Towards Graph Foundation Model: Node Feature Transfer Invariant Modeling on General Graphs

Jitao Zhao[†]
zjtao@tju.edu.cn
Tianjin University
Tianjin, China

Yi Wang[†]
wangyi076@tju.edu.cn
Tianjin University
Tianjin, China

Yawen Li
warmly0716@126.com
Beijing University of Posts and Telecommunications
Beijing, China

Dongxiao He[*]
hedongxiao@tju.edu.cn
Tianjin University
Tianjin, China

Di Jin
jindi@tju.edu.cn
Tianjin University
Tianjin, China

Zhiyong Feng
zyfeng@tju.edu.cn
Tianjin University
Tianjin, China

Weixiong Zhang
weixiong.zhang@polyu.edu.hk
The Hong Kong Polytechnic University
Kowloon, Hong Kong

Abstract

A significant challenge in developing graph foundational models is to achieve universality and generalizability across general graphs, beyond text-attribute graphs. Unfortunately, most graph neural networks are designed for specific applications in particular fields and are difficult to generalize across different graphs. The most critical problem is the lack of Transfer Invariant Metadata (TIM) for graphs, akin to pixels for images and vocabularies for text, which prevents the development of graph foundational models. TIM is particularly problematic, albeit critical, on graph nodes with semantically variable features, making it difficult to transfer knowledge across graphs. Here, we analyze TIM and propose a theoretical approach to mining TIM in node features. It extracts semantically consistent information across domains and unifies data space with relatively low information loss. We then introduce a Transfer-Invariant Graph (TIG) foundational model to transform features of different dimensions into a unified structural representation. This transformation can effectively learn and extract TIM and intrinsic graph knowledge by self-supervised learning. We conducted extensive experiments, and the results showed that TIG even outperformed some models trained on data from targeted domains. In one-shot cross-domain scenarios, TIG achieved high accuracy with less training data and without any prompt tuning.

[†]Both authors contributed equally to this research.
[*]Corresponding author.

CCS Concepts

• **Computing methodologies → Neural networks**.

Keywords

Graph Foundation Model, Graph Pre-training, Graph Representation Learning

ACM Reference Format:
Jitao Zhao, Yi Wang, Yawen Li, Dongxiao He, Di Jin, Zhiyong Feng, and Weixiong Zhang. 2018. Towards Graph Foundation Model: Node Feature Transfer Invariant Modeling on General Graphs. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the remarkable success of foundation models and Large Language Models (LLMs), Graph Foundation Models (GFMs) have recently gained significant attention [12]. Research on GFMs centers on developing universal and intelligent graph models that leverage extensive graph data to acquire highly generalizable knowledge, and can be directly applied across various data-sparse graph scenarios. Given the inherently complex nature of graph data, the exploration of GFMs is still in its early stages. Nonetheless, GFMs hold great potential to advance applications in domains such as e-commerce [30] and molecular prediction [27].

A primary challenge for GFMs is achieving universality in graph modeling, which involves learning generalizable knowledge applicable across different graphs. In Computer Vision (CV) and Natural Language Processing (NLP), foundational models benefit from semantically consistent data or shared features, such as pixels in images and vocabularies in text. This commonality, termed Transfer-Invariant Metadata (TIM), enables easy generalization. However, TIM is absent in graphs, which are manually defined abstract representations of diverse real-world objects. Graphs from different domains vary widely in structure, node features, and semantics. Although some GFMs address structural universality [18, 38], most Graph Neural Networks (GNNs) struggle to generalize when node features differ significantly in dimension and meaning. Even within the same domain, discrepancies in node features often necessitate rebuilding and retraining models for adaptation [37]. While GFMs combined with LLMs achieve universality for text-attributed graphs, their performance remains suboptimal for general graph, significantly limiting the capabilities of graph models.

We categorize the challenge of node feature universality into two aspects: form unification and cross-domain transferability. **Form unification** involves aligning node features of varying dimensions into a consistent format to meet the fixed input dimension requirements of GNNs, while preserving semantics relevant to downstream tasks. Although essential for formal universality, the more critical aspect is **cross-domain transferability**, i.e., the unification of node feature semantic spaces across domains, similar to a shared vocabulary in NLP [15]. Node features vary widely across applications, for example, diverse word usages, molecular properties, and topological or semantic embeddings. Identifying commonalities in such semantically diverse data to create universally transferable invariant metadata is highly challenging. Without such cross-domain invariant metadata, formal unification alone cannot ensure effective model transferability across different graph datasets.

Recent efforts toward universal graph models have largely overlooked these challenges. Existing methods primarily address form unification but neglect cross-domain transferability. They often rely on data preprocessing techniques, such as Principal Component Analysis (PCA) [21, 35], or use LLMs [11] to align node features arbitrarily within a latent space of uniform dimensionality, frequently resulting in significant information loss. While LLMs handle graphs with textual features effectively, they perform suboptimally on general graphs dominated by numeric data, as evidenced by the FUG study [37]. Two notable recent works, GraphAny and FUG, demonstrate promising transferability on general graphs [37, 38]. However, GraphAny depends on labels and retraining for each test dataset, limiting its practical application, while FUG depends on data sampling and lacks robustness. Therefore, these approaches succeed only in formal feature unification, failing to guarantee transfer-invariance critical for cross-domain generalization.

To address these challenges, we investigate the hidden Transfer-Invariant Metadata (TIM) within node features. Drawing inspiration from pixels in CV and vocabularies in NLP, we propose a theory to mine latent TIM in general graph node features. This approach extends beyond traditional node encoding by leveraging relational information across different feature dimensions to extract TIM, thereby ensuring a unified embedding space. Therefore, this theory can effectively address the two previously mentioned problems to facilitate cross-domain unification of node features on general graphs. Building on this theoretical foundation, we propose a Transfer Invariant Graph foundation model (TIG). TIG considers each dimension of node features as a new node, thereby transforming raw features into transfer-invariant structures and edge weights. Additionally, dimensional augmentation is applied to these new nodes to capture transfer-invariant dimension semantics. TIG integrates an advanced graph pre-training technique [6] to self-supervise the learning of broadly generalizable graph knowledge.

In summary, we make the following contributions in this paper:

- **Theory.** We propose a novel theory for extracting TIM from node features in general graphs. To our knowledge, this is the first effort to derive such metadata from highly diverse node features, providing crucial guidance for the advancement of GFMs. Furthermore, the TIM theory holds potential for extension to broader data types beyond graphs.
- **Method.** Building on this theory, we introduce a transfer-invariant graph foundational model that unifies the node feature space with relatively low information loss while extracting TIM. This enables self-supervised learning to acquire cross-domain graph knowledge.
- **Experiment.** We conduct extensive experiments to validate the effectiveness of our theory and model across in-domain, cross-domain, and few-shot setting on six widely used datasets. In cross-domain tests, TIG even outperforms some in-domain models. For cross-domain one-shot scenarios, TIG achieves state-of-the-art performance without prompt tuning and with reduced training data.

2 Related Work

Graph Neural Networks (GNNs). GNNs are one of the most powerful methods for graph representation learning[40]. They typically follow the paradigm $Z = \sigma(\hat{A}XW)$, which simultaneously encodes both structural and feature information of the graph. Here, $\hat{A}$ is the propagation matrix, X is the node feature matrix, W represents learnable parameters, and σ is the activation function. GCN [8] is a classic and powerful GNN that directly aggregates information from neighboring nodes. GAT [25] computes attention weights between each pair of nodes to adaptively aggregate topological information. However, these methods are often constrained by the fixed input dimensionality requirements of the neural network parameters W, meaning they are typically designed for small, domain-specific models. To adapt to a new dataset, a new model must be rebuilt and trained, leading to high computational cost and poor scalability.

Graph Self-Supervised Learning. Training GNNs requires a large amount of labeled data. Graph self-supervised learning aims to mine latent information from data as self-supervised samples to train models, thereby reducing dependence on labeled data. Self-supervised learning in graphs primarily includes generative methods based on auto-encoders and contrastive pretraining approaches. Generative methods encode the original graph into representations and then reconstruct the graph structure for training, such as GAE and VGAE [7]. Contrastive methods train the GNNs by identifying positive and negative samples in the data, ensuring that nodes with similar semantics have close representations, while those with different semantics are uniformly distributed on a hyper-sphere. GRACE [43] treats different augmentations of the same node as positive samples and different nodes as negative samples. BGRL [23] trains solely by predicting the bootstrap output of positive samples. However, most existing graph self-supervised learning methods still suffer from the inherent limitations of GNNs, making it challenging for them to generalize effectively to new datasets.

Universal Graph Foundational Models. With the remarkable generalization capabilities demonstrated by large language models (LLMs) [39], researchers in the graph domain have increasingly explored universal graph models that can generalize across arbitrary datasets and downstream tasks [14]. Universal graph models typically follow a pre-training and fine-tuning framework. One of the earliest works to achieve cross-dataset generalization was GCC [16], which primarily focused on structural encoding but was limited to graphs without node features. More recent efforts can be categorized into two main approaches: LLM-integrated methods and non-LLM-based methods. LLM-integrated methods leverage LLMs to encode graph features or structures, enabling cross-dataset generalization such as OFA [11]. However, these methods are inherently constrained by LLMs, making them effective for graphs with textual features, while ineffective for graphs with numerical features. Non-LLM-based methods arbitrarily unify data dimensions through techniques such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), or similarity embeddings, as seen in AnyGraph [31] and GraphControl [42]. However, these approaches often loss valuable information in data preprocessing and fail to preserve cross-domain semantic consistency. More recently, advanced methods such as FUG [37] and GraphAny [38] have been proposed to accommodate diverse graph data. FUG introduces the idea of dimensional encoding, where each feature distribution is encoded from a set of node samples to generate universal representations. However, this approach relies on the sampling method, and mismatches between the new dataset's sampling distribution and the original dataset can lead to suboptimal performance. GraphAny employs multiple linearGNNs with a topology selector, thereby enabling applicability to both homophily and heterophily structures. However, GraphAny still requires labeled data for each new dataset to reconstruct the linearGNN, limiting its universality.

3 Preliminary

General Graph. Given a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ represents the set of nodes, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ represents the set of edges. The graph has a numerical node feature matrix $X \in \mathbb{R}^{n \times d}$ and an adjacency matrix A, where $a_{ij} = 1$ if $e_{ij} \in \mathcal{E}$, or 0 otherwise. A collection of various general graph datasets G forms a dataset set $\mathcal{D}_{\text{GG}} = \{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, ..., \mathcal{G}^{(n)}\}$.

Graph Foundation Model. Derived from the concept of foundational models in CV and NLP, a graph foundation model refers to a model that can be trained on a broad range of graph datasets and applied to a variety of downstream scenarios [12]. Given a graph foundation model $f(\cdot)$, it is trained using a general supervised or self-supervised loss, i.e., $\min \sum_{\mathcal{G}^{(i)} \in \mathcal{D}_{\text{GG}}^{\text{train}}} L(f(\mathcal{G}^{(i)}))$, and for any new dataset $\mathcal{G}^{(k)} \in \mathcal{D}_{\text{GG}}^{\text{test}}$, the model can be directly adapted to downstream tasks DownTask $= p(f(\mathcal{G}^{(k)}))$, where $p(\cdot)$ is a fine-tuning or prompt strategy for the downstream task.

4 Transfer-Invariant Metadata in Graphs

We now analyze Transfer-Invariant Metadata (TIM) to understand how it can be effectively extracted from general graphs.

Brief Overview. From the fields of CV and NLP, we deduce that extracting TIM requires using the same extraction method (Ext$(\cdot)$) to derive the same physical properties (Asp$(\cdot)$) of any object (R). Classical neural networks encode all features ($x_{i,:}$) for a node (v_i), but these node features in different graphs often do not satisfy TIM. TIM theory proposes transforming objects from individual nodes to the set of all features ($\mathcal{A}^{(k)}$) within a dataset (DS$^{(k)}$), and using a consistent extraction method ($\text{Ext}_{\mathcal{A}^{\text{Raw}}}(\cdot)$) to embed the same physical property, which is the relationship among elements within this set, thereby acquiring augmented TIM attributes for each feature. Ultimately, these feature attributes are propagated back to the node to obtain the node's TIM embeddings.

4.1 Transfer-Invariant Metadata

Definition 4.1 (TIM)**.** Given a set of datasets $\mathcal{D}$:

$$\begin{aligned} &\mathcal{D} = \{\text{DS}^{(1)}, \text{DS}^{(2)}, \ldots, \text{DS}^{(N_{\text{DS}})}\}, \\ &\text{DS}^{(i)} = \{\text{Data}_1^{(i)}, \text{Data}_2^{(i)}, \ldots, \text{Data}_{N_{\text{Data}^{(i)}}}^{(i)}\}, \\ &\text{Data}_j^{(i)} = \{\text{Ext}_1(\text{Asp}_1(R_j^{(i)})), \ldots, \text{Ext}_{N_A}(\text{Asp}_{N_A}(R_j^{(i)}))\}, \end{aligned} \tag{1}$$

where DS is the datasets, and sample $\text{Data}_j^{(i)}$ is a representation of real-world objects $R_j^{(i)}$, $\text{Asp}_k(R_j^{(i)})$ denotes the k-th physical property of $R_j^{(i)}$, and $\text{Ext}_k(\cdot)$ represents the mathematical extraction method applied to that property. If there exists a fixed $\text{Ext}_t(\cdot)$ and $\text{Asp}_t(\cdot)$ satisfying:

$$\begin{aligned} &\forall\, \text{DS}^{(i)} \in \mathcal{D},\ \text{Data}_j^{(i)} \in \text{DS}^{(i)}, \exists\, M_j^{(i)}, \\ &s.t.\ M_j^{(i)} = \text{Ext}_t(\text{Asp}_t(R_j^{(i)})) \in \text{Data}_j^{(i)}, \end{aligned} \tag{2}$$

we then say that $\mathcal{D}$ contains transfer-invariant metadata M with respect to R.

We use Asp, Ext, R and M without subscripts to represent their corresponding sets. Definition 4.1 essentially states that TIM represents data extracted with the same method $\text{Ext}_t(\cdot)$ and from the same physical property $\text{Asp}_t(\cdot)$ across datasets. To facilitate understanding, consider an example from CV, where pixels serve as TIM. For an image set R_{CV}, $\text{Asp}_{\text{CV}}(\cdot)$ refers to the optical property of objects and $\text{Ext}_{\text{CV}}(\cdot)$ converts this property to a numerical matrix using RGB channels. Since both $\text{Asp}_{\text{CV}}(\cdot)$ and $\text{Ext}_{\text{CV}}(\cdot)$ are consistent across datasets, RGB-based pixels are TIM. The invariance of $\text{Asp}_t(\cdot)$ ensures data semantics are consistent, while the invariance of $\text{Ext}_t(\cdot)$ ensures the semantics are encoded in a uniform mathematical form. Obviously, there is no TIM in the raw node features in the set of general graphs $\mathcal{D}_{\text{GG}}$ with respect to the nodes $\mathcal{V}$.

Corollary 4.2 (Nesting Property of TIM)**.** *Consider the setting where $M_j^{(i)} = Ext_a(Asp_a(R_j^{(i)}))$ is TIM, if there exists $\widehat{M}$ along with invariant $Ext_b(\cdot)$ satisfying:*

$$\begin{aligned} &\forall\, DS^{(i)} \in \mathcal{D},\ Data_j^{(i)} \in DS^{(i)}, \exists\, \widehat{M}_j^{(i)}, \\ &s.t.\ \widehat{M}_j^{(i)} = Ext_b(M_j^{(i)}) \in Data_j^{(i)}, \end{aligned} \tag{3}$$

then $\widehat{M}$ is also TIM. Conversely, if M is not TIM, then $\widehat{M}$ cannot be TIM.

Brief proof of Corollary 4.2. Let $\text{Asp}_b(\cdot) = \text{Ext}_a(\text{Asp}_a(\cdot))$. If M satisfies TIM, then $\text{Asp}_a(\cdot)$ and $\text{Ext}_a(\cdot)$ are invariant across $\mathcal{D}$, implying $\text{Asp}_b(\cdot)$ is also invariant. Consequently, $\widehat{M} = \text{Ext}_b(\text{Asp}_b(\cdot))$ satisfies the TIM definition. Conversely, if $\text{Asp}_b(\cdot)$ is not invariant, $\widehat{M}$ cannot be TIM.

Discussion 1. TIM focuses solely on the cross-domain invariance of data, without providing any guarantee regarding its reliability or sufficiency for downstream tasks. Specifically, when data satisfy the TIM property, it means that the data encapsulate the same semantics across different domains. However, this does not imply that it will provide sufficient information. In practice, it is impossible to enumerate all possible $\mathrm{Asp}(\cdot)$. Each extraction of features from R inevitably results in some information loss. Therefore, extracting TIM must consider how well the extracted data aligns with the requirements of the downstream tasks.

Discussion 2. Existing methods are ineffective at mining TIM. (1) Research [3] on out-of-distribution generalization in graphs is unrelated to the focus of this paper and is unable to extract TIM from node features. These approaches typically involve learning a specific domain adaptation mapping, $\mathrm{Proj}(\cdot)$, for two different datasets, A and B, to satisfy:

$$\text{Minimize DomDif}\Big[\mathrm{Proj}_A(\mathrm{Ext}_A(\mathrm{Asp}_A(R_A))),\ \mathrm{Proj}_B(\mathrm{Ext}_B(\mathrm{Asp}_B(R_B)))\Big], \tag{4}$$

here, $\mathrm{DomDif}(\cdot,\cdot)$ measures the distributional differences between the two domains. These methods focus on generalizing knowledge between two specific domains, while we aim to improve the data generalization across all general graphs. (2) The latest works, such as FUG [37] and GraphControl [42] introduce techniques based on dimensional encoding and inter-node similarity encoding to address the transferability of features. However, FUG depends on data sampling and therefore cannot preserve complete dataset information, while GraphControl maps high-dimensional features into a single-dimensional connection weight. Both approaches may lead to substantial information loss.

Discussion 3. Definition 4.1 means that $\mathrm{Ext}(\cdot)$ and $\mathrm{Asp}(\cdot)$ must remain invariant across domains, whereas R may vary. This follows common knowledge about TIM in CV, where an image could be an animal, a building, or a person. This suggests that the key to discovering TIM lies in identifying an appropriate R. Moreover, as highlighted in Discussion 1, R and $\mathrm{Asp}(\cdot)$ must provide sufficient information to support diverse tasks.

4.2 Mining TIM in Features with Low Information Loss

We now consider how to mine TIM from node features. As discussed in Section 4.1, the key to finding TIM lies in identifying domain-invariant Ext, Asp, and appropriate real-world objects R. In addition, as noted in Section 1, when mining TIM, it is essential to ensure that it maintains a unified form to fit the fixed input of neural networks, while not losing information in features. We define the original graph node features as $x_{ij} = \mathrm{Ext}_j^{\mathrm{Raw}}(\mathrm{Asp}_j^{\mathrm{Raw}}(R_i^{\mathrm{Raw}}))$, where R_i^{Raw} corresponds to node v_i.

Reconstruction of R. From Definition 4.1, $\mathrm{Asp}^{\mathrm{Raw}}$ is the artificial selection of the physical properties of R, which can vary significantly across different graph datasets. Without prior knowledge, it is impossible to understand the meaning of $\mathrm{Asp}^{\mathrm{Raw}}$. Therefore, transforming $\mathrm{Asp}^{\mathrm{Raw}}$ into part of the entity R, could help eliminate this variability. The subsequent TIM mining theory is centered around three reconstructions of R. To facilitate understanding, we briefly introduce the reconstructions here: 1) Treat node v_i and $\mathrm{Asp}_j^{\mathrm{Raw}}$ as one object, forming R^1. 2) Consider all $\mathrm{Asp}_j^{\mathrm{Raw}}$ within a graph as one object, forming R^2. 3) Take node v_i and all its related $\mathrm{Asp}_j^{\mathrm{RAW}}$ as one object, forming R^3.

R^1 is formally constructed as follows:

$$R^1 = \{(v_i, \mathrm{Asp}_j^{\mathrm{Raw}}) | v_i \in R^{\mathrm{Raw}}, \mathrm{Asp}_j^{\mathrm{Raw}} \in \mathrm{Asp}^{\mathrm{Raw}}\},$$
$$x_{ij} = \mathrm{Ext}_j^{\mathrm{Raw}}(\mathrm{Asp}_j^{\mathrm{Raw}}(v_i)) = \mathrm{Ext}_j^{\mathrm{Raw}}(R_{ij}^1) = \mathrm{Ext}_j^{\mathrm{Raw}}(v_i, \mathrm{Asp}_j^{\mathrm{Raw}}), \tag{5}$$

where the meaning of x_{ij} changes to represent the relationship between the node v_i and $\mathrm{Asp}_j^{\mathrm{Raw}}$, rather than the j-th feature of v_i. R^1 eliminates the variability of $\mathrm{Asp}^{\mathrm{Raw}}$. Next, we analyze $\mathrm{Ext}^{\mathrm{Raw}}$.

Assumption 4.3 (The assumption of $\mathrm{Ext}(\cdot)$). All $\mathrm{Ext}_j^{\mathrm{Raw}}(\cdot)$ satisfy the signal strength encoding $\mathrm{Sig}(\cdot)$ on the physical properties of the object $\mathrm{Asp}_i^{\mathrm{Raw}}$ and are scaled by a matrix E_j.

Theorem 4.4. *When Assumption 4.3 holds, let $\widehat{X}$ represent the normalized node feature, then $\widehat{X}$ is TIM with respect to R^1 in the dataset set $\mathcal{D}_{GG}$.*

The proofs of Theorem 4.4 and Assumption 4.3 are given in Appendix A.1. Assumption 4.3 is stringent; however, the subsequent theory generally holds true even if this assumption is not satisfied. The $\mathrm{Sig}(\cdot)$ converts physical properties into a form of latent numerical signals, which are then processed into values by a matrix E_j. E_j suggests that this process is continuous. For instance, when processing temperature, $\mathrm{Sig}(\cdot)$ extracts the intensity of the temperature, and then E_j converts it into Celsius or Fahrenheit. Empirically, features can be categorized as discrete and continuous. Continuous features satisfy Assumption 4.3. Discrete features often arise from threshold-based sampling (e.g., age, word presence, one-hot embedding). We convert the discrete feature to the sum of a latent value and a perturbation ϵ. If $\mathbb{E}(\epsilon) = 0$, it can be eliminated through sum/average operations in Corollary 4.5. If $E(\epsilon) \neq 0$, it can be removed through noise-resistant $Ext(\cdot)$ in Theorem 4.6. Assumption can be more flexible in practice (see Appendix A.1 for more details). Extreme cases that invalidate Assumption 4.3 (e.g., using a single feature to represent multiple classes, which is uncommon and performs poorly in most neural networks) are discussed in the Limitations section.

The $\widehat{X}$ is TIM only with respect to R^1 and is not TIM for R^{Raw}. However, R^1 can help construct TIM for R^{Raw}. $\mathrm{Asp}^{\mathrm{Raw}}$ and $\mathcal{V}$ can be considered as two types of heterogeneous nodes, and $\widehat{X}$ represents the transfer-invariant relationship between them. Therefore, we can construct a structure with the domain-invariant physical property of relationships as new $\mathrm{Asp}(\cdot)$ and use the domain-invariant extraction method as a new $\mathrm{Ext}(\cdot)$ to mine TIM for R^{Raw}.

Corollary 4.5. *For a given dataset $DS^{(k)} \in \mathcal{D}_{GG}$, which contains $X^{(k)} \in \mathbb{R}^{n^{(k)} \times d^{(k)}}$, let $\mathcal{A}^{(k)} = \{Asp_1^{Raw(k)}, \ldots, Asp_{d^{(k)}}^{Raw(k)}\}$. Treat elements in $\mathcal{A}^{(k)}$ as nodes, use $\widehat{X}^{(k)}$ to model the structure with a domain-invariant method, and apply a domain-invariant structure encoding method, $Ext_{\mathcal{A}^{Raw}}(\cdot)$, to extract its attributes $X_{Asp}^{(k)}$. Then, X_{Asp} is a TIM in $\mathcal{D}_{GG}$ with respect to $R^2 = \{\mathcal{A}^{(k)} | DS^{(k)} \in \mathcal{D}_{GG}\}$.*

The proof is in Appendix A.2. Here is an intuitive explanation: $\mathrm{Asp}_{A^{\mathrm{Raw}}}(\cdot)$ is a graph construction method based on $\widehat{X}$, which is a TIM and therefore domain-invariant. Meanwhile, $\mathrm{Ext}_{\mathcal{A}^{\mathrm{Raw}}}(\cdot)$ can be any domain-invariant structure encoding method. Both methods are domain-invariant and model the relationships among all $\mathrm{Asp}^{\mathrm{Raw}(k)}$ within $\mathrm{DS}^{(k)}$. Thus, X_{Asp} is a TIM with respect to R^2.

Theorem 4.6. *For any dataset $DS^{(k)} \in \mathcal{D}_{GG}$ containing $\mathcal{A}^{(k)}$, and for any node $v_i \in DS^{(k)}$, model the topology of $(v_i, \mathcal{A}^{(k)})$ using $\widehat{x}^{(k)}_{i,:}$ as the relationship. Let $X^{(k)}_{Asp}$ represent the attributes of $\mathcal{A}^{(k)}$, and employ a domain-invariant method $Ext_{FN}(\cdot)$ to extract its representation $X^{(k)}_{FNi}$. Then X_{FN} is a TIM with respect to $R^3 = \{(v_i, \mathcal{A}^{(k)}) | v_i \in DS^{(k)}, DS^{(k)} \in \mathcal{D}_{GG}\}$ in the dataset set $\mathcal{D}_{GG}$.*

The proof of Theorem 4.6 is in Appendix A.3. Combining all these results, we have theoretically mined TIM from node features. Furthermore, in Theorem 4.6, the original node features X are largely retained within the structure between $\mathcal{V}$ and $\mathcal{A}^{(k)}$, resulting in low information loss. Theorem 4.6 also unifies dimensions through augmentation for each feature (not dimensional reduction). Hence, Theorem 4.6 adequately addresses the two main challenges of node feature universality raised in Section 1.

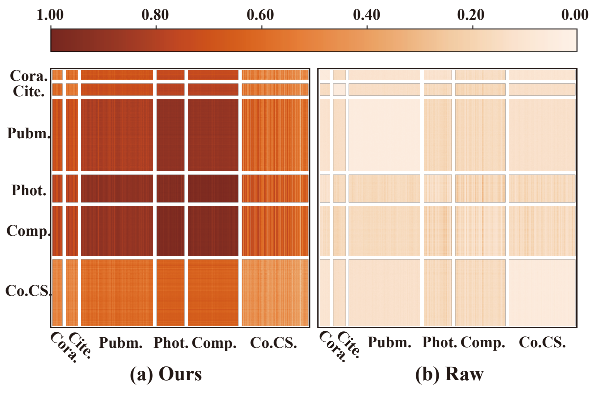

Figure 1: Feature transfer correlations. We calculated the similarity between nodes from general graphs. Ours are pre-trained on Cora to provided the extracted TIM features. For Figure (b), raw features are unified dimensions by SVD.

Experimental Validation. To validate our theory, we evaluated the transfer correlations of raw data and the extracted transfer-invariant metadata (TIM) based on Theorem 4.6, as illustrated in Figure 1. Specifically, we calculate the similarity between each node across different graph datasets. Across all datasets, ours TIM exhibited significantly greater similarity (10x greater than the raw data). This robustly demonstrates that TIM extracted based on Theorem 4.6 indeed possesses domain-invariant semantics. Additionally, these findings offer strong theoretical support for the superior performance of our proposed approach in downstream tasks.

5 Method

Based on the theoretical foundations in Section 4, we proposed a Transfer-Invariant Graph foundation model (TIG). The overview of the model is shown in Figure 2.

Feature TIM Modeling. As described in Section 4, given an input graph $\mathcal{G} \in \mathcal{D}_{GG}$, we first apply Min-Max normalization to obtain the TIM representing the node-Asp$^{\text{Raw}}$ relationship, $\widehat{X}$. Based on this, we structure the features into a new graph $\mathcal{G}^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)})$, where $\mathcal{V}^{(1)} = \mathcal{V} \cup \text{Asp}^{\text{Raw}}$. All nodes in $\mathcal{G}^{(1)}$ have no features. To prevent overly large or small edge weights, we set $e^{(1)}_{ij} = \widehat{x}_{ij} / \sum_j \widehat{x}_{ij}$, which, according to Corollary 4.2, does not alter the properties of the TIM. Next, we model all nodes in the set Asp$^{\text{Raw}}$ together. As in Theorem 4.5, we model the relationships between any two nodes in Asp$^{\text{Raw}}$ through Node-Asp-Node paths, generating the graph $\mathcal{G}^{(2)} = (\mathcal{V}^{(2)}, \mathcal{E}^{(2)})$ with $\mathcal{V}^{(2)} = \text{Asp}^{\text{Raw}}$, and edge weights as follows:

$$e^{(2)}_{ij} = \text{Min-Max}(\sum_{k \in \mathcal{V}} e^{(1)}_{ik} e^{(1)}_{kj}), \tag{6}$$

here, we apply Min-Max normalization to maintain the similarity of structural weights. For $\mathcal{G}^{(2)}$, inspired by existing methods [31], we use an SVD-based algorithm to embed the structure as dimensional augmentation:

$$U, \Lambda, \Gamma = \text{SVD}(\mathcal{E}^{(2)}),\ F^{(2)} = U\sqrt{\Lambda} + \Gamma\sqrt{\Lambda}, \tag{7}$$

where $U, \Gamma \in \mathbb{R}^{n \times s}, \Lambda \in \mathbb{R}^{s \times s}$ are from the SVD of the adjacency matrix of $\mathcal{G}^{(2)}$. Since s are defined by a hyperparameter, the dimension of $F^{(2)}$ can be consistent across different datasets. Based on Theorem 4.5, $F^{(2)}$ is TIM. We then combine $F^{(2)}$ with $\mathcal{G}^{(1)}$ to generate the final TIM embedding, as shown in the equation below:

$$F^{(3)} = f_\phi(F^{(2)}, \mathcal{E}^{(1)}) = A^{(1)} F^{(2)} W_\phi, \tag{8}$$

where $A^{(1)}$ is the adjacency matrix of $\mathcal{E}^{(1)}$, and W_ϕ is a learnable matrix to eliminate disturbances when Assumption 4.3 may not hold, as detailed in Appendix A.1. Consequently, for each node, we extract a TIM embedding representing the relationship between the node and all objects in Asp$^{\text{Raw}}$. It is worth noting that the dimension of $F^{(2)}$ is consistent across different datasets, and all the original feature information is embedded in $A^{(1)}$. As a result, the $F^{(3)}$ fully unifies the original node features dimensions.

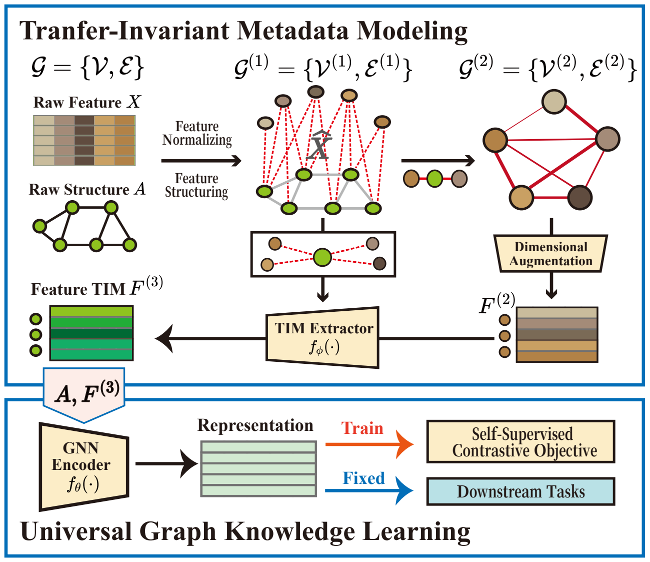
Tranfer-Invariant Metadata Modeling
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
$\mathcal{G}^{(1)} = \{\mathcal{V}^{(1)}, \mathcal{E}^{(1)}\}$
$\mathcal{G}^{(2)} = \{\mathcal{V}^{(2)}, \mathcal{E}^{(2)}\}$
Raw Feature X
Feature Normalizing
Feature Structuring
Raw Structure A
$\widehat{X}$
Dimensional Augmentation
Feature TIM $F^{(3)}$
TIM Extractor $f_\phi(\cdot)$
$F^{(2)}$
$A, F^{(3)}$
GNN Encoder $f_\theta(\cdot)$
Representation
Train
Fixed
Self-Supervised Contrastive Objective
Downstream Tasks
Universal Graph Knowledge Learning

Figure 2: The overview of our TIG.

Self-Supervised Universal Graph Knowledge Learning. We adopt the existing graph pre-training paradigm to achieve cross-dataset self-supervised learning of graph knowledge. TIG builds

Table 1: Performance on cross-domain node classification. Methods with "*" are reported from the original paper. FUG-C (ACG) means the average performance of FUG [37] which were pre-trained and tested on the datasets as we did. TIG (Cora) means the TIG pre-trained on Cora dataset.

Target datasets	Cora	CiteSeer	PubMed	Photo	Computers	CS	Physics
OFA*	75.90 ± 1.26	-	78.25 ± 0.71	-	-	-	-
GraphControl*	-	-	-	89.66 ± 0.56	-	-	94.31 ± 0.12
FUG-C (AVG)	82.45 ± 0.87	67.50 ± 1.98	84.48 ± 0.16	91.58 ± 2.87	84.66 ± 4.85	92.52 ± 0.19	95.36 ± 0.11
TIG (Cora)	84.85 ± 2.29	70.96 ± 2.17	86.05 ± 0.41	92.44 ± 1.17	88.65 ± 0.60	93.11 ± 0.55	95.84 ± 0.31
TIG (CiteSeer)	82.57 ± 3.19	70.96 ± 3.21	86.16 ± 0.62	92.47 ± 0.75	88.63 ± 0.68	92.94 ± 0.60	95.73 ± 0.29
TIG (PubMed)	83.46 ± 2.99	71.02 ± 1.90	85.96 ± 0.53	92.65 ± 0.97	88.58 ± 0.68	93.17 ± 0.44	95.70 ± 0.36
TIG (Photo)	83.09 ± 2.76	71.50 ± 2.34	86.17 ± 0.29	92.76 ± 0.85	88.46 ± 0.67	93.03 ± 0.65	95.83 ± 0.29
TIG (Computers)	83.53 ± 3.83	72.34 ± 2.14	86.35 ± 0.38	92.47 ± 0.70	88.62 ± 0.74	93.04 ± 0.53	95.79 ± 0.26
TIG (CS)	82.50 ± 3.49	71.32 ± 1.79	86.17 ± 0.57	92.55 ± 1.08	88.65 ± 0.93	93.21 ± 0.52	95.71 ± 0.33
TIG (Physics)	83.68 ± 2.32	71.62 ± 2.67	86.32 ± 0.20	92.60 ± 0.75	88.50 ± 0.78	93.21 ± 0.52	95.74 ± 0.34

on the representation scattering concept proposed in SGRL [6], a recent graph contrastive learning method. To reduce model complexity, we omit its asymmetric topological aggregation framework. Specifically, given the original graph structure A and the extracted TIM features $F^{(3)}$, we first generate its representation using a graph neural network $f_\theta(\cdot)$, $Z = f_\theta(A, F^{(3)})$. Next, we train the $f_\theta(\cdot)$ and $f_\phi(\cdot)$ using the existing graph contrastive objective:

$$\mathcal{L}(v_i) = -\,||\mathrm{Norm}(z_i) - \mathrm{Mean}(\mathrm{Norm}(Z))||_2 + \frac{1}{|\mathcal{N}(v_i)|}\sum_{v_j \in \mathcal{N}(v_i)} ||z_i - z_j||_2, \tag{9}$$

where Mean$(\cdot)$ computes the mean vector of all node representations, and Norm$(\cdot)$ denotes vector normalization. This objective directly aligns each node representation with its neighbors while encouraging dispersion across the graph.

6 Experiment

To thoroughly validate the proposed TIM theory and TIG model, we conducted experiments on seven widely used node-level datasets: Cora, CiteSeer, PubMed, Photo, Computer, CS, and Physics. Considering that TIG is a graph foundational model, we completely evaluate TIG in various pre-training and fine-tuning scenarios, which include: (1) Cross-domain pretraining and finetuning scenarios, (2) In-domain self-supervised scenarios, (3) Cross-domain one-shot scenarios (pseudo zero-shot scenario), and (4) Cross-domain few-shot scenarios. Considering that model performance in cross-domain settings may be influenced by various environmental factors, which can potentially prevent existing models from achieving optimal performance. Therefore, for the compared methods, we strive to report the most advanced and reliable results across all scenarios from [35, 37, 44]. More experimental setting details can be found in Appendix B. In all tables, the best results are bolded, and the second-best results are underlined. *In addition, we have supplemented: (a) experiments on graph-level tasks in Appendix C.1, and (b) experiments on molecular graphs in Appendix C.2, the analysis.*

6.1 Performance Analysis

Cross-Domain Scenarios. As shown in Table 1, in cross-domain settings, our TIG consistently outperforms all existing advanced methods when pre-trained on any single dataset. This validates both the correctness of our TIM theory and the strong transferability and generalization ability of the TIG model. Table 1 exhibits an interesting phenomenon: in more than half of the cases, the highest performance is not achieved when the pre-training and test datasets are the same. This indicates that our model can effectively generalize knowledge across different datasets. Therefore, the model can effectively utilize the knowledge learned from other datasets to enhance its performance. The TIM guarantees this transfer ability. Furthermore, the performance differences across different training datasets are minimal, indicating that these datasets share similar feature semantics. This further confirms the consistency of the extracted TIM across domains. Additionally, TIG outperforms FUG, demonstrating that although FUG can unify node feature dimensions, the encoded semantics cannot be invariant across different domains. Additionally, TIG outperforms GraphControl, which constructs a new structure using feature similarity, resulting in a significant loss of information. These results further confirm our view in Discussion 2 of Section 4.

In-Domain Scenarios. As shown in Table 2, TIG achieves either the best or second-best performance in most cases. This first confirms that the TIM extractor can effectively capture information from node features, enabling it to outperform methods that directly use raw features in an in-domain scenario. Additionally, applying SVD and PCA to raw features results in significantly worse performance than TIG, suggesting that TIG's success does not stem from an SVD-like approach but rather from the comprehensive TIM extraction theory.

Cross-Domain 1-shot (pseudo zero-shot) Scenarios. The results are shown in Table 3. In this scenario, TIG does not employ any prompt-tuning strategy. Instead, it just randomly selects a node from each class as the class prototype vector and predicts the node class based on its similarity to the prototype. Since no fine-tuning is involved, we refer to this setting as a pseudo zero-shot scenario.

Table 2: Performance on in-domain scenarios. Methods with "*" are reported from [37].

Method	Cora	CiteSeer	PubMed	Photo	Computers	CS	Physics
Raw features*	57.90 ± 3.90	57.60 ± 2.85	79.55 ± 0.98	80.99 ± 1.65	75.59 ± 1.69	89.92 ± 0.95	93.18 ± 0.45
PCA*	56.36 ± 4.14	48.29 ± 3.18	82.80 ± 0.91	74.92 ± 1.82	71.26 ± 1.34	87.12 ± 0.73	92.48 ± 0.47
SVD	57.87 ± 4.58	63.77 ± 2.28	82.69 ± 0.96	80.34 ± 3.81	71.70 ± 1.67	88.17 ± 0.22	92.22 ± 0.77
GRACE*	83.20 ± 1.87	70.99 ± 2.29	85.46 ± 0.54	91.93 ± 0.83	85.36 ± 0.82	91.84 ± 0.37	OOM
BGRL*	81.57 ± 2.07	70.10 ± 2.04	83.67 ± 0.84	92.34 ± 0.73	86.51 ± 1.53	92.12 ± 0.63	95.42 ± 0.41
GBT*	82.45 ± 1.97	70.16 ± 2.32	85.69 ± 1.22	92.44 ± 0.48	87.46 ± 0.82	92.39 ± 0.78	95.07 ± 0.23
DGI*	83.24 ± 2.12	71.23 ± 2.37	84.62 ± 0.83	92.32 ± 0.49	86.12 ± 0.73	92.47 ± 0.60	94.47 ± 0.50
FUG*	<u>84.45 ± 2.45</u>	72.43 ± 2.92	85.47 ± 1.13	93.07 ± 0.82	88.42 ± 0.98	92.89 ± 0.45	95.45 ± 0.27
SGRL	84.21 ± 2.50	**73.81 ± 1.79**	<u>85.82 ± 0.50</u>	<u>93.73 ± 1.45</u>	**89.51 ± 1.02**	<u>93.24 ± 0.76</u>	<u>95.84 ± 0.44</u>
TIG	**85.51 ± 2.09**	<u>73.47 ± 0.78</u>	**86.88 ± 0.61**	**93.99 ± 0.87**	<u>89.01 ± 0.70</u>	**93.49 ± 0.63**	**95.86 ± 0.47**
GCN (Supervised)	82.80 ± 0.00	72.00 ± 0.00	84.80 ± 0.00	92.42 ± 0.00	86.51 ± 0.00	93.03 ± 0.00	95.65 ± 0.00

Table 3: Performance on cross-domain one-shot scenarios (pseudo zero-shot scenarios). Methods with "*" are reported from [35]. "-AVG", "-MIN", "-MAX" denote the average, min and max performance of our model pre-trained on one of four cross-domain datasets. PT means prompt tuning.

Target datasets	PT	Cora	CiteSeer	PubMed	Photo	Computers
GPPT*	✓	15.37 ± 4.51	23.24 ± 2.94	36.56 ± 5.31	16.19 ± 4.73	19.22 ± 8.71
GraphPrompt*	✓	35.90 ± 7.10	32.76 ± 7.66	43.34 ± 10.66	49.88 ± 8.31	43.03 ± 10.35
GPF*	✓	37.84 ± 11.07	37.61 ± 8.87	46.36 ± 7.48	49.42 ± 7.04	37.00 ± 6.52
GCOPE*	✓	33.38 ± 6.86	35.56 ± 6.81	42.10 ± 8.07	48.52 ± 7.78	40.22 ± 7.82
MDGPT*	✓	<u>42.26 ± 10.18</u>	<u>42.40 ± 9.26</u>	<u>49.82 ± 8.38</u>	<u>64.82 ± 10.53</u>	**49.77 ± 11.00**
TIG-AVG	×	**48.43 ± 1.13**	**45.56 ± 0.58**	**52.80 ± 0.50**	**65.68 ± 1.01**	<u>49.35 ± 0.96</u>
TIG-MIN	×	46.85 ± 8.55	45.05 ± 9.46	52.16 ± 9.91	64.49 ± 9.50	48.34 ± 9.95
TIG-MAX	×	49.95 ± 9.22	46.48 ± 9.14	53.38 ± 11.00	67.29 ± 9.91	50.87 ± 11.09

As shown in Table 3, despite other methods employing carefully designed prompt-tuning strategies and leveraging more training data than TIG (TIG is pretrained using only a single dataset, whereas other methods utilize all four remaining datasets). TIG achieves significant improvements on four out of five datasets and performs comparably to MDGPT on the remaining one. This demonstrates TIG's strong generalization ability in few-shot settings. TIG significantly outperforms all methods except MDGPT. Most of these competing methods rely on arbitrary data pre-processing to unify node feature dimensions. However, in cross-domain few-shot scenarios, this approach struggles when faced with new node features that carry different semantics, leading to accuracy degradation. In contrast, MDGPT employs a well-designed prompt-tuning strategy to align domains, allowing it to adapt to differences between upstream and downstream datasets. TIG, however, inherently encodes cross-domain invariant node features, enabling it to generalize effectively even without additional prompt-tuning.

Cross-Domain Few-Shot Scenarios. The results in Table 4 compare cross-domain trained TIG directly with in-domain trained models in few-shot settings, as competing methods exhibit extremely low performance in cross-domain scenarios. To show the best performance of the compared methods, we report the highest accuracy achieved by the recent outstanding benchmark [44], where each prompt-tuning framework is paired with its most effective pre-trained model. Even under this unfavorable comparison, TIG still surpasses existing methods on two out of three datasets and achieves comparable performance on the PubMed dataset, further validating its strong generalization ability. Additionally, a recurring observation from Tables 1, 3, and 4 is that different training datasets have a minimal impact on model performance. This is because TIM extraction unifies node features into a shared space with consistent semantics, ensuring transfer-invariant node features. As a result, TIG effectively leverages knowledge learned during pre-training, maintaining its strong generalization ability even in cross-domain scenarios. Furthermore, in contrast to Table 1, where many cross-domain results of TIG were superior to in-domain results of TIG, Table 4 shows that TIG generally performs better when trained and tested within the same domain. We hypothesize that when supervision is extremely limited, some randomly selected labeled nodes may be of low quality, meaning their representations lie near class boundaries. In-domain trained models adapt better to these noisy supervision signals, resulting in improved performance.

Table 4: Performances on cross-domain few-shot scenarios. Methods with "*" are reported from [44]. Methods with "-MAX" means report their best performance with different pre-train models.

		3-shot			**5-shot**		
		Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
In-domain	Fine-tuning-MAX*	51.97 ± 2.84	45.08 ± 2.09	65.40 ± 3.00	62.66 ± 3.55	39.54 ± 3.54	**70.91 ± 4.87**
	GPPT-MAX*	43.84 ± 6.11	42.34 ± 8.31	**67.43 ± 2.96**	51.98 ± 3.43	45.77 ± 7.41	66.97 ± 3.70
	AIO-MAX*	48.09 ± 4.83	48.09 ± 8.18	65.79 ± 5.79	30.45 ± 0.19	27.93 ± 10.59	46.16 ± 15.83
Cross-domain	TIG (Cora)	-	**52.04 ± 5.75**	65.92 ± 6.55	-	59.71 ± 3.97	64.60 ± 5.55
	TIG (CiteSeer)	52.99 ± 5.48	-	59.47 ± 6.64	**71.21 ± 3.05**	-	65.74 ± 5.63
	TIG (PubMed)	**56.19 ± 5.89**	51.67 ± 5.71	-	69.13 ± 3.44	**60.02 ± 3.99**	-

Table 5: Ablation study of our feature TIM mining method.

		Cora	**CiteSeer**	**PubMed**	**Photo**	**Computers**	**CS**	**Physics**
Continuous + SVD + GCN (Ours)		85.51 ± 2.09	73.47 ± 0.78	86.88 ± 0.61	93.99 ± 0.87	89.01 ± 0.70	93.49 ± 0.63	95.86 ± 0.47
$\mathcal{G}^{(2)}$ Construction	Discrete	84.26 ± 2.30	73.11 ± 1.43	86.79 ± 0.35	93.41 ± 1.04	88.97 ± 0.73	93.30 ± 0.47	95.80 ± 0.41
	Discrete with threshold	85.00 ± 2.64	72.34 ± 1.40	85.76 ± 0.59	93.59 ± 0.98	88.74 ± 0.77	93.14 ± 0.66	95.85 ± 0.38
	Continuous with threshold	85.29 ± 2.06	71.98 ± 1.54	85.95 ± 0.66	93.31 ± 1.17	88.53 ± 0.64	93.24 ± 0.53	95.82 ± 0.30
Augmentation	PCA	84.78 ± 2.93	68.92 ± 1.69	86.28 ± 0.77	91.29 ± 0.62	89.32 ± 0.59	92.76 ± 0.29	95.78 ± 0.37
	Node2Vec	83.82 ± 2.06	70.96 ± 2.50	83.78 ± 0.60	92.05 ± 1.23	87.70 ± 0.24	91.12 ± 0.92	95.47 ± 0.44
TIM Extractor	Two layers of GCN	84.71 ± 3.10	71.92 ± 1.29	86.67 ± 0.57	93.59 ± 1.27	88.75 ± 0.73	93.22 ± 0.55	95.70 ± 0.45
	GCN + Linear	85.15 ± 2.88	71.98 ± 1.15	86.54 ± 0.80	93.49 ± 1.28	88.88 ± 0.82	93.17 ± 0.60	95.81 ± 0.42

6.2 Model Analysis

To evaluate the impact of different Asp and Ext configurations on TIM extraction, we conducted an ablation study on different TIM mining methods, as shown in Table 5. Specifically, we tested the graph ($\mathcal{G}^{(2)}$) construction method, the dimensional augmentation method, and the implementation of TIM extractor. From Table 5, we derive two key observations. First, the choice of details has a relatively minor impact on model performance. This aligns with the theoretical findings in Section 5, which state that if the methods achieve cross-domain invariance, they can successfully extract TIM. This further substantiates that TIG's success primarily stems from the proposed TIM theory, rather than specific implementation details. Second, among all factors, dimensional augmentation has the most significant impact on model performance. According to Theorem 4.5, this technique converts topological signals into numerical embeddings. However, as mentioned in Discussion 1 of Section 4.1, this extraction process inevitably incurs information loss. Thus, selecting an appropriate augmentation may enhance the extraction of useful information, resulting in improved model performance. Our SVD-based approach effectively captures structural information, contributing to superior results.

6.3 Computing Overhead and Scalability to Large-Scale Graphs

Our method, which expands each feature into a node, may seem intuitively complex. However, this complexity is acceptable for large-scale datasets, and TIG can be easily applied to them. This is mainly due to two reasons: **1) Low feature dimensionality in large-scale datasets.** While large-scale graphs contain a high number of nodes (n), the feature dimensionality (d) is typically limited, i.e., $d \ll n$. For example, the OGB-Product dataset has 2.44 million nodes but only 100-dimensional features. Thus, the preprocessing time for feature nodes remains negligible. **2) Optimization and acceleration strategies.** We used several coding tricks in the implementation. For example, during the construction of $\mathcal{G}^{(2)}$, we design acceleration strategies such as randomized sampling to efficiently approximate inter-feature relationships. Furthermore, the augmented feature embeddings are propagated back to original nodes using batch processing and optimized message-passing schemes, which greatly improves efficiency.

Complexity Analysis. Compared to standard GNNs, TIG introduces additional complexity during the construction and processing of $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$. Specifically, constructing the graphs requires sampling s Asp-Node-Asp paths, which incur a cost of $O(s)$. We then perform SVD on $\mathcal{G}^{(2)}$, which has a complexity of $O(d^3)$, where d denotes the feature dimensionality. These steps are conducted entirely during the preprocessing stage. During training and downstream forward propagation, the model propagates feature embeddings to nodes and performs encoding, which has a computational complexity of $O((d+1)n)$. Thus, the overall additional complexity introduced by TIG can be summarized as:

- Preprocessing: $O(s + d^3)$.
- Encoding: $O((d+1)n)$.

For large-scale datasets (where $n \gg d$ and $n \gg s$), the overhead from d and s is negligible, resulting in an overall complexity of approximately $O(n)$.

7 Conclusion and Future Work

In this paper, we focus on exploring the Transfer-Invariant Metadata (TIM) within general graph node features, providing insights for graph foundation models. We propose a TIM theory to universally extract TIM that shares semantic spaces across different datasets and introduce the Transfer-Invariant Graph foundation model (TIG). TIG demonstrates outstanding performance across various cross-domain pre-training and fine-tuning scenarios, even outperforming other in-domain models in few-shot settings. We hope that the TIM theory proposed in this work will further support the development and design of graph foundation models in the future. Additionally, upon acceptance of this paper, we will release a pre-trained TIG model that can be directly applied to extract universal representations for various general graphs, enabling the development of diverse graph-based downstream tasks.

8 Limitations

The limitations of this work primarily lie in two aspects. First, our study focuses on exploring transfer-invariant metadata within node features, without explicitly investigating the transfer invariance of graph structures. Therefore, our approach is generally effective on homophily graphs, but it may encounter challenges on heterophily graphs. To address this, one potential solution, as Discussion 3 in Section 4, is to leverage prompts to unify the differences in R between homophily and heterophily graphs. Alternatively, as suggested in Section 2, a topology selector could be incorporated as a complementary mechanism. We will explore this in the future. Second, although Assumption 4.3 holds in most cases, and our design of the TIM Extractor effectively mitigates errors introduced by discontinuities in E. In some extreme scenarios, the assumption's constraints on Asp may no longer be valid. In such cases, it would be necessary to redesign the TIM Extractor to ensure proper adaptation and maintain reliable performance across diverse graph datasets.

References

[1] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V. Chawla. 2022. Graph Barlow Twins: A self-supervised representation learning framework for graphs. *Knowl. Based Syst.* 256 (2022), 109631. https://doi.org/10.1016/j.knosys.2022.109631

[2] Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024. LLaGA: Large Language and Graph Assistant. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.* OpenReview.net. https://openreview.net/forum?id=B48Pzc4oKi

[3] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* 295–304.

[4] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2023. Universal Prompt Tuning for Graph Neural Networks. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[5] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2024. Towards Foundation Models for Knowledge Graph Reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=jVEoydFOl9

[6] Dongxiao He, Lianze Shan, Jitao Zhao, Hengrui Zhang, Zhen Wang, and Weixiong Zhang. 2024. Exploitation of a Latent Mechanism in Graph Contrastive Learning: Representation Scattering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems.*

[7] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *CoRR* abs/1611.07308 (2016). arXiv:1611.07308 http://arxiv.org/abs/1611.07308

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[9] Nils Kriege and Petra Mutzel. 2012. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483* (2012).

[10] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024. ZeroG: Investigating Cross-dataset Zero-shot Transferability in Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 1725–1735. doi:10.1145/3637528.3671982

[11] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023. One for All: Towards Training One Graph Model for All Classification Tasks. *CoRR* abs/2310.00149 (2023). arXiv:2310.00149 doi:10.48550/ARXIV.2310.00149

[12] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. 2023. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829* (2023).

[13] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 417–428. doi:10.1145/3543507.3583386

[14] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. [n. d.]. Position: Graph Foundation Models Are Already Here. In *Forty-first International Conference on Machine Learning.*

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[16] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1150–1160. doi:10.1145/3394486.3403168

[17] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[18] Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip Yu. 2025. Riemanngfm: Learning a graph foundation model from riemannian geometry. In *Proceedings of the ACM on Web Conference 2025.* 1154–1165.

[19] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1717–1727. doi:10.1145/3534678.3539249

[20] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 2120–2131. doi:10.1145/3580305.3599256

[21] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2024. All in One: Multi-task Prompting for Graph Neural Networks (Extended Abstract). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024.* ijcai.org, 8460–8465. https://www.ijcai.org/proceedings/2024/942

[22] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. GraphGPT: Graph Instruction Tuning for Large Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 491–500. doi:10.1145/3626772.3657775

[23] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. 2021. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning.*

[24] Raja Vavekanand and Kira Sam. 2024. Llama 3.1: An in-depth analysis of the next-generation large language model.

[25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.* OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[26] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *7th International Conference*

on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net. https://openreview.net/forum?id=rklz9iAcKQ

[27] Sebastian Vlaic, Theresia Conrad, Christian Tokarski-Schnelle, Mika Gustafsson, Uta Dahmen, Reinhard Guthke, and Stefan Schuster. 2018. ModuleDiscoverer: Identification of regulatory modules in protein-protein interaction networks. *Scientific reports* 8, 1 (2018), 433. https://doi.org/10.1038/s41598-017-18370-2

[28] Kai Wang and Siqiang Luo. 2024. Towards Graph Foundation Models: The Perspective of Zero-shot Reasoning on Knowledge Graphs. *CoRR* abs/2410.12609 (2024). arXiv:2410.12609 doi:10.48550/ARXIV.2410.12609

[29] Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. 2025. Multi-Domain Graph Foundation Models: Robust Knowledge Transfer via Topology Alignment. *CoRR* abs/2502.02017 (2025). arXiv:2502.02017 doi:10.48550/ARXIV.2502.02017

[30] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[31] Lianghao Xia and Chao Huang. 2024. Anygraph: Graph foundation model in the wild. (2024).

[32] Lianghao Xia, Ben Kao, and Chao Huang. 2024. Opengraph: Towards open graph foundation models. *arXiv preprint arXiv:2403.01121* (2024).

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

[34] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. *CoRR* abs/2407.10671 (2024). arXiv:2407.10671 doi:10.48550/ARXIV.2407.10671

[35] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. 2024. Text-Free Multi-domain Graph Pre-training: Toward Graph Foundation Models. *CoRR* abs/2405.13934 (2024). arXiv:2405.13934 doi:10.48550/ARXIV.2405.13934

[36] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. 2024. All in One and One for All: A Simple yet Effective Method towards Cross-domain Graph Pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 4443–4454. doi:10.1145/3637528.3671913

[37] Jitao Zhao, Di Jin, Meng Ge, Lianze Shan, Xin Wang, Dongxiao He, and Zhiyong Feng. 2024. FUG: Feature-Universal Graph Contrastive Pre-training for Graphs with Diverse Node Features. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems.*

[38] Jianan Zhao, Hesham Mostafa, Mikhail Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2024. Graphany: A foundation model for node classification on any graph. *arXiv preprint arXiv:2405.20445* (2024).

[39] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[40] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.

[41] Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. 2025. Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025.* 2183–2197.

[42] Yun Zhu, Yaoke Wang, Haizhou Shi, Zhenshuo Zhang, Dian Jiao, and Siliang Tang. 2024. GraphControl: Adding Conditional Control to Universal Graph Pre-trained Models for Graph Domain Transfer Learning. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 539–550. doi:10.1145/3589334.3645439

[43] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. *CoRR* abs/2006.04131 (2020). arXiv:2006.04131 https://arxiv.org/abs/2006.04131

[44] Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. 2024. ProG: A Graph Prompt Learning Benchmark. *CoRR* abs/2406.05346 (2024). arXiv:2406.05346 doi:10.48550/ARXIV.2406.05346

A Supplement of TIM Theory

A.1 The Proof of Theorem 4.4 and The Discussion on Assumption 4.3

Discussion on Assumption 4.3. Assumption 4.3 imposes constraints on how numerical datasets are mathematically modeled and how an object R_i is expressed in terms of its $\text{Asp}(\cdot)$ features. Specifically, it assumes that for each physical property $\text{Asp}_j(\cdot)$, there exists a measurable representation, such as length, width, or temperature. The corresponding physical signal strength encoding is denoted as $\text{Sig}(\cdot)$ which is consistent among common physical properties, while E_j encodes this signal's intensity into a mathematically defined range. For example, if $\text{Asp}_j(\cdot)$ represents temperature, then $\text{Sig}_j(\cdot)$ extracts the temperature of object R_i, while E_j scales this measurement into either the Celsius or Fahrenheit space, depending on the chosen metric system. Assumption 4.3 defines E_j as a matrix, implying that it satisfies linear and continuous transformations, which hold for most continuously measured data.

However, when the data is discontinuous, an additional perturbation term ϵ is introduced under the data after transformation E_j, similar to the idea in Gumbel-Max Trick. In Theorem 4.4, data discontinuity invalidates the result because ϵ_j varies across different $\text{Asp}_j(\cdot)$ and cannot be eliminated. In contrast, Corollary 4.5 remains valid because it models the internal relationships of Asp within the entire feature X, allowing the summation process to eliminate the Gaussian-distributed perturbation ϵ i.e. the expectation of ϵ is 0. Furthermore, in Theorem 4.6, introducing a denoising function $\text{Ext}_{\text{FN}}(\cdot)$ or using discrete-to-continuous fitting methods ensures that the theorem remains valid.

Therefore, Assumption 4.3 holds in most scenarios, and cases where it fails due to the non-continuity of E do not affect the correctness of Corollary 4.5 and Theorem 4.6.

Proof of Theorem 4.4. As mentioned in Assumption 4.3, $x_{ij} = \text{Sig}(v_i, \text{Asp}_j^{\text{Raw}})E_j$, where $\text{Sig}(\cdot)$ is consistent across datasets. For simplicity, we consider $t_{ij} = \text{Sig}(v_i, \text{Asp}_j^{\text{Raw}})$ and the normalization method is Min-Max normalization. Then, we have:

$$\begin{aligned}
\widehat{x}_{ij} &= \text{Min-Max}(x_{ij}) = \text{Min-Max}\big[t_{ij}E_j\big] \\
&= \frac{t_{ij}E_j - \text{Min}(t_{:j}E_j)}{\text{Max}(t_{:j}E_j) - \text{Min}(t_{:j}E_j)} = \frac{\big[t_{ij} - \text{Min}(t_{:j})\big]E_j}{\big[\text{Max}(t_{:j}) - \text{Min}(t_{:j})\big]E_j} \\
&= \big(\big[t_{ij} - \text{Min}(t_{:j})\big]E_j\big)\big(\big[\text{Max}(t_{:j}) - \text{Min}(t_{:j})\big]E_j\big)^{-1} \\
&= \big[t_{ij} - \text{Min}(t_{:j})\big]\big[\text{Max}(t_{:j}) - \text{Min}(t_{:j})\big]^{-1} \\
&= \frac{\big[t_{ij} - \text{Min}(t_{:j})\big]K}{\big[\text{Max}(t_{:j}) - \text{Min}(t_{:j})\big]K} \\
&= \text{Min-Max}(t_{ij}) = \text{Min-Max}\big[\text{Sig}(v_i, \text{Asp}_j^{\text{Raw}})K\big]
\end{aligned} \tag{10}$$

here, we use two mathematical tricks to proof. First, we calculate the pseudo-inverse matrix of E_j. Second, we assume a universal transformation K that converts physical signal strengths into numerical values. In above equation, $\text{Min-Max}(\cdot)$ is consistent across datasets, $\text{Sig}(\cdot)K$ is consistent across datasets. When we consider the $\text{Sig}(\cdot)K$ as a kind of $\text{Asp}(\cdot)$, and $\text{Min-Max}(\cdot)$ as a kind of $\text{Ext}(\cdot)$, $\widehat{X}$ is TIM with respect to $R^1 = \{(v_i, \text{Asp}_j^{\text{Raw}}) | v_i \in R^{\text{Raw}}, \text{Asp}_j^{\text{Raw}} \in \text{Asp}^{\text{Raw}}\}$ in the dataset set $\mathcal{D}_{\text{GG}}$. For other normalization methods, E_j can also be eliminated, thereby transforming the cross-dataset variant in E_j into an invariant $\mathrm{Ext}(\cdot)$.

Table 6: Dataset statistics.

	#Nodes	#Edges	#Features	#Classes
Cora	2,708	4,732	1,433	7
CiteSeer	3,327	5,429	3,703	6
PubMed	19,717	44,338	500	3
Photo	7,650	119,081	745	8
Computers	13,752	245,861	767	10
CS	18,333	81,894	6,805	15
Physics	34,493	247,962	8,415	5

A.2 The Proof of Corollary 4.5.

Consider an object set $R^* = \mathcal{V}^{(k)} \cap \mathcal{A}^{(k)}$. From Theorem 4.4, we know that for each pair $(v_i^{(k)}, \mathrm{Asp}_i^{\mathrm{Raw}(k)})$ within R^*, there exists a TIM, $\widehat{X}^{(k)}$ that captures their relationship. For simplicity, we denote the structured node of $\mathrm{Asp}_i^{\mathrm{Raw}}(k)$ as a_i. Using $\widehat{X}^{(k)}$, we model the connectivity between any two nodes a_i and a_j, where the edge weight w_{a_i,a_j} represents their interaction in R^*. This is formulated as:

$$w_{a_i,a_j} = \mathrm{Agg}\big[\mathrm{Comb}(\widehat{x}_{a_i,v_1}, \widehat{x}_{a_j,v_1}),\mathrm{Comb}(\widehat{x}_{a_i,v_2}, \widehat{x}_{a_j,v_2}), \ldots,\mathrm{Comb}(\widehat{x}_{a_i,v_n}, \widehat{x}_{a_j,v_n})\big] \tag{11}$$

where $\mathrm{Comb}(\cdot)$ is a function that combines $\widehat{x}_{a_i,v_k}$ and $\widehat{x}_{a_j,v_k}$ (e.g., multiplication), and $\mathrm{Agg}(\cdot)$ aggregates multiple Asp-Node-Asp relationships (e.g., averaging). When both $\mathrm{Agg}(\cdot)$ and $\mathrm{Comb}(\cdot)$ are consistent across datasets, Corollary 4.2 ensures that w_{a_i,a_j} remains a TIM, meaning the constructed graph structure is a TIM. The resulting graph $G^{(\mathrm{New})}$ consists solely of nodes from $\mathcal{A}^{(k)}$, capturing the physical property of "relation" among all objects in $\mathcal{A}^{(k)}$. Next, we employ a domain-invariant structural embedding method, $\mathrm{TopoEmb}(\cdot)$, as follows:

$$X_{\mathrm{Asp}}^{(k)} = \mathrm{TopoEmb}(G^{(\mathrm{New})}) \tag{12}$$

According to Corollary 4.2, $X_{\mathrm{Asp}}^{(k)}$ is a TIM. Therefore, in the $\mathcal{D}_{\mathrm{GG}}$, $X_{\mathrm{Asp}}^{(k)}$) is a TIM with respect to $R^2 = \{\mathcal{A}^{(k)} \mid \mathrm{DS}^{(k)} \in \mathcal{D}_{\mathrm{GG}}\}$.

A.3 The Proof of Theorem 4.6.

For a graph dataset $\mathrm{DS}^{(k)} \in \mathcal{D}_{\mathrm{GG}}$ containing $\mathcal{V}$ and $\mathcal{A}$, according to Theorem 4.4 and Corollary 4.5, we have the attribute TIM of $\mathcal{A}$ $X_{\mathrm{Asp}}^{(k)}$ and the relation TIM $\widehat{X}^{(k)}$, which captures the relationship between a node v_i and $Asp_j^{\mathrm{Raw(k)}}$. Then, we have:

$$\begin{aligned} X_{\mathrm{FN}i}^{(k)} &= \mathrm{Ext}_{\mathrm{FN}}(\widehat{X}_{i,:}^{(k)}, X_{\mathrm{Asp}}^{(k)}) \\ &= \mathrm{Agg}(\mathrm{Enc}(\widehat{X}_{i,1}^{(k)}, X_{\mathrm{Asp}\ 1}^{(k)}), \mathrm{Enc}(\widehat{X}_{i,2}^{(k)}, X_{\mathrm{Asp}\ 2}^{(k)}), \\ &\quad \ldots, \mathrm{Enc}(\widehat{X}_{i,d}^{(k)}, X_{\mathrm{Asp}\ d^{(k)}}^{(k)})) \end{aligned} \tag{13}$$

where $\mathrm{Agg}(\cdot)$ is an aggregation function (e.g., averaging), and $\mathrm{Enc}(\cdot)$ represents an encoding function. If both $\mathrm{Agg}(\cdot)$ and $\mathrm{Enc}(\cdot)$ are domain-invariant, then $\mathrm{Ext}_{\mathrm{FN}}(\cdot)$ is also domain-invariant. According to Corollary 4.2, X_{FN}) is a TIM with respect to $R^3 = \{(v_i, \mathcal{A}^{(k)}) \mid v_i \in \mathrm{DS}^{(k)}, \mathrm{DS}^{(k)} \in \mathcal{D}_{\mathrm{GG}}\}$ in $\mathcal{D}_{\mathrm{GG}}$.

B Experimental Settings

B.1 Datasets

We follow existing studies [37, 44] and select seven widely used general graph datasets. These datasets are divided into three groups: (1) Citation datasets including Cora, CiteSeer, and PubMed; (2) Copurchase datasets including Photo and Computers; (3) Co-author datasets including CS and Physics. The statistics of these datasets are provided in Table 6.

B.2 Compared Methods

In each scenario, we compared TIG with outstanding works, which include: (a) Universal graph pretraining models: OFA [11], GraphControl [42], All-in-One [20], FUG [37], GCOPE [36], ULTRA (3g) [5], SCORE [28], MDGFM [29], LLaGA [2], ZeroG [10], OpenGraph [32], GraphGPT [22], GraphCLIP [41], and MDGPT [35]. (b) Large Language Models: Qwen2 [34] and LLaMA3.1 [24]. (c) Graph self-supervised learning methods: GRACE [43], BGRL [23], GBT [1], DGI [26], and SGRL [6]. (d) Prompt-based methods: GPPT [19], GraphPrompt [13], GPF [4], and GPF-plus [4]. (e) Other meaningful baselines, including Raw features, PCA (node feature processed by PCA), SVD (node feature processed by SVD), GCN [8].

B.3 Implementation Details

In scenarios (1) and (2), we followed the setup from previous work [37], where the model was first self-supervised on the training dataset. Then, a simple downstream L2 classifier was finetuned with the pretrained model fixed. We split the training, validation, and test sets in a 10%:10%:80% ratio and ran the experiment 20 times to report the mean and standard deviation. In scenarios (3) and (4), we also first self-supervised the TIG model, keeping it fixed in downstream settings. For each class, n samples (where $n = 1$ in scenarios (3) and $n = 3$ or $n = 5$ in scenarios (4) were randomly chosen and labeled. In scenario (3), for TIG, we used a single sample per class as the prototype vector without any prompts or finetuning, and predicted labels were determined based on the highest similarity between the node and prototype vectors. In scenario (4), we used 3 or 5 samples to finetune a downstream L2 classifier. In all few-shot scenarios, we run the experiments 500 times to report the average and standard deviation. We employed a two-layer GCN as the graph encoder, a one-layer GCN as the TIM extractor, and PReLU as the activator.

All of our experiments were conducted on a server equipped with an Intel Core i5-12400 CPU, 32GB of RAM, an NVIDIA RTX 3090 graphics card, and 24GB of video memory.

Hyper-Parameter Settings. For Table 2, we use different parameters among different datasets, because in this scenario, all models are rebuilt and re-trained in each dataset. For Tables 1, 3 and 4, we use the same hyper-parameters across different datasets in one scenario. The hyper-parameters of TIG are summarized in Table 7. Here, λ is the weight for the positive part of the loss in the Scattering loss. Dim #1 represents the number of dimensions used

Table 7: Hyper-parameters settings of TIG.

Scenarios	Models	**Learning rate**	**Weight decay**	**#Epochs**	**Dim #1**	**Dim #2**	**Dim #3**	λ
Scenario (1)	TIG	0.0001	1e-3	300	256	512	1024	0.15
Scenario (2)	TIG (Cora)	0.0001	1e-3	300	128	1024	1024	0.1
	TIG (CiteSeer)	0.0001	1e-3	300	128	512	2048	0.1
	TIG (PubMed)	0.0001	1e-3	300	128	1024	2048	0.1
	TIG (Photo)	0.0001	1e-3	300	256	1024	2048	0.15
	TIG (Computers)	0.0001	1e-3	300	256	512	2048	0.15
	TIG (CS)	0.0001	1e-3	200	64	512	1024	0.2
	TIG (Physics)	0.0001	1e-3	200	256	1024	2048	0.15
Scenarios (3) and (4)	TIG (1-shot)	0.0001	5e-4	200	16	128	256	0.1
	TIG (3-shot)	0.0001	5e-4	200	16	16	256	0.1
	TIG (5-shot)	0.0001	5e-4	200	32	128	256	0.1
Graph classification	TIG	0.00001	5e-4	300	256	1024	1024	0.2

Table 8: Cross-domain graph classification. Our test settings followed SCORE [28]. Methods with "*" are reported from SCORE.

Setting	Method	IMDB-B		COLLAB		COX2	
		Acc	F1	Acc	F1	Acc	F1
One-shot	GPPT* [19]	50.15 ± 0.75	44.16 ± 6.70	47.18 ± 5.93	42.87 ± 7.70	**78.23 ± 1.38**	44.68 ± 1.17
	AIO* [21]	60.07 ± 4.81	56.88 ± 0.80	51.66 ± 0.26	47.78 ± 0.10	76.14 ± 5.51	49.62 ± 10.42
	GPF-plus* [4]	57.93 ± 1.62	55.55 ± 2.03	47.24 ± 0.29	41.24 ± 0.31	33.78 ± 1.52	30.90 ± 11.56
Zero-shot	ULTRA (3g)* [5]	49.25 ± 0.00	38.87 ± 0.00	64.53 ± 0.00	55.36 ± 0.00	77.75 ± 0.00	43.74 ± 0.00
	SCORE* [28]	61.83 ± 1.60	60.91 ± 2.18	65.45 ± 1.05	57.71 ± 1.82	78.08 ± 1.33	49.24 ± 3.55
Zero-shot	TIG (IMDB-B)	-	-	**69.69 ± 0.71**	**68.31 ± 0.74**	56.37 ± 3.99	51.00 ± 3.87
	TIG (COLLAB)	66.40 ± 3.54	66.24 ± 3.60	-	-	59.07 ± 4.82	**53.91 ± 4.51**
	TIG (COX2)	**69.05 ± 1.90**	**68.85 ± 1.96**	69.32 ± 0.81	68.12 ± 0.73	-	-

for dimensional augmentation, Dim #2 represents the output dimensionality of the TIM extractor, and Dim #3 denotes the number of dimensions in the representations.

C Supplementary Experiments.

C.1 Additional Experiments for Graph-Level Task

We further validate the performance of TIG on graph-level tasks. Although universality across various tasks is not the primary scope of this paper, we report TIG's performance on graph classification tasks as shown in Table 8, as TIG is a graph foundation model. We selected three commonly used graph-level datasets: 1) IMDB-BINARY, which is a movie collaboration dataset [33], 2) COLLAB, a scientific collaboration dataset [33], and 3) COX2, a chemical dataset [9, 17]. It is worth noting that IMDB-BINARY and COLLAB do not provide raw node features; therefore, we generated their features via one-hot encoding of node degrees.

For our experimental setup, similar to node classification, we first generate node representations using TIG and then obtain graph representations through pooling. During testing, we followed existing work [28] by utilizing class prototype vectors generated from the training set for zero-shot scenario. The hyper-parameter settings of TIG are in Table 7.

The experimental results are presented in Table 8. It is worth noting that the methods we compare are state-of-the-art techniques specifically designed for graph classification, many of which incorporate additional optimization strategies tailored to the task. However, TIG does not include such specific optimizations. Even so, TIG still achieved the best and second-best results in five out of six settings, further validating the model's strong generalization capabilities. This also demonstrates that focusing solely on extracting Transfer-Invariant Metadata can significantly enhance a graph foundational model's performance.

C.2 Additional Experiments on Molecular Graphs

To comprehensively assess TIG's cross-domain generalization, we evaluated its performance when transferring from citation datasets to molecular datasets.

For our experimental setup, we followed excellent work GPF [4], which focuses on domain-specific pretraining and fine-tuning, and compared our cross-domain results with those reported for within-domain graph classification which favors their methods. Besides, to ensure a fair comparison, the compared GPF uses GCL as the backbone—identical to our proxy task—and reports both its direct fine-tuning results (which are consistent with our setup) and its results with GPF-based promoting.

Table 9: Fifty-shot test ROC-AUC (%) performance on molecular prediction benchmarks. The experimental settings follow those used in GPF[4]. The results for GCL-FT, GCL-GPF, and GCL-GPF-plus are taken directly from the GPF paper. FT refers to directly fine-tuning a downstream classifier, while GPF and GPF-plus refer to using the promotion algorithms proposed in the GPF framework. TIG (Cora) means a TIG model pretrained on Cora dataset.

Method	BBBP	Tox21	ToxCast	ClinTox	MUV	HIV
GCL-FT (ZINC15)	54.40 ± 2.87	48.35 ± 1.67	50.29 ± 0.19	54.05 ± 4.16	46.73 ± 1.88	60.05 ± 3.80
GCL-GPF (ZINC15)	53.87 ± 2.17	50.58 ± 0.49	52.64 ± 0.50	64.44 ± 4.64	47.22 ± 3.55	**64.86 ± 1.29**
GCL-GPF-plus (ZINC15)	55.89 ± 1.58	50.14 ± 1.09	53.25 ± 0.95	**65.22 ± 4.51**	47.88 ± 1.77	63.99 ± 1.60
TIG (Cora)	**63.70 ± 00.18**	**54.33 ± 0.06**	**53.29 ± 0.16**	54.39 ± 4.49	**50.60 ± 7.48**	61.16 ± 0.67

It is worth noting that the above experimental settings heavily favor the baseline methods compared. To be specific, they perform pretraining on the molecular dataset ZINC15, which is within-domain, whereas our model is pretrained on the citation dataset Cora, which is cross-domain. Additionally, their model framework is more tailored for graph classification, whereas our approach consistently employs a two-layer GCN, identical to the one used for node classification.

The results are shown in Table 9. As observed, TIG achieves competitive performance on molecular datasets, even in cross-domain settings. Please note that TIG focuses purely on feature alignment, without any specific mechanisms for transferring domain knowledge. Despite this, its performance remains strong.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009