end{aligned}$$

Thus, for any $\theta \in (0,1)$, with probability at least $1-\theta$,

$$\begin{aligned}\epsilon_t(\hat{h}_t) \leqslant \epsilon_t(h_t^*) &+ \frac{2\sum_{j=1}^{k_t}(n_j-1)\delta_t(\varphi_j^t+\tau_t) + 2\sum_{j=1}^{k_s}(n_j-1)\delta_s(\varphi_j^s+\tau_s)}{m} \\ &+ 4\sqrt{\frac{\bar{\alpha}^2}{\bar{\beta}} + \frac{(1-\bar{\alpha})^2}{1-\bar{\beta}}}\sqrt{\frac{2\log[(2(1-\bar{\beta})m)^{d_s^A} + (2\bar{\beta}m)^{d_t^A}] + 2\log\frac{8}{\theta}}{2m}} \\ &+ (1-\bar{\alpha})(\hat{d}_{\mathcal{A}\triangle\mathcal{A}}(G_s,G_t)) + 4\sqrt{\frac{2d + \log[1+(2M)^{|2d_s^A - 2d_t^A|}] + \log\frac{1}{\theta_1}}{M}} + \lambda,\end{aligned}$$

where $M = \min\{\bar{\beta}m, (1-\bar{\beta})m\}$ and $d = \min\{d_s^A, d_t^A\}$.

It is worth noting that the source-target weighting factor $\bar{\alpha}$ controls the relative importance of source and target domain data during training. For source networks with high-quality, well-aligned labels, a higher $\bar{\alpha}$ (e.g., $0.6-0.8$) is beneficial. Conversely, when there is considerable domain shift or label noise in the source network, a lower $\bar{\alpha}$ (e.g., $0.2-0.4$) helps emphasize the target distribution more effectively. As for the sampling ratio of target data for pseudo-labeling $\bar{\beta}$ affects how many pseudo-labeled target nodes are used to guide the learning process. When target domain data has clear cluster structures or reliable pseudo-label quality, a larger $\bar{\beta}$ can improve performance. Otherwise, a conservative $\bar{\beta}$ prevents overfitting to noisy pseudo-labels. Moreover, in our proposed NQDU framework, nodes with higher $\bar{\alpha}$-weighted impact on $\hat{\epsilon}_{\bar{\alpha}}$ are preferred, so the generalization bound implies that: (1) The selected nodes maximize transferability when their inclusion reduces the empirical divergence $\hat{d}_{\mathcal{A}\triangle\mathcal{A}}$ and aligns conditional distributions across domains. (2) The fewer the highly deviated node groups (e.g., lower δ, φ, τ), the tighter the bound.

Appendix D The Proposed NQDU Framework

We propose a novel framework called NQDU. This framework systematically exploits both attribute- and structure-level similarities between nodes in the source and target networks. Within NQDU, we propose an unsupervised CNNC method. The overall design of NQDU comprises two core components (as illustrated in Figure 1): (1) High-quality node selection from the source network. (2) Cross-network node classification, leveraging the selected nodes to learn network-invariant node representations. The two parts of the proposed framework are presented with detailed explanations in the following sections.

Appendix D.1 High-Quality Node Selection

To select the high-quality nodes from the source network for cross-network node embedding, we employ a node quality evaluator modeled by a deep neural network. This evaluator assesses the likelihood that nodes in the source network will be used during the training process of a cross-network node classification model. We utilize a reinforcement learning method with a sampling process to train the node quality evaluator, specifically leveraging the reward signals obtained from the target network. The detailed process of selecting the high-quality nodes is elaborated below.

Given a set of binary groups of nodes and labels in the source network, $G_s = \{(v_{s_i}, y_{s_i})\}_{i=1}^{n}$, a set of binary groups of nodes and labels in the target network, $G_t = \{(v_{t_j}, y_{t_j})\}_{j=1}^{m}$, and a set of binary groups of nodes and labels randomly sampled from the target network, $G_r = \{(v_{r_k}, y_{r_k})\}_{k=1}^{l}$, where v represents the node feature information in each network, referring to attribute feature, A, topological structure feature, X, or combined feature information of both attribute and topological structure features, $X \oplus A$, and y denotes the label information of each node. It is worth noting that the features in the source network obey the distribution P, while the features in the target network obey the distribution Q. These distributions do not need to be consistent between the source and target networks. The purpose of high-quality node selection is to identify nodes whose features with small distribution difference from the nodes in the target network. These selected nodes, deemed high-quality, are then used to assist in the node classification task in the target network.

As shown in Figure 1(a), a batch of binary groups, $\{(v_{s_i}, y_{s_i})\}_{i=1}^{m}$, is randomly selected from the source network, G_s, and fed into the node quality evaluator. The node quality evaluator outputs the selection probability of each binary groups, representing the estimated value of the node. These selection probabilities follow a multinomial distribution, $\omega_i = \ell_\nu(v_{s_i}, y_{s_i})$, for $i = 1, 2, ..., m$. Based on these selection probabilities, the sampler generates the selection vectors, $\Phi = (\phi_1, \phi_2, ..., \phi_m)$, where $\phi_i \in \{0, 1\}$, for $i = 1, 2, ..., m$. The classifier for node classification task employs nodes with $P(\phi_i = 1) = \omega_i$ for training. Simultaneously, nodes in the target network are fed into the classifier of node classification task for training the classifier model. To calculate the reinforcement signal, the sampled binary groups, $G_r = \{(v_{r_k}, y_{r_k})\}_{k=1}^{l}$, are fed into the classifier model for evaluation to obtain the loss. The loss is then compared with the previous loss to determine the value of the reward, after which the node quality evaluator can be updated.

Specifically, NQDU includes two learnable function, the node classification classifier model f_θ, and the node quality evaluator model ℓ_ν. The node classification classifier model f_θ optimizes different loss functions $\mathcal{L}_f$ based on the the specific classifier models employed. The function f_θ can be any trainable function with the parameter θ. The details of f_θ are as follows:

$$f_\theta = \arg\min \frac{1}{n} \sum_{i=1}^{n} \ell_\nu(v_{s_i}, y_{s_i}) \mathcal{L}_f. \tag{D1}$$

The node quality evaluator model ℓ_ν is designed to determine the node selection probability for the training classifier model, thereby generating the selection vector. The corresponding optimization problem is detailed as follows:

$$\begin{aligned} &\min_{\ell_\nu} \mathbb{E}_{(v_r, y_r)\sim Q} \left[\mathcal{L}_\ell(f_\theta(v_r), y_r)\right], \\ &\text{s.t. } f_\theta = \arg\min \mathbb{E}_{(v,y)\sim P} \left[\ell_\nu(v, y)\mathcal{L}_f\right], \end{aligned} \tag{D2}$$

where $\omega = \ell_\nu(v, y)$ represents the probability that the binary groups (v, y) from the source network is selected for training the classifier model. Let $\ell_\nu(G_s) = \{\ell_\nu(v_{s_i}, y_{s_i})\}_{i=1}^{n}$ be the probability distribution of all nodes in the source network. Let $\Phi \in \{0, 1\}^n$ be a binary vector indicating whether each node is selected ($\phi_i = 1$) or not ($\phi_i = 0$) for training the classifier model. The term $\pi_\nu(G_s, \Phi) = \prod_{i=1}^{n} [\ell_\nu(v_{s_i}, y_{s_i})^{\phi_i} (1 - \ell_\nu(v_{s_i}, y_{s_i}))^{1-\phi_i}]$ denotes the probability of choosing a specific selection vector Φ based on $\ell_\nu(G_s)$. We consider the output $\omega = \ell_\nu(v, y)$ from the node quality evaluator model as an estimate of the node quality and use this estimate to rank nodes in the source network. This ranking process identifies a subset of nodes used to train the classifier model and subsequently perform cross-network node embedding. In this study, reinforcement learning is employed to optimize the strategy gradient, with the reward value calculated using a set of nodes G_r randomly extracted from the target network. The detailed calculation process for the loss function $L(\nu)$ is as follows:

$$L(\nu) = \mathbb{E}_{(v_r, y_r)\sim Q} \left[\mathbb{E}_{\Phi\sim\pi_\nu(G_s,\cdot)} \left[\mathcal{L}_\ell(f_\theta(v_r), y_r)\right]\right] = \int Q(v_r) \left[\sum_{\Phi\in\{0,1\}^n} \pi_\nu(G_s, \Phi) \left[\mathcal{L}_\ell(f_\theta(v_r), y_r)\right]\right] \mathrm{d}v_r, \tag{D3}$$

we can then calculate the gradient of $L(\nu)$ as follows:

$$\begin{aligned}\nabla_\nu L(\nu) &= \int Q(v_r)\left[\sum_{\Phi\in\{0,1\}^n}\nabla_\nu\pi_\nu(G_s,\Phi)\left[\mathcal{L}_\ell(f_\theta(v_r),y_r)\right]\right]\mathrm{d}v_r \\ &= \int Q(v_r)\left[\sum_{\Phi\in\{0,1\}^n}\nabla_\nu\log(\pi_\nu(G_s,\Phi))\pi_\nu(G_s,\Phi)\left[\mathcal{L}_\ell(f_\theta(v_r),y_r)\right]\right]\mathrm{d}v_r \\ &= \mathbb{E}_{(v_r,y_r)\sim Q}\left[\mathbb{E}_{\Phi\sim\pi_\nu(G_s,\cdot)}\left[\mathcal{L}_\ell(f_\theta(v_r),y_r)\right]\nabla_\nu\log(\pi_\nu(G_s,\Phi))\right],\end{aligned} \tag{D4}$$

where the term $\nabla_\nu\log(\pi_\nu(G_s,\Phi))$ can be calculated as follows:

$$\begin{aligned}\nabla_\nu\log(\pi_\nu(G_s,\Phi)) &= \nabla_\nu\sum_{i=1}^{n}\log\left[\ell_\nu(v_{s_i},y_{s_i})^{\phi_i}(1-\ell_\nu(v_{s_i},y_{s_i}))^{1-\phi_i}\right] \\ &= \sum_{i=1}^{n}\phi_i\nabla_\nu\log\left[\ell_\nu(v_{s_i},y_{s_i})\right]+(1-\phi_i)\nabla_\nu\log\left[1-\ell_\nu(v_{s_i},y_{s_i})\right].\end{aligned} \tag{D5}$$

To improve the stability of training, we utilize the moving average of previous losses as the evaluation standard for the current loss.

Appendix D.2 Cross-Network Node Classification

Based on the feature we employed for the high-quality node selection, we propose an unsupervised CNNC approach (as shown in Figure 1(b)) to classify nodes in the target network which comprises a network embedding module, classifier model, and a network discriminator.

Appendix D.2.1 *Network Embedding*

As shown in Figure 1(b), the network embedding module comprises an attribute-level feature extractor, a topology-level feature extractor, a concatenation layer, and a topological consistency constraint.

Attribute-Level Feature Extractor. The attribute-level feature extractor processes the attribute information of each node through a fully connected deep neural network. This extractor, comprising l hidden layers, is defined as:

$$g_{al}^{(k)}(x_i)=\mathrm{ReLU}(g_{al}^{(k-1)}(x_i)W_{al}^{(k)}+b_{al}^{(k)}), \tag{D6}$$

where $1\leqslant k\leqslant l$. The parameters $W_{al}^{(k)}$ and $b_{al}^{(k)}$ comprise all weights and biases. Specifically, $g_{al}^{(0)}(x_i)=x_i\in\mathbb{R}^{1\times r}$ denotes the initial attribute vector of node v_i. The term $g_{al}^{(k)}(x_i)\in\mathbb{R}^{1\times e_k}$ represents the attribute-based node embeddings of v_i learned by the k-th hidden layer of the attribute-level feature extractor. Here, x_{ik} is the k-th attribute value of v_i; if $x_{ik}=0$, it means that v_i does not possess the k-th attribute.

Topology-Level Feature Extractor. For the topology-level feature extractor, the attributes of neighboring nodes are input into another fully connected deep neural network. The topology-level feature extractor, with l hidden layers, is described as follows:

$$g_{tl}^{(k)}(x_i^n)=\mathrm{ReLU}(g_{tl}^{(k-1)}(x_i^n)W_{tl}^{(k)}+b_{tl}^{(k)}), \tag{D7}$$

where $1\leqslant k\leqslant l$. The parameters $W_{tl}^{(k)}$ and $b_{tl}^{(k)}$ comprise all weights and biases. Specifically, $g_{tl}^{(0)}(x_i^n)=x_i^n\in\mathbb{R}^{1\times r}$ is the initial neighbor attribute vector of node v_i. The term $g_{tl}^{(k)}(x_i^n)\in\mathbb{R}^{1\times e_k}$ indicates the attribute representations of node v_i's neighbors learned by the k-th hidden layer. To calculate x_i^n, we first determine the k-step transition probability matrices, $\{T^{(k)}\}_{k=1}^K$, based on the topological proximity matrix A, where K indicates the number of steps from one node to another within the network. We then aggregate the transition probability matrices by assigning higher weights to closer neighbors, resulting in the aggregated transition probability matrix $T=\sum_{k=1}^K T^{(k)}/k$. The Positive Pointwise Mutual Information (PPMI) matrix is then computed based on T. Utilizing the PPMI matrix and node attributes, we finally derive the neighbors 'attributes:

$$x_{ik}^n=\sum_{j=1,j\neq i}^{p}\frac{a_{ij}}{\sum_{g=1,g\neq i}^{p}a_{ig}}x_{jk}, \tag{D8}$$

where a_{ij} is the element in the i-th row and j-th column of A. A higher positive value of a_{ij} indicates a stronger network affinity between nodes v_i and v_j. Conversely, if $a_{ij}=0$, it means that nodes v_i and v_j are not neighbors within k steps in $\mathcal{G}$. The number of hidden layers l and the dimensionality of each k-th hidden layer e_k are set to the same for both extractors.

Concatenation Layer. To obtain concatenated node representations $e_i\in\mathbb{R}^{1\times d}$, we combine the representations $g_{al}^{(l)}(x_i)$ and $g_{tl}^{(l)}(x_i^n)$ learned by the attribute-level feature and the topology-level feature extractors, respectively, through a concatenation layer:

$$e_i=\mathrm{ReLU}((g_{al}^{(l)}(x_i)\oplus g_{tl}^{(l)}(x_i^n))W_c+b_c), \tag{D9}$$

where $g_{al}^{(l)}(x_i)\oplus g_{tl}^{(l)}(x_i^n)$ denotes the concatenation of $g_{al}^{(l)}(x_i)$ and $g_{tl}^{(l)}(x_i^n)$. W_c and b_c are the differentiable parameters of the concatenation layer. This integration ensures that the attribute relationships and topological similarities between nodes are adequately preserved.

Topological Consistency Constraint. To maintain topological consistency between nodes in the source and target networks, we impose the following constraint on the obtained node representations:

$$\mathcal{L}_{tcc} = \frac{1}{n_s} \sum_{v_i, v_j \in V_s} \phi_i a_{ij} \left\| e_i - e_j \right\|^2 + \frac{1}{n_t} \sum_{v_i, v_j \in V_t} a_{ij} \left\| e_i - e_j \right\|^2, \tag{D10}$$

where n_s and n_t denote the total number of nodes in $\mathcal{G}_s$ and $\mathcal{G}_t$, respectively. By minimizing $\mathcal{L}_{tcc}$, closely connected nodes in the source and target networks will have similar node vector representations. ϕ_i represents the node quality selection probability in the source network, as determined by the node quality evaluator in the proposed NQDU framework. The differentiable parameters of the network embedding can be represented as $\theta_e = \{\{W_{al}^{(k)}, b_{al}^{(k)}\}_{k=1}^{l}, \{W_{tl}^{(k)}, b_{tl}^{(k)}\}_{k=1}^{l}, W_c, b_c\}$.

Appendix D.2.2 *Classifier Model*

For various node classification tasks, such as multi-class or multi-label classification, we employ different functions to predict node labels. Generally, a linear layer maps the output of the deep neural network embedding layer to a single value:

$$\hat{y}_i = f_\theta(e_i W_y + b_y), \tag{D11}$$

where $\hat{y}_i \in \mathbb{R}^{1 \times C}$ represents the output probabilities of node v_i over the C label classes, and $f_\theta(\cdot)$ is the classifier's output function. Different classification functions, such as Softmax or Sigmoid, can be employed for different tasks. The parameters $\theta_y = \{W_y, b_y\}$ include all the weights and biases. After obtaining $\hat{y}_i$, the following loss function is applied for multi-class node classification:

$$\mathcal{L}_y = -\frac{1}{n_s} \sum_{v_i \in V_s} \sum_{k=1}^{C} \phi_i y_{ik} \log(\hat{y}_{ik}), \tag{D12}$$

where y_{ik} is the true label of node v_i, if node v_i is associated with label k, then $y_{ik} = 1$; otherwise, $y_{ik} = 0$. $\hat{y}_{ik}$ denotes the output probability of node v_i being classified under class k. Similarly, the Sigmoid cross-entropy loss is formulated as:

$$\mathcal{L}_y = -\frac{1}{n_s} \sum_{v_i \in V_s} \sum_{k=1}^{C} \phi_i \left[y_{ik} \log(\hat{y}_{ik}) + (1 - y_{ik}) \log(1 - \hat{y}_{ik}) \right], \tag{D13}$$

where ϕ_i in Eq. (D12) and Eq. (D13) represents the node quality selection probabilities in the source network, as determined by the node quality evaluator in the proposed NQDU framework.

Appendix D.2.3 *Network Discriminator*

The network discriminator aims to align the probability distributions between the source and target networks. First, the node vector representations learned from the network embedding module are fed into a network discriminator, which identifies whether the node belongs to the source or target networks:

$$\hat{d}_i = \text{Softmax}(g^{(l_{nd})}(e_i) W^{(l_{nd}+1)} + b^{(l_{nd}+1)}), \tag{D14}$$

where

$$g^{(k)}(e_i) = \text{ReLU}(g^{(k-1)}(e_i) W^{(k)} + b^{(k)}), \tag{D15}$$

and $1 \leqslant k \leqslant l_{nd}$, $g^{(0)}(e_i) = e_i$. The term $g^{(k)}(e_i) \in \mathbb{R}^{1 \times d_k}$ represents the network representations of node v_i learned by the k-th hidden layer in the network discriminator. The parameters l_{nd} and d_k denote the number of hidden layers and the dimensionality of the k-th hidden layer, respectively. The parameters $\theta_{nd} = \{W^{(k)}, b^{(k)}\}_{k=1}^{l_{nd}+1}$ include all the weights and biases. The loss for the network discriminator is computed as:

$$\mathcal{L}_{nd} = -\frac{1}{n_s + n_t} \sum_{v_i \in V_s, v_j \in V_t} \phi_i \left[(1 - d_i) \log(1 - \hat{d}_i) + d_i \log(\hat{d}_i) \right] + (1 - d_i) \log(1 - \hat{d}_i) + d_i \log(\hat{d}_i), \tag{D16}$$

where d_i indicates the network label of node v_i. If $v_i \in V_t$, then $d_i = 1$; if $v_i \in V_s$, then $d_i = 0$. $\hat{d}_i$ indicates the predicted probability of node v_i belonging to the target network. ϕ_i represents the node quality selection probability in the source network, as determined by the node quality evaluator in the proposed NQDU framework.

To ensure that the node representations learned from the deep neural network are network-invariant, the network embedding module and network discriminator engage in an adversarial process. Specifically, the objective function $\min_{\theta_{nd}} \{\gamma \mathcal{L}_{nd}\}$ enables the network discriminator to accurately distinguish between the node representations of the source network and the target network. Conversely, the objective function $\min_{\theta_e} \{-\gamma \mathcal{L}_{nd}\}$ is used to train the network embedding module. The goal is to deceive the network discriminator by generating node representations that are indistinguishable across networks.

Appendix D.2.4 *Overall Loss and Model Training*

Finally, the optimization objective of the NQDU can be obtained by integrating the network embedding module, classifier model, and network discriminator as follows:

$$\min_{\theta_e,\theta_y} \{\mathcal{L}_y + \eta\mathcal{L}_{tcc} - \lambda \max_{\theta_{nd}} \{\mathcal{L}_{nd}\}\}, \tag{D17}$$

where η and λ are trade-off parameters. To update the parameters simultaneously during the backpropagation procedure, we incorporate a gradient reversal layer between the network embedding module and the network discriminator. Thus, the NQDU can be trained using stochastic gradient descent (SGD) as:

$$\begin{aligned} \theta_e &\leftarrow \theta_e - \gamma(\frac{\partial \mathcal{L}_y}{\partial \theta_e} + \eta\frac{\partial \mathcal{L}_{tcc}}{\partial \theta_e} - \lambda\frac{\partial \mathcal{L}_{nd}}{\partial \theta_e}), \\ \theta_y &\leftarrow \theta_y - \gamma\frac{\partial \mathcal{L}_y}{\partial \theta_y}, \\ \theta_{nd} &\leftarrow \theta_{nd} - \gamma\frac{\partial \mathcal{L}_{nd}}{\partial \theta_{nd}}, \end{aligned} \tag{D18}$$

where γ is the learning rate.

Table D1 The detailed description of datasets used in this study.

Datasets		#Nodes	#Edges	#Attributes	#Labels
Social Networks	Blog1	2,300	33,471	8,189	6
	Blog2	2,896	53,836		
Citation Networks	Citationv1	3,935	15,113	6,775	5
	DBLPv7	3,484	8,130		
	ACMv9	4,360	15,602		
Protein-Protein Interaction Networks	PPI-1	1,689	34,085	50	121
	PPI-2	1,333	31,081		
	PPI-3	2,190	61,907		
	PPI-4	2,260	67,769		
	PPI-5	1,533	37,740		

Table D2 Parameter setting for the three networks in our study.

Dataset	β	B_s	μ	B_C	d
Social Networks	0.001	500	0.02	100	128
Citation Networks	0.001	500	0.02	100	128
PPI Networks	0.0001	500	0.01	200	512

Appendix E Experiments and Analysis

We conducted extensive experiments on three kinds of real-world datasets (as shown in TABLE D1) including social networks, citation networks, and protein-protein interaction networks (i.e., PPI) to answer the following six key questions:

- **Q1:** How do our models perform compared to the state-of-the-art node classification methods?
- **Q2:** What criteria do we employ to select features for different datasets in implementing our proposed node quality-driven unsupervised cross-network node classification?
- **Q3:** What is the impact of the high-quality node selection module on the performance of our proposed framework?
- **Q4:** How does our proposed NQDU cross-network node classification framework impact existing CNNC methods?
- **Q5:** How do the learning rate β, the domain adaptation weight λ, the batch sizes (i.e., B_s and B_c), and the dimensionality of node representations d, affect the performance of our models?
- **Q6:** How does the model convergence of the NQDU method?

Appendix E.1 Experimental Settings

Appendix E.1.1 *Datasets*

We conducted experiments on the following three real-world datasets, i.e., social networks [26], citation networks [15], and protein-protein interaction networks [9] (i.e., PPI). For each dataset, we constructed different source and target networks, totaling 28 transfer learning tasks, where A→B denotes the transfer of knowledge from source A to target B. The detailed description of datasets can be found bellow.

Social Networks. We utilized two distinct subnetworks, i.e., Blog1 and Blog2, sourced from the BlogCatalog dataset [26]. Each node within these networks represents a blogger, while each edge denotes a follower relationship between bloggers.

The attribute information for each node comprises keywords extracted from the blogger's self-description. Additionally, each node is assigned a single label that represents the blogger's interest group. Consequently, we established two transfer tasks: Blog1→Blog2 and Blog2→Blog1, where A→B denotes the transfer of knowledge from source A to target B.

Citation Networks. We employed three distinct citation networks, Citationv1 (C), DBLPv7 (D), and ACMv9 (A), extracted from the ArnetMiner dataset [15]. In these networks, nodes correspond to academic papers, and edges represent citation links. These citation networks are modeled as undirected graphs. Each node's attribute information consists of sparse bag-of-words features derived from the paper titles. Nodes may possess multiple labels to indicate their relevant research domains. The varying sources and formation periods of Citationv1, DBLPv7, and ACMv9 result in inherently different data distributions across these networks. Consequently, we considered six transfer tasks, including C→A, C→D,..., and D→C, among others.

Protein-Protein Interaction Networks. We utilized five distinct protein-protein interaction networks, namely PPI-1, PPI-2, PPI-3, PPI-4, and PPI-5, derived from the comprehensive protein-protein interaction network [9]. In these networks, nodes represent individual proteins, and edges denote the interactions and influences between them. Each node is described by 50 features, including motif sets and immunological characteristics, and can have up to 121 labels providing detailed information about the protein's attributes and locations. Consequently, we formulated 20 transfer tasks, including PPI-1→PPI-2, PPI-1→PPI-3,..., and PPI-5→PPI-4, to evaluate our methods.

Table E1 Cross-network node classification accuracy (%) on social networks. The best and best baseline results are shown in boldface and underline, respectively.

G_s	G_t	F1	GCN	GAT	GraphSAGE	UDAGCN	ACDNE	ASN	GRADE	SpecReg	A2GNN	TFGDA	HOGDA	NQDU_{A}	NQDU_{X}	$\text{NQDU}_{\text{X}\oplus\text{A}}$	$\text{NQDU}_{\text{X}\oplus\text{P(A)}}$
Blog1	Blog2	Macro	43.43	49.02	57.88	47.91	64.69	61.14	57.95	62.09	<u>64.73</u>	62.05	62.47	87.23	**87.91**	87.89	87.44
		Micro	50.79	53.07	58.67	56.28	65.09	62.29	59.03	63.27	<u>65.69</u>	63.22	64.17	87.43	88.09	88.05	**88.64**
Blog2	Blog1	Macro	43.68	44.61	60.02	45.26	<u>62.73</u>	59.86	54.58	60.40	61.89	60.90	60.99	90.27	90.40	90.93	**90.57**
		Micro	47.04	52.60	60.83	55.91	<u>62.78</u>	60.98	56.31	61.54	62.65	61.84	62.01	90.43	90.52	**91.09**	90.70
Average		Macro	43.56	46.82	58.95	46.59	<u>63.71</u>	60.50	56.27	61.25	63.31	61.48	61.73	88.75	89.16	**89.41**	89.06
		Micro	48.92	52.84	59.75	56.10	63.94	61.64	57.67	62.41	<u>64.17</u>	62.53	63.09	88.93	89.31	89.57	**89.67**
Improvement		Macro	45.85	42.59	30.46	42.82	25.70	28.91	33.14	28.16	26.10	27.93	27.68	0.66	0.25	—	0.35
		Micro	40.75	36.83	29.92	33.57	25.73	28.03	32.00	27.26	25.50	27.14	26.58	0.74	0.36	0.10	—

Appendix E.1.2 *Experimental Environment and Parameter Setting*

All experiments were conducted on an Ubuntu-based platform equipped with Intel E5-series CPUs and NVIDIA RTX 2080TI GPUs. Development was performed using PyCharm, with Python 3.8 and PyTorch 1.7.1 as the software environment. The moving average window for the loss was set to $T = 20$. Node classification performance was evaluated using Micro-F1 and Macro-F1 scores, which are widely adopted metrics for assessing multi-class and class-imbalanced classification tasks [28–30].

The detailed parameter configurations for each dataset are summarized in Table D2. Specifically, β denotes the learning rate for the node quality evaluator, B_s and B_c represent the batch sizes for the evaluator and the classifier, respectively, μ refers to the initial learning rate of the classifier, and d indicates the dimensionality of the learned node embeddings. The classifier learning rate μ is scheduled to decay following the formulation $\mu_\eta = \frac{\mu_0}{(1+10\eta)^{0.75}}$, where μ_0 is the initial learning rate and $\eta \in [0, 1]$ is the weight of the topological consistency constraint. This constraint governs the extent to which structural information is preserved during training: higher values of η (closer to 1) are recommended for densely connected graphs to strengthen structural coherence, whereas moderate values are preferable for sparse networks to avoid amplifying noise. The domain adaptation weight λ, which regulates the alignment between source and target domains during adversarial training, is adjusted using the formula $\lambda = \frac{2}{1+\exp(-10\eta)}$. A larger λ (e.g., 1) is beneficial when the source and target networks exhibit substantial distributional shifts, whereas a smaller λ (e.g., 0.1-0.3) helps prevent over-regularization when networks are structurally similar. To mitigate overfitting, we employed an L_2 regularization coefficient of 0.001. The choice of batch sizes B_s and B_c has a direct impact on training stability and convergence. Larger batches help stabilize training but incur higher computational costs, while smaller batches may improve generalization but at the expense of longer training durations. Based on empirical evaluation, we recommend $B_s \in [64, 512]$ and $B_c \in [32, 128]$.

Appendix E.1.3 *Comparison Methods*

We evaluated the performance of our NQDU models against six state-of-the-art baseline models, categorized into two groups: (1) non-transfer network node classification baselines, including GCN [7], GraphSAGE [9], and GAT [8]. (2) cross-network node classification baselines, including AdaGCN [16], UDAGCN [18], ACDNE [17], ASN [27], GRADE [11], SpecReg [28], A2GNN [29], TFGDA [30], and HOGDA [31]. These baseline models were selected due to their relevance and prominence in the field. For each baseline, experiments were conducted using each source network, the best results from the target network were selected, and the average performance across five iterations was reported for each model. Within the NQDU framework, we designed four CNNC models based on different node feature selection strategies, i.e., NQDU_{A}, NQDU_{X}, $\text{NQDU}_{\text{X}\oplus\text{A}}$, and $\text{NQDU}_{\text{X}\oplus\text{P(A)}}$. Specifically, NQDU_{A} uses only attribute features, NQDU_{X} employs topological structure features, $\text{NQDU}_{\text{X}\oplus\text{A}}$ combines attribute and topology features, and $\text{NQDU}_{\text{X}\oplus\text{P(A)}}$ extends this combination by applying PPMI to enhance the semantic expressiveness of attribute features.

Appendix E.2 Performance Comparison and Analysis

To answer the six key questions **Q1-Q6**, we conducted the following experiments and analyzed the corresponding results.

Table E2 Cross-network node classification accuracy (%) on citation networks. The best and best baseline results are shown in boldface and underline, respectively.

G_s	G_t	F1	GCN	GAT	GraphSAGE	UDAGCN	ACDNE	ASN	GRADE	SpecReg	A2GNN	TFGDA	HOGDA	NQDU$_{\mathrm{A}}$	NQDU$_{\mathrm{X}}$	NQDU$_{\mathrm{X\oplus A}}$	NQDU$_{\mathrm{X\oplus P(A)}}$
C	D	Macro	54.18	48.79	60.62	41.57	76.02	73.98	70.02	73.64	77.04	72.60	72.90	85.92	**86.70**	**86.70**	86.29
		Micro	66.00	53.58	63.58	55.09	77.54	76.36	73.95	75.74	78.13	76.00	77.10	85.66	86.38	**86.43**	86.00
D	C	Macro	52.65	43.44	49.97	34.34	80.04	75.17	69.32	77.78	79.74	74.40	75.00	95.99	96.95	96.97	**97.05**
		Micro	62.78	48.92	52.05	47.22	81.89	78.23	74.32	79.04	81.54	78.90	80.30	96.08	97.06	97.06	**97.11**
C	A	Macro	51.22	44.27	49.76	34.52	78.51	73.17	69.34	73.15	77.57	72.30	72.60	97.15	**97.94**	97.93	97.92
		Micro	61.64	49.32	53.71	49.60	79.26	72.14	69.55	72.04	76.15	73.60	74.40	97.07	**97.84**	97.82	97.76
A	C	Macro	54.59	46.94	46.19	41.48	80.86	77.81	72.52	78.83	81.30	78.90	79.20	95.73	95.72	**95.74**	95.70
		Micro	64.87	50.59	46.91	52.90	82.73	80.64	76.04	80.55	82.64	81.00	82.40	**95.54**	95.49	**95.54**	95.46
D	A	Macro	46.80	42.01	47.70	26.88	75.95	71.49	59.35	72.34	75.69	64.30	65.10	96.73	97.23	**97.35**	97.14
		Micro	57.00	47.68	50.51	42.57	76.38	70.15	63.72	71.01	74.12	66.90	68.20	96.46	96.83	**96.91**	96.76
A	D	Macro	53.25	43.93	64.47	43.02	75.27	71.40	63.03	73.98	75.78	73.20	73.80	**79.24**	78.68	78.84	78.18
		Micro	65.11	49.97	67.02	52.12	76.63	73.80	68.22	75.93	77.43	75.30	76.50	**78.16**	77.33	77.59	76.93
Average		Macro	52.12	44.90	53.12	36.97	77.78	73.84	67.26	74.95	77.85	72.62	73.10	91.79	92.20	**92.26**	92.05
		Micro	62.90	50.01	55.63	49.92	79.07	75.22	70.97	75.72	78.34	75.28	76.48	91.50	91.82	**91.89**	91.67
Improvement		Macro	40.14	47.36	39.14	55.29	14.18	18.42	25.00	17.31	14.41	19.64	19.16	0.47	0.06	—	0.21
		Micro	28.99	41.88	36.26	41.97	12.82	16.67	20.92	16.17	13.55	16.61	15.41	0.39	0.07	—	0.22

Table E3 Cross-network node classification accuracy (%) on PPI networks. The best and best baseline results are shown in boldface and underline, respectively.

G_s	G_t	F1	GCN	GraphSAGE	GAT	AdaGCN	UDAGCN	ACDNE	ASN	GRADE	SpecReg	A2GNN	TFGDA	HOGDA	NQDU$_{\mathrm{A}}$	NQDU$_{\mathrm{X}}$	NQDU$_{\mathrm{X\oplus A}}$	NQDU$_{\mathrm{X\oplus P(A)}}$
PPI-2	PPI-1	Macro	44.37	43.75	33.63	32.12	46.74	51.46	46.92	44.05	46.95	51.40	45.21	47.09	64.79	64.68	65.42	**65.68**
		Micro	63.31	63.05	41.45	53.45	63.36	64.52	60.07	57.62	60.14	64.55	58.96	59.30	70.82	70.71	71.40	**71.50**
PPI-3	PPI-1	Macro	43.94	45.15	33.23	27.46	46.50	52.15	58.89	55.68	58.90	52.17	54.22	54.58	66.14	66.73	**67.02**	66.70
		Micro	63.06	62.73	38.61	51.93	63.23	64.80	61.76	59.01	62.02	64.85	60.03	61.20	72.06	72.54	**72.83**	72.78
PPI-4	PPI-1	Macro	43.57	44.47	20.48	29.72	45.90	52.33	47.73	44.96	47.68	52.29	45.34	46.61	**64.75**	64.24	64.61	64.26
		Micro	62.74	62.48	41.17	51.90	63.02	64.61	60.02	57.83	60.21	64.58	63.00	63.79	**70.66**	70.18	70.34	70.37
PPI-5	PPI-1	Macro	43.78	42.94	35.21	30.94	46.06	51.36	47.38	45.18	47.51	51.40	48.89	49.97	**66.74**	66.24	66.58	66.36
		Micro	62.75	61.58	41.46	53.48	63.34	64.79	60.28	47.01	60.16	64.80	62.28	64.36	**72.48**	72.02	72.19	72.21
PPI-1	PPI-2	Macro	45.91	45.75	34.99	28.81	46.84	52.45	48.09	45.66	48.30	52.39	48.81	48.90	64.42	64.62	**64.83**	64.51
		Micro	64.94	64.89	40.41	55.86	64.98	67.01	63.55	60.32	63.09	66.98	64.27	65.09	71.19	70.93	71.18	**71.22**
PPI-3	PPI-2	Macro	46.81	47.52	25.92	29.67	49.91	56.02	51.18	47.97	52.23	56.13	52.47	53.55	70.70	71.31	**71.56**	70.31
		Micro	66.28	66.06	40.99	55.48	66.99	68.68	64.57	60.99	65.08	68.71	64.96	65.83	75.95	76.42	**76.87**	75.87
PPI-4	PPI-2	Macro	46.23	46.90	33.15	32.96	49.60	54.30	50.24	47.25	49.97	54.26	49.58	50.11	64.42	64.78	**65.00**	64.70
		Micro	66.04	65.48	41.01	55.99	65.78	67.80	63.01	61.00	62.79	67.77	63.31	63.77	70.60	70.89	**71.19**	70.92
PPI-5	PPI-2	Macro	46.08	44.29	36.96	31.79	48.71	51.68	47.78	44.36	48.06	51.74	47.09	47.64	**62.64**	62.06	62.33	62.52
		Micro	65.28	64.70	45.24	56.16	65.95	65.37	62.53	59.78	63.14	65.40	63.58	64.16	**69.12**	68.74	68.93	68.97
PPI-1	PPI-3	Macro	42.61	43.53	32.74	26.90	43.90	50.45	47.00	43.53	46.49	50.22	44.85	45.07	65.02	64.89	**65.05**	64.71
		Micro	62.47	62.60	41.29	53.04	62.85	64.21	60.03	56.84	59.91	63.96	57.25	58.99	71.11	71.14	**71.26**	71.14
PPI-2	PPI-3	Macro	44.12	43.33	33.80	30.25	50.51	53.36	49.76	46.83	50.24	53.19	48.05	49.24	69.55	70.31	69.48	**69.89**
		Micro	63.85	63.42	41.89	54.35	65.31	66.09	63.07	60.42	64.15	65.97	63.20	63.86	74.42	75.04	74.42	**74.74**
PPI-4	PPI-3	Macro	43.59	44.29	32.30	30.02	53.27	55.70	51.21	47.93	51.20	55.46	50.98	52.04	**78.95**	78.80	78.67	78.23
		Micro	63.67	63.33	40.83	52.56	66.38	67.47	63.22	60.05	63.19	67.51	60.77	62.18	**82.60**	82.51	82.35	82.08
PPI-5	PPI-3	Macro	43.45	41.67	25.64	28.91	51.72	53.29	49.89	46.43	50.27	53.34	47.68	49.00	70.77	71.36	**71.93**	71.01
		Micro	63.26	62.61	39.53	53.42	65.49	65.95	61.04	57.22	61.45	66.04	58.96	59.73	75.82	76.29	**76.65**	75.84
PPI-1	PPI-4	Macro	42.20	42.64	25.11	26.70	43.77	49.30	46.17	43.86	46.83	50.08	43.66	45.05	62.56	62.34	62.52	**62.70**
		Micro	62.18	62.08	37.38	52.57	62.49	63.66	59.96	56.32	60.31	63.74	57.11	57.68	69.06	68.86	68.94	**69.20**
PPI-2	PPI-4	Macro	43.29	42.81	26.70	32.60	46.76	52.39	48.78	46.06	48.66	52.30	47.08	48.67	**63.77**	63.44	62.74	62.83
		Micro	63.20	63.05	37.32	54.59	64.36	64.28	60.32	57.34	60.30	64.12	58.90	60.22	**69.69**	69.42	69.00	69.00
PPI-3	PPI-4	Macro	42.87	44.01	31.46	28.20	52.39	54.98	50.41	47.65	51.05	55.04	49.15	49.63	**78.44**	77.74	78.07	77.35
		Micro	63.54	63.20	39.92	52.17	65.95	67.04	64.43	60.19	65.33	67.11	62.03	63.27	**82.00**	81.48	81.75	81.30
PPI-5	PPI-4	Macro	42.24	41.09	26.24	29.11	47.99	51.77	47.79	44.32	47.74	51.60	48.52	49.06	68.82	68.51	69.14	**69.17**
		Micro	62.68	62.23	39.29	53.23	63.88	64.94	61.41	57.03	62.09	64.86	63.34	64.71	73.97	73.78	74.11	**74.30**
PPI-1	PPI-5	Macro	44.04	44.68	31.03	26.08	46.56	50.76	45.50	42.16	45.53	50.56	43.02	43.77	65.78	66.73	66.47	**66.75**
		Micro	63.38	63.55	40.67	52.86	63.92	64.82	60.12	58.73	61.05	64.58	58.00	59.46	71.92	72.52	72.36	**72.55**
PPI-2	PPI-5	Macro	44.80	44.05	30.55	32.42	49.48	51.35	48.01	45.25	48.25	51.50	44.11	44.56	**61.62**	61.14	61.14	61.22
		Micro	63.74	63.54	39.61	55.13	64.64	64.46	60.30	57.77	60.54	64.31	56.04	56.49	**68.21**	67.65	67.64	67.74
PPI-3	PPI-5	Macro	44.20	45.41	24.68	30.41	48.12	54.80	51.28	47.05	50.79	53.98	50.25	50.58	73.47	72.62	**73.78**	72.90
		Micro	63.84	63.65	41.06	53.37	64.81	66.85	63.19	61.22	62.97	66.00	62.23	63.06	78.02	77.27	**78.30**	77.69
PPI-4	PPI-5	Macro	44.10	44.73	21.10	28.68	48.20	54.30	52.79	48.89	53.25	54.35	51.08	51.95	**71.25**	69.76	70.36	70.51
		Micro	63.81	63.17	40.83	52.14	64.69	65.89	63.07	61.32	64.63	65.98	63.76	63.88	**75.98**	74.72	75.43	75.45
Average		Macro	44.11	44.15	29.75	29.69	48.15	52.71	49.34	46.25	49.55	52.67	48.00	48.85	67.73	67.62	**67.84**	67.62
		Micro	63.70	63.37	40.50	53.68	64.57	65.80	61.80	58.40	62.13	65.59	61.10	62.05	73.28	73.16	**73.36**	73.24
Improvement		Macro	23.73	23.69	38.09	38.15	19.69	15.13	18.50	21.59	18.29	15.17	19.84	18.99	0.11	0.22	—	0.22
		Micro	9.66	9.99	32.86	19.68	8.79	7.56	11.56	14.96	11.23	7.77	12.26	11.31	0.08	0.20	—	0.12

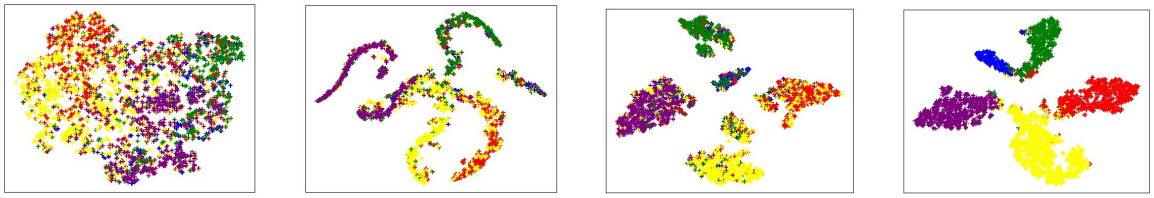
Result 1 Performance Comparison (for Q1): To answer **Q1**, we evaluated the performance of the proposed NQDU methods against six baseline models. For the non-transfer node classification models, we trained them on the source network and reported the best classification results on the target network. For the cross-network node classification models, the same procedure was followed. The classification performances of the baseline models and our NQDU methods are presented in TABLE E1, TABLE E2, and TABLE E3. In these tables, the best results are highlighted in bold, while the best baseline results are underlined. Importantly, our proposed methods (i.e., NQDU_{A}, NQDU_{X}, $\text{NQDU}_{\text{X}\oplus\text{A}}$, and $\text{NQDU}_{\text{X}\oplus\text{P(A)}}$) consistently outperform the baseline methods in classification tasks, regardless of the features used for high-quality node selection.

As illustrated in TABLE E1, our proposed methods outperform existing approaches in cross-network node classification tasks on social networks. Specifically, the highest Micro-F1 score achieved by our method outperforms the best baseline method, A2GNN, by an average of 25.50%. Additionally, the highest Macro-F1 score exceeds ACDNE's by an average of 25.70%.

Similarly, TABLE E2 demonstrates that our proposed method achieves Micro-F1 score improvements ranging from 12.82% to 41.97% over the baseline methods. The Macro-F1 scores show even more substantial improvements, ranging from 14.18% to 55.29%.

Furthermore, as shown in TABLE E3, the best Micro-F1 score of our proposed method on PPI networks indicates an average improvement of 7.56% over ACDNE. The best Macro-F1 score shows an average increase of 15.13% compared to ACDNE. These results underscore the effectiveness and robustness of our methods across different types of networks.

To further demonstrate the effectiveness of our proposed method, we utilized t-SNE to visualize the learned node feature representations. This visualization provides an intuitive assessment of the classification performance. Taking the Citationv1→DBLPv7 transfer task as an example, Figure E1 shows the two-dimensional t-SNE visualizations of node feature representations in the target network DBLPv7, obtained through various methods. Different colors indicate distinct label categories. The figure clearly illustrates that, compared to the baseline methods UDAGCN, AdaGCN, and ACDNE, our proposed method achieves superior classification performance by more effectively distinguishing nodes with different label categories.



(a) UDAGCN (b) AdaGCN (c) ACDNE (d) NQDU

Figure E1 The visualization analysis of UDAGCN, AdaGCN, ACDNE, and NQDU under the task of Citationv1→DBLPv7.

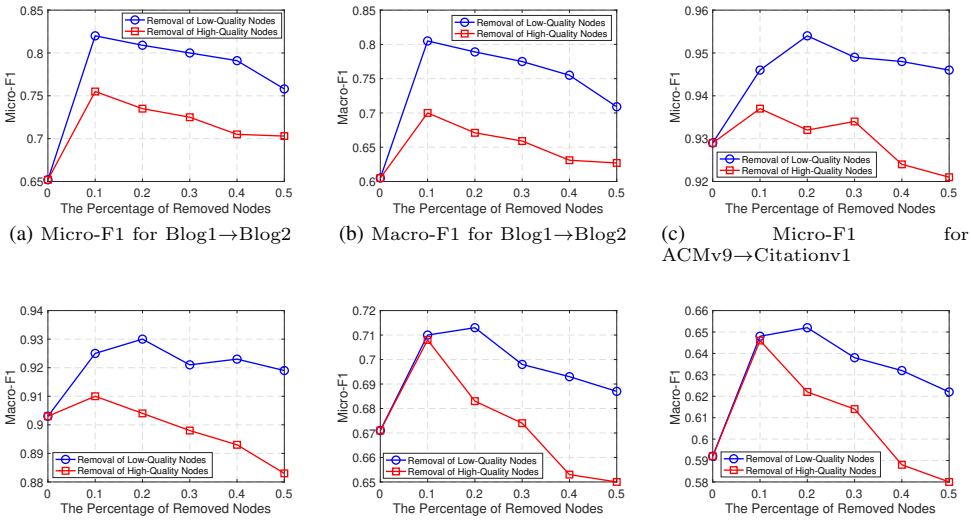
Result 2 Feature Selection Criteria for different Datasets (for Q2): To answer **Q2**, we analyzed the three network datasets used in the experiment to determine the most effective features for selecting high-quality nodes to enhance the classification accuracy of cross-network node classification tasks.

As shown in TABLE E1, in social networks, incorporating both attribute and topological structure features as inputs to the node quality evaluator leads to slightly better performance. This is likely due to the higher dimensionality and richness of attribute features in social networks, which contain fewer nodes. Consequently, attribute features alone are more informative than topological structure features alone, resulting in marginally better results.

In TABLE E2, the citation networks show similar classification results when using combined attribute and topological structure features compared to using only attribute features. This is because attribute features in citation networks inherently contain more information relevant to node classification. Therefore, attribute features significantly impact the prediction results, while topological structure features contribute less. However, since the classifier model utilizes the full set of node features for training, the final classification results obtained using only topological structure features as input to the node quality evaluator still outperform other baseline methods.

As reported in TABLE E3, in PPI networks, attribute features include motif sets and immunological characteristics, while structural features encompass protein positions and connection relationships. The node labels to be predicted indicate the protein's properties and location information. Using either attribute or structural features alone might yield the best prediction results. The results in TABLE E2 show that for most transfer tasks, utilizing only topological structure features provides better results, indicating that topological structure features are richer and more beneficial for node classification tasks in PPI networks. Additionally, the classification results improve when using combined features, suggesting that integrating features is more advantageous for node classification tasks. However, when topological structure features are processed using the PPMI metric, the results for most transfer tasks are slightly worse than those obtained using combined attribute and original topological structure features. This decline might be due to potential information loss during the compression process when applying PPMI to the original topological structure features, resulting in slightly inferior final classification results.

Result 3 Impact of High-Quality Node Selection (for Q3): To examine the impact of high-quality node selection on node classification performance in the target network, we utilized the selection probabilities obtained from the node quality evaluator. We incrementally removed low/high-quality nodes from the source network in proportions of 10%, 20%,



(a) Micro-F1 for Blog1→Blog2 (b) Macro-F1 for Blog1→Blog2 (c) Micro-F1 for ACMv9→Citationv1

(d) Macro-F1 for ACMv9→Citationv1 (e) Micro-F1 for PPI-1→PPI-2 (f) Macro-F1 for PPI-1→PPI-2

Figure E2 The experimental result of cross-network node classification with respect to the removal of low- and high-quality nodes.

30%, 40%, and 50%, and used the selected nodes for model training. The node classification results in the target network are presented in Figure E2.

Figure E2 shows that removing low-quality nodes from the source network enhances the classification performance in the target network. The notable performance improvement is observed after removing 10%-20% of the low-quality nodes. This indicates that filtering nodes based on the selection probabilities learned by the node quality evaluator and utilizing high-quality nodes for node embedding significantly benefits the target node classification task. However, as the proportion of removed nodes increases, model performance declines, especially in PPI networks with fewer nodes (As shown in the supplementary file). This suggests that having a higher number of nodes provides richer network information, which is advantageous for predicting node labels in the target network.

Additionally, removing a certain proportion of high-quality nodes also enhances node label prediction performance. This suggests that while high-quality nodes are more similar to the nodes in the target network, other lower-quality nodes also play a crucial role in learning node embeddings in the target network. Exclusively selecting either low or high-quality nodes might not accurately reflect the actual network and fails to provide a comprehensive evaluation of the entire network. Nodes not selected still contribute to learning node embeddings to some extent, providing feature information inherent to the network, thereby improving classification outputs.

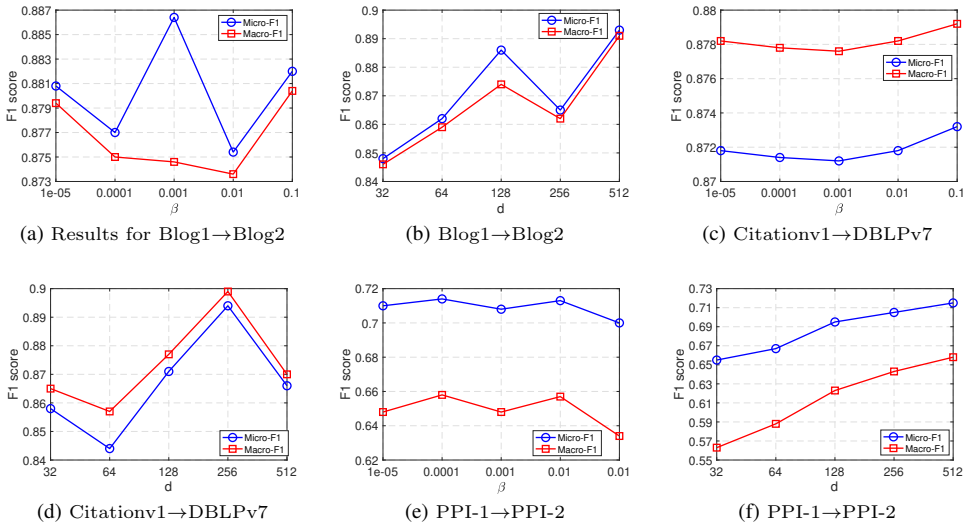
In summary, removing outliers or noisy node from the network and selecting higher-quality nodes for learning can significantly enhance the model's classification performance. Nevertheless, a larger number of nodes can offer more diverse information, thereby improving classification accuracy. Nodes not selected still play a role in the overall feature learning of the network, providing feature information that benefits model classification performance.

Result 4 Impact of NQDU framework for Existing CNNC methods (for Q4): To answer **Q4**, we integrated three SOTA baseline methods for cross-network node classification, i.e., A2GNN [29], TFGDA [30], and HOGDA [31], into our proposed NQDU framework and conducted experiments on citation network datasets. The experimental results are presented in TABLE E4. As shown TABLE E4, all three methods exhibit substantial improvements in node classification accuracy when integrated with our framework. Specifically, under the Micro-F1 metric, the classification accuracy improves by 18.07%, 16.04%, and 15.19%, respectively, compared to the original methods. Similarly, under the Macro-F1 metric, the improvements are 16.09%, 15.76%, and 16.01%, respectively, further validating the effectiveness of our proposed NQDU framework.

Table E4 Cross-network mode classification accuracy (%) comparison between baselines and baselines integrated with our NDQU framework.

Methods	C→D		C→A		D→A		Avgerage		Improvement	
	Macro	Micro	Macro	Micro	Macro	Micro-F1	Macro	Micro	Macro	Micro
A2GNN	77.04	78.13	77.57	76.15	75.69	74.12	76.77	76.13	-	-
NDQU+A2GNN	84.12	86.89	97.80	97.97	96.66	97.73	92.86	94.20	**16.09** ↑	**18.07** ↑
TFGDA	72.60	76.00	72.30	73.60	64.30	66.90	69.73	72.17	-	-
NDQU+TFGDA	81.04	85.32	90.35	91.59	85.09	87.73	85.49	88.21	**15.76** ↑	**16.04** ↑
HOGDA	72.90	77.10	72.60	74.40	65.10	68.20	70.20	73.23	-	-
NDQU+HOGDA	81.61	85.44	91.15	91.79	85.86	88.04	86.21	88.42	**16.01** ↑	**15.19** ↑

Moreover, these results lead to the following conclusion: existing CNNC approaches primarily focus on reducing the

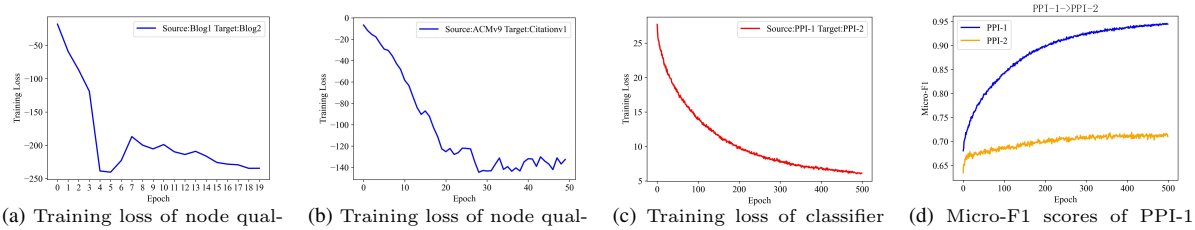


(a) Results for Blog1→Blog2 (b) Blog1→Blog2 (c) Citationv1→DBLPv7

(d) Citationv1→DBLPv7 (e) PPI-1→PPI-2 (f) PPI-1→PPI-2

Figure E3 The experimental result of cross-network node classification with respect to the trade-off parameter β and the dimension of d.

distribution discrepancy between the source and target networks at the algorithmic level, without fundamentally addressing the issue of data quality (e.g., data noise) within networks. The presence of low-quality data samples limits the performance of these methods. Therefore, enhancing data quality has a significant and positive impact on the performance of CNNC.



(a) Training loss of node quality evaluator on social networks (b) Training loss of node quality evaluator on citation networks (c) Training loss of classifier model on PPI networks (d) Micro-F1 scores of PPI-1 and PPI-2

Figure E4 The convergence analyses of the node quality evaluator and the classifier model.

Result 5 Impact Analysis of Parameters β and d (for Q5): First, to evaluate the impact of the learning rate β for the node quality evaluator, we kept other parameters unchanged and varied β at 10^{-5}, 10^{-4}, 0.001, 0.01, and 0.1. We then observed the resulting changes in Micro-F1 and Macro-F1 scores across three transfer tasks: Blog1→Blog2 and Citationv1→DBLPv7. Figure E3(a) and Figure E3(c) show the variations in Micro-F1 and Macro-F1 scores within each transfer task as β changes. In Figure E3(a), for the Blog1→Blog2 transfer task, the highest Micro-F1 score is achieved with β set to 0.001, while both Micro-F1 and Macro-F1 scores are higher when β is 0.1. Figure E3(c) illustrates that for the Citationv1→DBLPv7 transfer task, changes in β result in minimal fluctuations in Micro-F1 and Macro-F1 scores, indicating a negligible impact of β on classification performance.

Then, to evaluate the effect of the dimensionality d of the learned node embeddings, we varied d to 32, 64, 128, 256, and 512, keeping all other parameters unchanged. We then observed the resulting changes in Micro-F1 and Macro-F1 scores across the specified transfer tasks. Figure E3(b) and Figure E3(d) demonstrate that different values of d yield varying Micro-F1 and Macro-F1 scores, underscoring the significant influence of embedding dimensionality on classification performance. Figure E3(b) reveals that for the Blog1→Blog2 transfer task, increasing d generally enhances both Micro-F1 and Macro-F1 scores. Despite a decrease in performance at $d = 256$, the classification accuracy improves further, reaching its highest value when d is increased to 512. Figure E3(d) shows that for the Citationv1→DBLPv7 transfer task, the best classification result is achieved with d set to 256.

Result 6 Model Convergence of NQDU (for Q6): To validate the effectiveness of the node quality evaluator during training, we tracked the loss function values by recording the training loss at each epoch and plotting the curves to analyze the trend. A gradual decrease in the loss function values with increasing epochs, followed by stabilization at a lower value, indicates model convergence.

Figs. E4(a) and E4(b) illustrate the training loss curves for the node quality evaluator in two transfer tasks: social networks and citation networks. These figures clearly demonstrate a continuous decrease in loss values as training progresses, signifying that the node quality evaluator is successfully learning and optimizing. As the number of epochs increases, the

loss curve's descent slows and eventually stabilizes, indicating that the model has reached convergence.

Additionally, to evaluate the convergence of the classifier model, we recorded the training loss in the source network, along with the Micro-F1 scores in both the source and target networks throughout the training process. Figure E4(c) and E4(d), which depict the changes in PPI network tasks, show that the training loss decreases and stabilizes as epochs increase. Concurrently, the Micro-F1 curves for both the source and target networks also stabilize, verifying that the model has converged.

References

1 Y. Wang, C. Feng, L. Chen, H. Yin, C. Guo, and Y. Chu, "User identity linkage across social networks via linked heterogeneous network embedding," *World Wide Web*, vol. 22, no. 6, pp. 2611–2632, 2019.

2 S. H. Lee, D. Hwang, T.-W. Goo, and E.-Y. Yun, "Prediction of intestinal stem cell regulatory genes from drosophila gut damage model created using multiple inducers: Differential gene expression-based protein-protein interaction network analysis," *Developmental & Comparative Immunology*, vol. 138, p. 104539, 2023.

3 M. Xu, D. L. Sanz, P. Garces, F. Maestu, Q. Li, and D. Pantazis, "A graph gaussian embedding method for predicting alzheimer's disease progression with meg brain networks," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 5, pp. 1579–1588, 2021.

4 K. Liu, X. Sun, L. Jia, J. Ma, H. Xing, J. Wu, H. Gao, Y. Sun, F. Boulnois, and J. Fan, "Chemi-net: a molecular graph convolutional network for accurate drug property prediction," *International Journal of Molecular Sciences*, vol. 20, no. 14, p. 3389, 2019.

5 Y. Zhou and H. Li, "Asset diversification and systemic risk in the financial system," *Journal of Economic Interaction and Coordination*, vol. 14, pp. 247–272, 2019.

6 M. Xu, "Understanding graph embedding methods and their applications," *SIAM Review*, vol. 63, no. 4, pp. 825–853, 2021.

7 T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

8 P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.

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

10 D. Sánchez, L. Servadei, G. N. Kiprit, R. Wille, and W. Ecker, "A comprehensive survey on electronic design automation and graph neural networks: Theory and applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 2, pp. 1–27, 2023.

11 J. Wu, J. He, and E. Ainsworth, "Non-iid transfer learning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 10342–10350, 2023.

12 Y. Jin, K. Chen, and Q. Yang, "Selective cross-city transfer learning for traffic prediction via source city region re-weighting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 731–741, 2022.

13 S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

14 Y. Zhang, G. Song, L. Du, S. Yang, and Y. Jin, "Dane: domain adaptive network embedding," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4362–4368, 2019.

15 X. Shen, Q. Dai, S. Mao, F.-l. Chung, and K.-S. Choi, "Network together: Node classification via cross-network deep network embedding," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

16 Q. Dai, X.-M. Wu, J. Xiao, X. Shen, and D. Wang, "Graph transfer learning via adversarial domain adaptation with graph convolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4908–4922, 2022.

17 X. Shen, Q. Dai, F.-l. Chung, W. Lu, and K.-S. Choi, "Adversarial deep network embedding for cross-network node classification," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, vol. 34, pp. 2991–2999, 2020.

18 M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised domain adaptive graph convolutional networks," in *Proceedings of the Web Conference 2020*, pp. 1457–1467, 2020.

19 H. Yang, H. He, W. Zhang, and Y. Bai, "Mtgk: Multi-source cross-network node classification via transferable graph knowledge," *Information Sciences*, vol. 589, pp. 395–415, 2022.

20 L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

21 K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

22 J. Yoon, S. Arik, and T. Pfister, "Data valuation using reinforcement learning," in *Proc. ICML*, pp. 10842–10851, PMLR, 2020.

23 P. Swazinna, S. Udluft, and T. Runkler, "Measuring data quality for dataset selection in offline reinforcement learning," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2021.

24 A. Abolfazli, G. Palmer, and D. Kudenko, "Data valuation for offline reinforcement learning," *arXiv e-prints*, pp. arXiv–2205, 2022.

25 B. Wu, X. Liang, X. Zheng, J. Wang, and X. Zhou, "Reinforced sample selection for graph neural networks transfer learning," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1281–1288, IEEE, 2022.

26 J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1041–1050, 2015.

27 X. Zhang, Y. Du, R. Xie, and C. Wang, "Adversarial separation network for cross-network node classification," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2618–2626, 2021.

28 Y. You, T. Chen, Z. Wang, and Y. Shen, "Graph domain adaptation via theory-grounded spectral regularization," in *The 11th International Conference on Learning Representations*, 2023.

29 M. Liu, Z. Fang, Z. Zhang, M. Gu, S. Zhou, X. Wang, and J. Bu, "Rethinking propagation for unsupervised graph domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 13963–13971, 2024.

30 J. Dan, W. Liu, X. Xie, H. Yu, S. Dong, and Y. Tan, "Tfgda: Exploring topology and feature alignment in semi-supervised graph domain adaptation through robust clustering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 50230–50255, 2024.

31 J. Dan, W. Liu, M. Liu, C. Xie, S. Dong, G. Ma, Y. Tan, and J. Xing, "Hogda: Boosting semi-supervised graph domain adaptation via high-order structure-guided adaptive feature alignment," in *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 11109–11118, 2024.