TouchUp-G: Improving Feature Representation through Graph-Centric Finetuning

Jing Zhu
University of Michigan, Ann Arbor
jingzhuu@umich.edu

Xiang Song
Amazon
xiangsx@amazon.com

Vassilis N. Ioannidis
Amazon
ivasilei@amazon.com

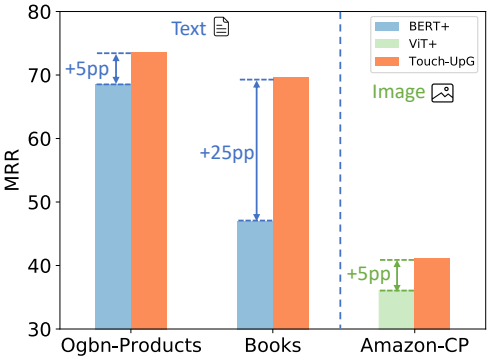
Danai Koutra
University of Michigan, Ann Arbor
Amazon
dkoutra@umich.edu

Christos Faloutsos
Carnegie Mellon University
Amazon
christos@cs.cmu.edu

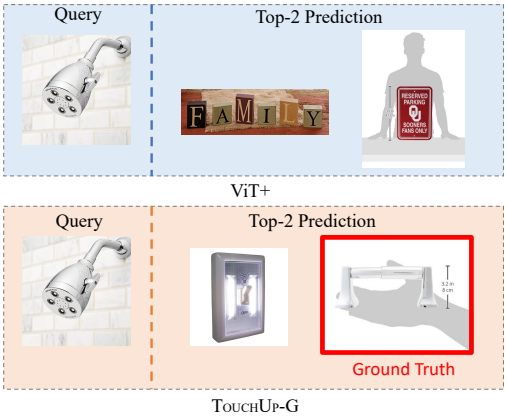
ABSTRACT

How can we improve node features obtained from Pretrained Models (PMs) for downstream graph tasks? Graph Neural Networks (GNNs) have demonstrated promising results in various graph learning tasks, including node classification and link prediction. Despite their remarkable success in high-impact applications, we have observed that for feature-rich graphs, it is a common practice to directly employ a PM for feature generation in GNNs without incorporating any domain adaptation techniques. However, this practice is suboptimal because the node features extracted from PM are graph-agnostic and it prevents fully utilize the potential correlations between graph structures and node features. So how can we improve node features obtained from a PM for downstream graph tasks? We found that the best way is to do graph-centric finetuning on the PM.

In this paper, we present TouchUp-G; a simple Touch-Up enhancement technique to improve Graphs' node features obtained from PMs via graph-centric pretraining. TouchUp-G has the following advantages: (a) **General**, can be applied to any downstream graph tasks; (b) **Multi-modal**, can improve raw features that come from any modality (e.g. images, texts); (c) **Principled**, we propose a novel metric: feature homophily to quantify the potential correlations between graph structures and node features; (d) **Effective**, outperforms baselines on 4 real datasets across various tasks and modalities, with up to 2× performance improvement on MRR.



(a) Quantative Result



(b) Qualitative Result

Figure 1: TouchUp-G wins: Compared with features obtained directly from pretrained models (BERT [9] or ViT [11]), (a) TouchUp-G improves the quantative performance by more than 25% across datasets and modalities. (b) shows a co-purchasing example that exists in Amazon-CP in the test split. TouchUp-G correctly predicts the ground truth in top-2 predictions while ViT+ fails.

1 INTRODUCTION

How can we improve node features obtained from a Pretrained Model (PM) for downstream graph tasks? Graphs or networks are foundational representations for relational structures and their analysis is useful in many scientific and industrial applications. Various tasks, including recommendation, question answering, and knowledge graph completion, can be cast graph tasks. Graph Neural Networks (GNNs) operate in a message passing mechanism, where the nodes pass their feature representations as messages to their neighbors and update their neighbors feature representations accordingly [12, 14, 25, 39, 45]. The feature representation for each node is initialized as the node's initial features. With the advantage of combining the topology structure and node features, GNNs have become the state of the art practice for most real-world graph tasks

Accepted to Second Workshop on Knowledge Augmented Methods for Natural Language Processing, in conjunction with KDD 2023.
Work done while doing an internship at Amazon

Figure 2: Why pretrained features may fail: We show a subgraph of Amazon Co-purchasing graph (Amazon-CP). Products have completely different visual features, but they are often bought together.

such as node classification. Besides the conventional node classification task, link prediction has also gained widespread popularity as a graph pretraining technique to tackle the challenge of insufficient labels for self-supervised learning on graphs [20, 21, 57].

Pretrained Models (PMs), like BERT, GPT, ViT [3, 9, 11] have shown remarkable performance on many natural language processing and computer vision tasks, such as question answering and image classification, and have became the foundations of modern ML systems. As a result, for feature-rich graphs (each node in the graph has raw features such as images, texts, audios), it is a common practice to directly employ a pretrained model to generate node features and and then directly adapt the generated features into GNNs, without any domain adaptation [5, 17, 18, 22, 55]. As a result, the features generated from PMs are completely unaware of the graph structures and yields poor performance when doing feature propagation on these node features using GNNs. As shown in Fig. 2, while these products, have completely different visual representations, they are often bought together according to user co-purchasing history. Directly using the image features from ViT will give completely different feature embeddings for each product and thus making GNNs fail to predict the products are likely to be co-purchased together. This is a widespread issue that persists whenever a PM is leveraged to generate the node features and the pretraining objective is irrelevant with the graph's structure information. So how can we improve node features obtained from a PM for the downstream graph tasks?

We found that the best approach is to do graph-centric finetuning on the PM. By doing graph-centric finetuning on the PM, the PM gets the structure information needed to extract meaningful topological aware features that can benefit the downstream graph tasks, and as a result, let GNNs better utilize the correlation between node features and graph structures. To quantify the correlation between the extracted node features and graph structure, we propose Feature Homophily, a extended metric from assortativity [34, 35] to

Table 1: TouchUp-G matches all specs: Comparing finetuning frameworks. TouchUpG is General: can be applied on any graph datasets and Multi-modal: can be used to features from any modality, Principled, the .

Property \ Method	*AdsGNN [28]*	*GIANT [5]*	*BERT+ [9]*	**TouchUp-G**
General			✓	✓
Multi-modal				✓
Principled		✓		✓
Effective	✓	✓	?	✓

assess the necessity of applying graph-centric finetuning on the PM. To improve node features obtained from a PM for graph tasks, we present TouchUp-G; a simple TouchUp enhancement technique that improves Graph's node features from any Pretrained Models. TouchUp-G is lightweight, and easily adaptable to a variety of pretrained models across multiple domains. The node features obtained from TouchUp-G exhibit a strong correlation with the graph structure and can achieve up to 2× performance improvement for GNNs on downstream graph tasks. A summary of our results are shown in Fig. 1. Our code will be made public upon acceptance of the paper.

Our contributions are summarized as follows:

- **General**: TouchUp-G can be applied to a variety of graph tasks, including node classification as well as link prediction.
- **Multi-modal**: TouchUp-G can be applied any pretrained models from any modality, e.g. texts, images etc.
- **Principled**: We propose a novel metric (see Eq. 2, section 4.1): feature homphily to measure the correlation between node features and graph structure when features are vectors instead of scalars/categorical.
- **Effective**: TouchUp-G outperforms baselines on 4 real datasets, with up to 2× performance improvement across various tasks, metrics and modalities.

2 RELATED WORK

Next we discuss related work. Notice that none of the methods below, satisfy all the specifications (see Table 1).

Pretrained Transformers as Feature Embeddings. Pretrained language models aim to learn general language representations from large-scale corpora, have been used to obtain a contextualized text representation for graphs with rich text information [9, 31, 32]. Yasunaga *et al.* pretrains LMs by leveraging links between documents and show that the joint pretraining of masked language modeling and document relation prediction improves the language representation of documents [51]. The breakthroughs of transformers [44] has also sparked great interest in the computer vision community. Vision Transformers (ViTs) have been proposed to learn a general visual representation for images [11, 27, 33, 38]. We refer to the survey for more details [24]. The typical approach for graph learning with rich text or image features is to adopt a

"cascaded architecture" where the feature information of each node is first extracted by a PM, and the extracted features are used as node feature embeddings for GNNs, without further pretraining on the graph domain. [5, 21–23, 28, 57]. Gururangan *et al.* shows the importance of doing domain-adaptative pretraining and language domains and the same applies to the graph domain [13]. It is essential to do domain adaptation on the features extracted from PMs for downstream graph tasks. Ioannidis *et al.* proposes a multi-step finetuning framework that can jointly train LMs and GNNs effectively and efficiently [21]. GIANT uses a neighborhood prediction objective to pretrain XR-Transformers with topology information for node classification on text-rich graphs [5]. GLEM proposes an alternating training framework that fuses graph structure and language learning with a variational Expectation Maximization framework [55].To the best of our knowledge, we are the first to propose finetuning on pretrained visual representations for graphs. We are also the first to propose finetuning technique which can be applied to any pretrained models from any modality, e.g. texts, images, and a variety of graph tasks, including node classification and link prediction.

Link Prediction using GNNs. Link prediction is the task of inferring missing links from an observed network. Techniques to solve this task range from heuristics–e.g., predicting links based on the number of common neighbors between a pair of nodes–to graph neural network (GNN) models , which rely on message passing and leverage both the graph structure and node features [30]. Methods that use GNNs for link prediction mainly fall into two categories: Graph Autoencoder (GAE)-based methods and enclosing subgraph-based methods. GAE-based methods use GNNs as the encoder of nodes, and edges are decoded by their nodes' encoding vectors using score functions [7, 26, 43]. For enclosing subgraph-based methods including SEAL [52, 54], IGMC [53], GraIL [42], NBFNet [59] first extract an enclosing subgraph for the target edge, apply GNNs to encode the node representations of nodes in enclosing subgraph, and then aggregate the node representations by pooling methods. The learned subgraph features are fed into a classifier to predict the existence of the target edge. In this work, we focus on link prediction as the main downstream graph task because link prediction gives meaningful embeddings for self-supervised learning on graph [18, 20] and we mainly focus on GAE-based approaches since they are more scalable and typically used to deal with graphs with millions or even billions of nodes.

Homophily and Heterophily in GNNs Heterophily has been an area of raising interest for GNNs, as conventional GNN designs are ineffective when dealing with heterophily [16, 58]. A variety of methods have been proposed to solve the case of heterophily on node classification. We refer to the survey for more details [56]. However, existing measures of homophily focus on label homophily, which quantifies the variation of node labels across edges. In this work, our focus is on feature homophily, which aims to measure the similarity of node features along connected edges. In [49, 50], the authors showed that GNNs penalizes deviations between the embeddings of two nodes sharing an edge and as a result, GNNs implicitly assumes feature homophily between connected edges. Scalar assortativity is first proposed to quantify the similarity of scalar or categorical features over connected edges [34, 35]. But the traditional notion of assortativity is not application to vectorized features obtained from PMs.ß To the best of our knowledge, our work is the first to introduce measures of feature homophily for vectorized features. Additionally, we pioneer in using feature homophily to quantify the correlation between node features and graph structures. The most closely related metric to our feature homophily measure is the feature smoothness score, which quantifies the propagation of features in the network [16]. However, feature smootheness score fails to be used as a measure of quantifying if graph-centric finetuning is needed because feature smoothness score (1) is unbounded in magnitude, making it unsuitable for comparing graphs of varying sizes, and (2) fails to differentiate the cases of negative or zero correlation of node features and graph structure.

Table 2: Major symbols and their definitions.

Symbols	Definitions
$\mathbf{G}$	Graph
$\mathbf{A}$	Adjacency Matrix
$\mathbf{T}$	Pretrained Model used to generate node feature
$\mathbf{S}$	set of all raw node features (e.g. texts, images)
$\mathbf{X}$	node feature extracted the pretrained model $\mathbf{T}$
h	Feature Homophily score

3 PRELIMINARIES

In this section, we formally define key notions that we use throughout the paper, as well as the problem that we seek to solve. The major symbols we use is defined in Tab. 2.

3.1 Definitions

Graphs.We consider a **graph** $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{S})$, where $\mathbf{V}$ is the set of vertices, $\mathbf{E}$ is the set of edges. Denote $\mathbf{A} \in \mathbb{R}^{|\mathbf{V}|\times|\mathbf{V}|}$ as the adjacency matrix for $\mathbf{G}$.

Node Feature X. We consider the case where $\mathbf{S}$ is the set of raw node features, e.g. raw text, raw images etc., $\mathbf{T}$ is a pretained model from any modalities, then we have $\mathbf{X} = \mathbf{T}(\mathbf{S})$. $\mathbf{X} \in \mathbb{R}^{|\mathbf{V}|\times d}$ is extracted d-dimensional node feature embedding from the pretrained model T, that we use during GNN training.

Link Prediction. Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{S})$, the link prediction task aims to determine whether there will be a link e_{ij} between a pair of nodes i and j, where $i, j \in \mathbf{V}$ and $e_{ij} \notin \mathbf{E}$ based on the extracted node features $\mathbf{X}$ and graph structure $\mathbf{A}$.

Node Classification. Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{S})$, the node classification task aims to determine the node label N based on the extracted node features $\mathbf{X}$ and the graph structure $\mathbf{A}$.

Graph Neural Networks (GNNs). GNN models utilize a neighborhood aggregation scheme to learn a representation vector h_v for each node v. In general, the node representation of node v can be formulated as a k-round neighborhood aggregation schema: $h_v^{(k)} = \text{COMBINE}^{(k)}(\{h_v^{(k-1)}, \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in N_k(v)\})\})$, where AGGREGATE(.) is typically mean or max pooling, and COMBINE(.) can be a sum/concatenation/attention on nodes' ego- and neighbor-embeddings. And $h_u^{(0)} = x_u$ is node u's feature from $\mathbf{T}$.

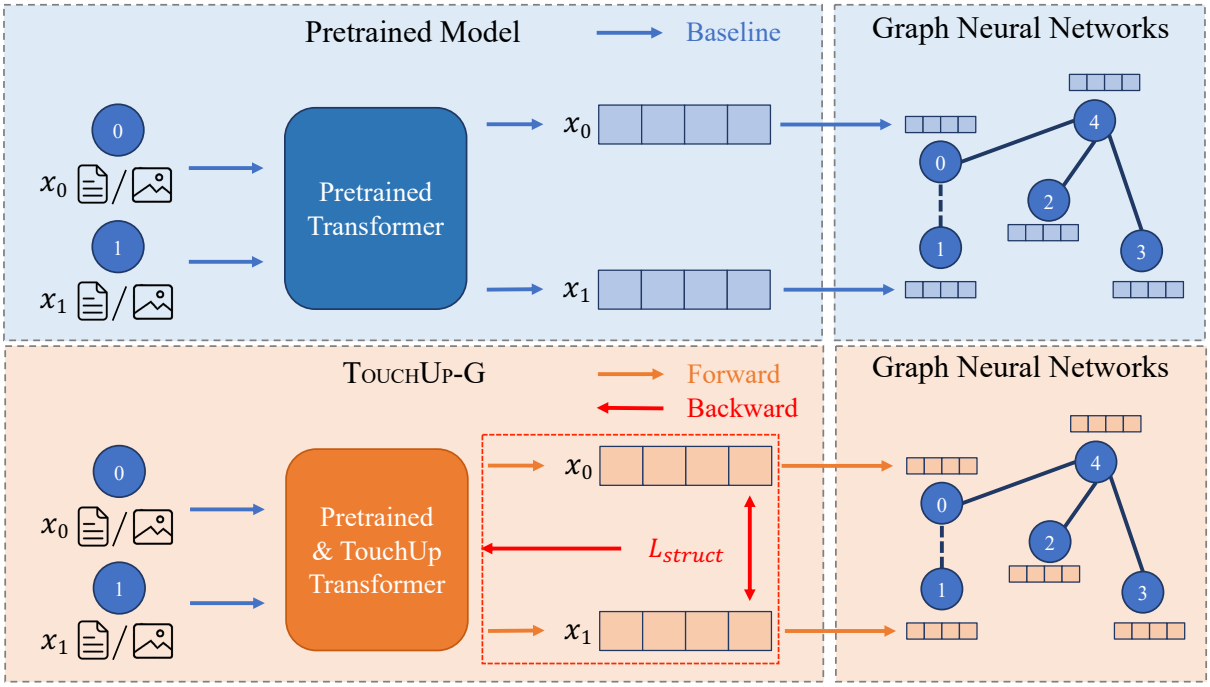


Figure 3: Overview of TouchUp-G. Typically, researchers use a pretrained transformer [9, 11] to extract features from raw text or images of each node, and train GNNs (blue). We propose TouchUp-G (orange), which apply graph-centric finetuning on the pretrained transformer using a structure loss L_{struct}. The features from the touchup transformer incoporates structure information, which boosts the performance of GNNs on downstream graph tasks

3.2 Problem Statement

We seek to finetune the pretrained model $\mathbf{T}$ by minimize the feature embedding distance between connected nodes and maximize the feature embedding distance between disconnected nodes (see Eq 1). Mathematically speaking,

- **Given** an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$, its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathbf{V}|\times|\mathbf{V}|}$ contains both the observed (training edge) and unobserved link(validation and test edge) in the graph $\mathbf{G}$, a pretrained model $\mathbf{T}$ and $\mathbf{X}' = \mathbf{T}(\mathbf{X})$.
- The set $\mathbf{S}$ of raw node features (e.g. texts, images) for all nodes $n \in \mathbf{V}$.
- A pretrained model $\mathbf{T}$ that transforms raw node features to node feature embeddings, $\mathbf{X} = \mathbf{T}(\mathbf{S})$, $\mathbf{X} \in \mathbb{R}^{|\mathbf{V}|\times d}$.
- **Find** the finetuned model $\mathbf{T}'$ such that

$$\mathrm{Tr}(\mathbf{X}'^{T}\mathbf{A}\mathbf{X}') - \mathrm{Tr}(\mathbf{X}'^{T}(\mathbb{1} - \mathbf{A})\mathbf{X}') \tag{1}$$

is minimized for a hold-out set (validation and test edges), where $\mathbf{X}' = \mathbf{T}'(\mathbf{S})$, $\mathbb{1} \in \mathbb{R}^{|\mathbf{V}|\times|\mathbf{V}|}$ is a matrix of all 1s. Note that the edges in the hold-out set is unobserved and are not used to train the model in the proposed methods.

4 PROPOSED METHOD

Next, we detail TouchUp-G, the first pretraining framework that enhances the node features by graph-centric multi-modal finetuning on the pretrained models from any modality (e.g. vision, language etc.). An overview of our method is shown in Fig. 3. We first propose feature homophily score to measure if TouchUp-G is needed. If finetuning is needed, the nodes' raw features are first passed through the pretrained transformer to get the extracted feature embeddings. The extracted feature embedding are then minimized using a structure-aware loss function to finetune the pretained transformers. The node embeddings from the finetuned transformer are used as features in the graph neural network for downstream graph tasks.

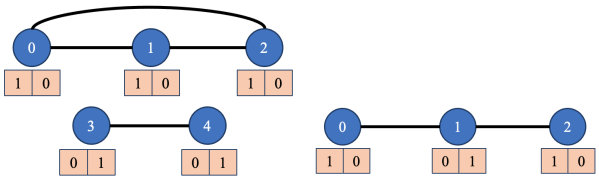
4.1 Proposed Homophily Measure

We first propose Feature Homophily to quantify the correlation between features and structures for any graph with features, and decide if TouchUp-G is needed. The Feature Homophily is an extended vectorized version of scalar assortativity, a matrix that quantifies the similarity of scalar features over connected edges.

Feature Homophily. In this work, we focus on using feature homophily h to characterize the correlation between node features and graph structure, and use it to as a threshold to decide whether feature enhancement is needed using TouchUp-G.

Definition 1. The feature homophily ratio h is defined as follows.

$$h = \frac{\sum_{ij\in E}(x_i - \bar{x}) \cdot (x_j - \bar{x})}{\sqrt{\sum_{ij\in E}(x_i - \bar{x}) \cdot (x_i - \bar{x})} \cdot \sqrt{\sum_{ij\in E}(x_j - \bar{x}) \cdot (x_j - \bar{x})}} \tag{2}$$



(a) h = 1 (b) h = -1

Figure 4: Example graphs that exhibits strong positive/negative correlation between features and structure. For example, for figure(a), every node over the connected edge has the exact same feature and for figure (b), every node over the connected edge has the exact opposite feature.

where $\bar{x} = \frac{\sum_{ij \in E}(x_i + x_j)}{2|E|}$. are the mean of node features over edges. For each edges $e_{ij} \in E$, denote the first node i as the head node and the second node j as the tail node. The feature homophily score h is essentially the pearson correlation between the set of all head node features x_i and the set of all tail node features x_j. As a result, the feature homophily h score quantifies the correlation between node features and graph structures, and h is bounded in the range $[-1, 1]$. In real-world, most graphs exhibit feature homophily $h > 0$ and we leave the case of $h < 0$ for future discussion. In Fig. 4, we give two example graphs that exhibits strong feature homophily/heterophily.

Feature Homophily vs. Label Homophily. Label homophily captures the different tendency between each pair of classes. While label homophily is connected with feature homophily, they are still different from each other in two ways. First, graphs that exhibit strong label homophily may not exhibit strong feature homophily. For example, Books has a label homophily of 0.654 , but its feature homophily score is only 0.194 [40]. Second, label homophily cannot be calculated for all graph tasks. For example, for graphs designed for link prediction, nodes do not have class labels, and thus label homophily score cannot be calculated.

Comparison wrt. Feature Smoothness. Feature smoothness score measures how features propagate throughout the network, which is defined as follows:

$$\lambda_f = \frac{\left\| \sum_{v \in V} \left(\sum_{v' \in N(v)} (x_v - x_{v'}) \right) \right\|_1}{|E|d} \tag{3}$$

where $\lambda_f \in [0, +\infty)$. Here a smaller λ_f indicates the feature vectors x_v and $x_{v'}$ are more likely to be similar for two connected nodes v and v' in the graph.

While similar to Feature Homophily, λ_f also captures the similarity of features for connected edges in the graphs, the magnitude of the feature smoothness score is unbounded and is directly related to the number of edges that a graph has. As a result, it cannot be compared against graphs of different sizes cannot be used to decide if finetuning is needed.

4.2 TouchUp-G

Since the feature homophily score is a pearson correlation between the feature distributions over connected edges and essentially quantifying the correlation between node features and graph features, we use the feature homophily score h as an indicator of whether finetuning is needed. According to [35], a scalar assortativity score greater than 0.5 is regarded as strong. Thus, we set the finetuning threshold to be 0.5. When $h > 0.5$, the graph exhibits strong feature homphily and thus we don't employ TouchUp-G. When $0 < h < 0.5$, the graph exhibits weak feature homophily, and thus we enhance the feature embeddings by using TouchUp-G. In Fig. 3, we show a specific case of how the TouchUp-G finetuning is done for a training edge $e_{0,1}$.

For each training edge $e_{u,v} \in \mathbf{E}$, the original feature of node u,v is x_u, x_v respectively. We first randomly sample a negative node m, where $e_{u,m} \notin \mathbf{E}$. $x'_u = \max(\mathbf{T}(x_u), 0), x'_v = \max(\mathbf{T}(x_v), 0), x'_m = \max(\mathbf{T}(x_m), 0)$ are the pretrained feature embedding obtained from a pretrained model $\mathbf{T}$ for node u, v, m respectively. We then finetune the pretrained transformer $\mathbf{T}$ using a binary cross entropy loss.

$$L_{\text{struct}} = -\frac{1}{|E|} \sum_{\substack{(u,v) \in E \\ (u,m) \notin E}} (\log(x'_u \cdot x'_v) + \log(1 - x'_u \cdot x'_m)) \tag{4}$$

Similar to knowledge graph embedding objectives such as DistMult [4], this structure-aware loss can be seen as a way to learn the graph's topological information in the nodes' representation through gradient descent. As a result, L_{struct} fuses graph's structure information with the raw features together. Note that this structure-aware loss L_{struct} can be used to any pretrained models from any modalities. No matter it is Language models, Vision Transformers or Convolutional Neural Networks for computer vision and audio processing. As long as the pretrained model generates an embedding for each node, it can be finetuned using the structure-aware objective to enhance its feature embeddings with graph's topological information. In practice, we use gradient clipping and early stopping to prevent overfitting on the training edges [29, 41].

Relation with Feature Homophily. In the optimal case, $L_{\text{struct}} = 0$, resulting in $x'_u \cdot x'_v = 1$ for all $(u, v) \in E$. And this indicates strong positive correlation between x'_u and x'_v. Since the feature homophily ratio h is essentially a pearson correlation of node features between the set of head entities u versus the set of tail entities v. As a result, when the pretrained model $\mathbf{T}$ learned a strong correlation between x'_u and x'_v, it would result in a Feature Homophily close to 1.

5 EXPERIMENTS

We aim to address the following research questions (RQ) through experiments.

- RQ1 - **Effective**: How accurate is TouchUp-G?
- RQ2 - **Multi-modal**: Can TouchUp-G handle other modalities (like, images), in addition to text?
- RQ3 - **General**: Can TouchUp-G handle other down-stream tasks (like node classification), besides link prediction?
- RQ4 - **Principled**: How well is the feature representation learnt by TouchUp-G, according to feature homophily score, compared with features obtained directly from pretrained models?

We first describe our experimental setup, including the datasets and baselines used in our empirical analysis. Then we show the

Table 3: Dataset used in TouchUp-G: These three datasets represent datasets of various scales and modalities. Books and Ogb-Products have original text descriptions as features, and a pretrained BERT is used to generate the feature embeddings. Amazon-CP have images of products as original visual features, and a pretrained ViT is used to generate feature embeddings. LP = Link Prediction. NC = Node Classification.

Name	Nodes	Edges	Description	Node Features	Pretrained Model	Downstream Task
Ogb-Products [18]	2,449,029	61,859,140	Purchasing Network	Text	BERT [9]	LP & NC
Books [46, 47]	1,098,672	33,619,434	Recommendation Network	Text	BERT [9]	LP
Amazon-CP [36]	379,770	4,102,444	Purchasing Network	Image	ViT [11]	LP
Ogb-Arxiv [18]	169,343	1,166,243	Citation Network	Text	SciBERT [1]	NC

practical quantitative improvements from TouchUp-G and a closer ablation study on the feature homophily score.

5.1 Experimental Setup

Data. We mainly focuses on analyzing and enhancing features from feature-rich graphs, where each node need to have natural features that comes from other domains e.g. text, images. In this work, we mainly focus on discussing the case where all nodes in the same graph have features that come from the same modality (text-rich graphs, image-rich graphs). And we leave the case of mixing textual and visual features in one graph to further study. We adapted four public datasets: Ogb-Products, Books, Amazon-CP, Ogb-arxiv. The details of these datasest are shown in Tab. 3.

(Ogb-products): is an Amazon product co-purchasing network where the feature is each products' description [18]. We obtain the raw text feature following [5].

(Ogb-Arxiv): is an a citation network where the feature is each paper's title and abstract [18]. We obtain the raw text feature following [5].

(Books): is a book recommendation dataset from Goodreads [46, 47]. Following [40], the node feature is each book's description and the links capture if a reader who recommenders one book will recommend the other book on Goodreads as well.

(Amazon-CP): is a dataset constructed using the metadata from Amazon-Review dataset [36]. We extract the copurchasing information from the metadata and each product's high resolution image as the raw node features. Nodes with missing visual features and have insufficient density ,degree<5 are eliminated.

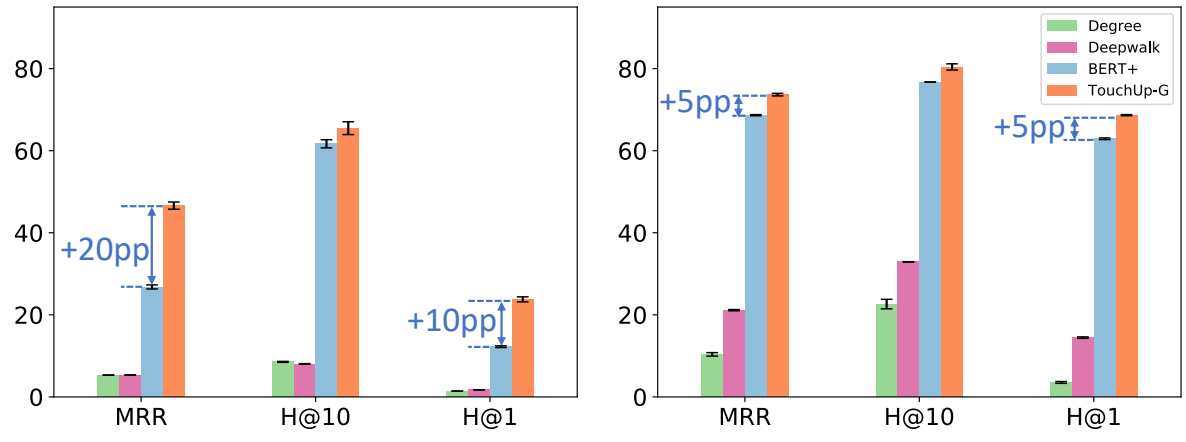
For node classification, we follow the split and evaluation in [18] for both Ogb-Arxiv and Ogb-Products. For link prediction, since we are adapting new datasets to link prediction and no public split is available, we do random split on Ogb-Products, Books, Amazon-CP. Ogb-products and Books are splitted at a ratio of 60%/10%/30%. Amazon-CP is using a 80%/10%/10% split ratio. For every validation and test edges, we randomly generate 1000 negatives for Books dataset, 100 negatives for Ogb-Products and 300 negatives for Amazon-CP. All of these graphs are homogeneous, undirected graphs. The dataset construction pipelines for Books and Amazon-CP dataset will be made public upon acceptance in a notebook, so that others can construct the same dataset following the steps.

Pretrained Models. For Ogb-Products and Books dataset, since their node features are raw text descriptions, we use the 12-layer BERT-base-uncased[3] to generate the pretrained feature embedding [9]. For Ogb-Arxiv, we use SciBERT[4] [1]. For both BERT and SciBERT, the last layer is dropped for both generating feature embeddings and finetuning using TouchUp-G. For Amazon-CP dataset, we adapt the ViT-b-16 model[5] [11] trained on ImageNet [8] to generate the pretrained visual feature embedding. The BERT, SciBERT and ViT models used here are also the pretrained and touchup models we use in TouchUp-G. For ViT, we drop the last layer and add a linear layer to project the embeddings from 768 to 256 dimensions.

TouchUp-G Variants. As shown in Tab. 4, for all four datasets we used, the feature homophily score using PMs directly is smaller than 0.5, thus we adapt TouchUp-G on all of the four datasets. In the experiment part, we mainly test TouchUp-G using two types of models: language models and vision models. When the raw node features are text, BERT is used as the pretrained and touchup model. When the raw node features are images, ViT is used as the pretrained and touchup model. We also consider three GNN backbones: SAGE [14], GCN [25] and GATv2 [2] as GAE-based models. Different from the conventional full-batch GCN, here we use a mini-batch GCN model that follows the same message passing functions as GCN [25]. As a result, we mainly test the property of TouchUp-G for the following five variants: TouchUp-G (BERT-GCN),TouchUp-G (BERT-SAGE), TouchUp-G (ViT-SAGE), and TouchUp-G (ViT-GATv2), TouchUp-G (SciBERT-SAGE).

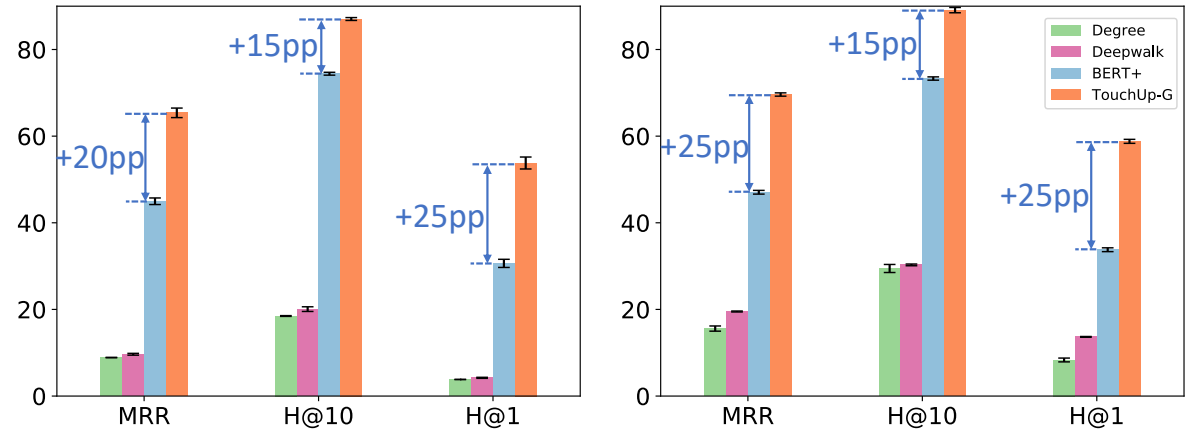
Evaluations. We mainly evaluate the effectiveness of our finetuned feature embedding in two ways. First, we compute the Feature Homophily scores for different feature representations, before using it in the downstream graph tasks. The higher the Feature Homophily score is, the more consistent the features and structures are, and as a result, it's more likely for GNNs to achieve good performance in the downstream graph task. Second, we evaluate the feature embeddings by using them as features for GNNs to do link prediction. We report MRR, Hits@10, and Hits@1, the three most commonly-used evaluation metrics for link prediction [18, 43, 54]. Hits@K counts the ratio of positive edges ranked at the K-th place or above against all negative edges. MRR (Mean Reciprocal Rank) computes the reciprocal rank of the true target node against 1,000 negative candidates, averaged over all the true source nodes. For all evaluation metrics, the higher the number is, the better. We don't report Hits@1 for Amazon-CP because it's 0 for all methods.

[3] https://huggingface.co/bert-base-uncased
[4] https://huggingface.co/allenai/scibert_scivocab_uncased
[5] https://pytorch.org/vision/main/models/generated/torchvision.models.vit_b_16.html



(a) GCN (b) SAGE

Figure 5: TouchUp-G is Effective. Link Prediction performance on Ogb-Products. GCN [25] and SAGE [14] are used as GNN backbones. TouchUp-G performs best against all baselines, by more than 20% of performance increase in MRR.



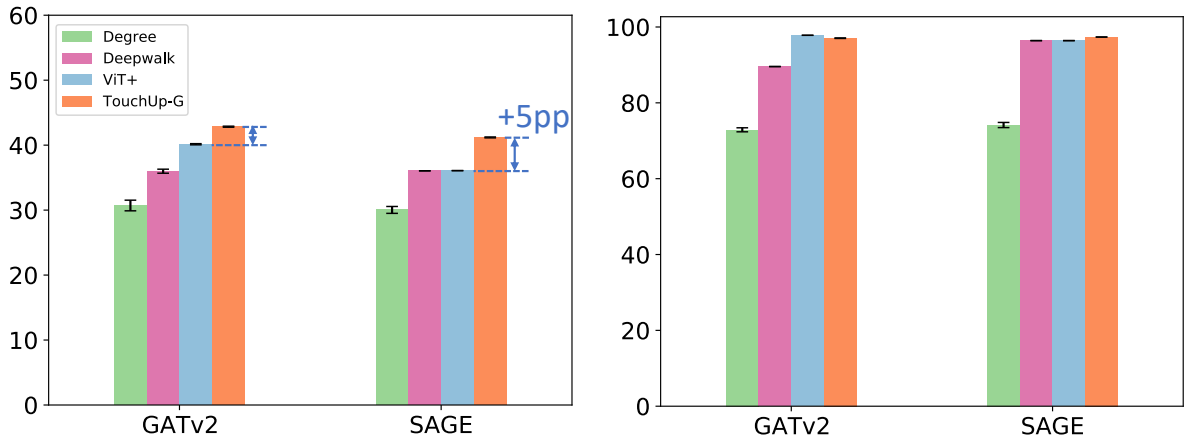
(a) GCN (b) SAGE

Figure 6: TouchUp-G is Effective. Link Prediction performance on Books. GCN [25] and SAGE [14] are used as GNN backbones. TouchUp-G performs best against all baselines, by more than 25% of performance increase in MRR.

Baselines. We mainly compare against various ways of generating feature embeddings as baselines to show that the embeddings generated from TouchUp-G is most effective across tasks and modalities. For link prediction, the main idea is to show that structure-fused feature embeddings works better than embeddings that captures feature or embeddings that captures topology alone. Following [6], we use degree to capture nodes' structure embedding and deepwalk to capture nodes' proximity embedding. For Ogb-Products and Books, we mainly consider BERT+ [9], Degree+ [6], Deepwalk+ [6, 37] as baselines. For Amazon-CP, we mainly consider ViT+ [11], Degree+ [6] and Deepwalk+ [6, 37] as baselines. BERT+indicates using BERT to obtain feature embeddings directly and the feature embeddings are then trained and evaluated on GNNs, and similarly for Degree+ and Deepwalk+.

For node classification, since Ogb-Arxiv and Ogb-Products are both text-rich graphs, we mainly compare against using various language representations to generate feature embeddings. We compare against Ogb+: the original feature embedding that the Ogb benchmark uses [18], BERT+for Ogb-Products, and SciBERT+for Ogb-Arxiv: the two language models we use as pretrain models, without any touchup enhancement. Following [55], we also report performance on DeBERTa+ [15].

Implementation Details. For PMs, we conduct extensive hyperparameter tuning using grid search. We search on the learning rates = {1e-1, 1e-2, 1e-3, 1e-4, 5e-4, 5e-5}. The training batch size is set to 64 for all datasets. We use gradient clipping and early stopping to prevent overfitting on the training edges [29, 41] We used four Nvidia A40 GPU to train the model. Due to the fact that validating on the full validation split for one epoch is extremely time-consuming, we subsample a small set from the full validation test (1% of the edges in the validation set), and use MRR over 5



(a) MRR (b) H@10

Figure 7: TouchUp-G is Multi-modal: Link Prediction performance on Amazon-CP. GATv2 [2] and SAGE [14] are used as GNN backbones. TouchUp-G performs well on image features, indicating TouchUp-G can work well on both text and image features.

negative examples to select the best checkpoint. The touchup model weights from the best performing epoch are then used to generate the finetuned feature embeddings. The full evaluation of how well the feature embeddings perform is done using GNNs.

For GNNs, we also do grid search of hyperparameters. We search on the learning rates = {1e-1, 1e-2, 1e-3, 1e-4, 5e-4, 1e-4, 5e-5} and the number of layers = $\{1, 2, 3\}$. The training batch size and hidden dimension is set to 512 for all datasets. For GATv2, the number of heads is 8. We report the best performing hyperparameters for each setting. We used one Nvidia v100 GPU to train the model and repeat our experiments with three different random seeds. The test results are reported on the epoch with the best validation performance.

5.2 RQ1 - Effective: Text Feature Enhancement

Setup & Evaluation. To evaluate TouchUp-G's effectiveness on enhancing features from a pretrained model (PM), we evaluate the performance of using TouchUp-G on text-rich graphs for link prediction. Link prediction has been widely recognized as foundations of other graphs tasks and can be leveraged to solve the insufficient label problem in tasks like node classification or graph classification [19, 20]. We report the link prediction performance of two widely-used GNN backbones: GCN and SAGE. For each dataset, the three most commonly-used evaluation metrics: MRR, H@10, H@1 are reported. For Ogb-Products, for each positive example, we use 100 negatives during evaluation and for Books, we use 1000 negatives for evaluation.

Results. The results are shown in Fig. 5 and Fig. 6. We can see that TouchUp-G consistently yields better performance across datasets, and with different GNN backbones. Especially, we can see more than 20 % MRR boost on Ogb-products when GCN is used as the backbone, and on Books datasets, in the case of both SAGE and GCN are used. This indicates that structure-fused text features is more effective than any text features or structure features alone, and justifies the effectiveness of TouchUp-G. Additionally, we can see that at all times, BERT+performs significantly better than baselines that focuses on structure information only Degree+and Deepwalk+. This justifies the fact that the information contained in the raw features is informative, even for downstream graph tasks. And the performance increase we get in TouchUp-G is an evidence that by making the node features and graph structure correlation with each other, GNNs are able to best utilize the both the features and the graph structure.

5.3 RQ2 - Multi-modal: Visual Feature Enhancement

Setup & Evaluation. Section 5.2 shows the effectiveness on text-rich graphs for link prediction. Besides text features, we are also interested in understanding TouchUp-G's multi-modal property, if TouchUp-G can work on features besides text. Here we focus on evaluating TouchUp-G on visual features for image-rich graphs. Due to the unavailability of audio- and tactile-graph datasets, we leave other modalities for future study. We report the link prediction performance of two GNN models: GATv2 and SAGE on Amazon-CP. Similar to Sec. 5.2, we also report MRR, H@10, H@1. We generate one negative per edge during training and 300 negatives during evaluation.

Results. The quantitative results are shown in Fig. 7. We can see that TouchUp-G consistently yields better performance on Amazon-CP. and with different GNN backbones. This indicates that structure-fused visual features are more effective than any visual features or structure features alone, and justifies the effectiveness of TouchUp-G. In Fig. 1b, we also provide a qualititative analysis. We show a co-purchasing example that exists in Amazon-CP in the test split. In Fig. 1b, we know that one customer bought a shower head on Amazon, and we want to predict what other products the customer would also buy on Amazon. In the top-2 predictions, ViT+ predicts the family decorations and parking signs, while TouchUp-G predicts the bathroom switch and the tissue hanger, both of which are essentials equipments in the bathroom, just like the shower head. The ground truth label: tissue hanger is also contained in TouchUp-G while missing in ViT+. This qualitative example shows that TouchUp-G yields more meaningful co-purchasing predictions

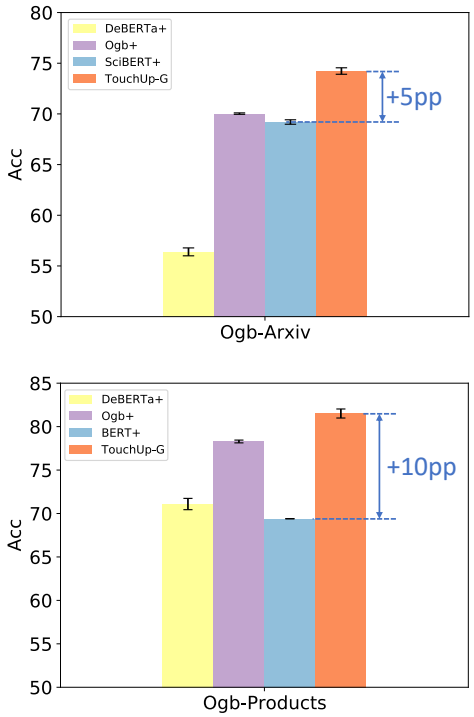


Figure 8: TouchUp-G is General: Node classification Results on Ogb-Arxiv and Ogb-Products. SAGE [14] is used as GNN backbone. TouchUp-G performs best against all baselines, indicating the generality of TouchUp-G to extend to downstream node classification tasks.

compared with ViT+alone, and justifies the ability of TouchUp-G to work on pretrained models from any modality.

5.4 RQ3 - General: Node Classification Results

In section 5.3, we evaluated TouchUp-G's effectiveness on link prediction tasks. To evaluate TouchUp-G's ability on generalizing to different graph downstream tasks, we also report the node classification performance on Ogb-Arxiv and Ogb-Products dataset.

Setup & Evaluation. Following [18], we use its standard evaluation pipeline and reports the test accuracy on the best performing epoch on validation set. The DeBERTa+results are directly adapted from [55], and the Ogb+results are directly adapted from [48]. BERT+ and SciBERT+ indicates the performance of using the features from the PM(BERT), PM(SciBERT) directly, without any finetuning [1, 9]. We use SciBERT for Ogb-Arxiv instead of BERT because Ogb-Arxiv main contains texts from scientific paper and SciBERT is pretrained on scientific paper. For evaluation metrics, we follow standard practice and report accuracy.

Results. The results are shown in Fig. 8. We can see that TouchUp-G consistently yields better performance, comparing against all baselines. Especially, we can see that TouchUp-G even outperforms all SciBERT+ and BERT+ models by 5 % and 10 % respectively. This indicates that the features obtained from TouchUp-G is **General**, can be used to any downstream graph tasks. Additionally, we can see that Ogb+consistently performs better than DeBERTa+, SciBERT+, BERT+. And this justifies our argument that directly using features from a PM without any domain adaption in graphs actually hurts the performance on downstream graph tasks.

Table 4: Principled: TouchUp-G wins. Feature Homophily score before and after TouchUp-G. TouchUp-G gives a much higher Feature Homophily score. As a result, the correlation between node features and graph structure, and it's more likely for GNNs to make best use of the features in downstream graph tasks, and this matches the experiment results in Fig. 5, 6, 7 8. - means BERT,ViT,SciBERT is not applicable to this dataset due to lack of original text/image data or better pretrained model available.

Dataset	BERT [9]	SciBERT [1]	ViT [11]	**TouchUp-G**
Ogb-Products	0.223	-	-	**0.762** (3.4x)
Books	0.137	-	-	**0.579** (4.2x)
Amazon-CP	-	-	0.173	**0.622** (3.6x)
Ogb-Arxiv	-	0.194	-	**0.408** (2.1x)

5.5 RQ4 - Principled: Feature Homophily Score

Setup & Evaluation. For each datasets in Tab. 3, we computes its Feature Homophily scores with embeddings from the PMs, e.g. BERT, ViT, SciBERT, or with embeddings from our touch-up models TouchUp-G, before training a GNN model.

Results. The results are shown in Tab 4. Before finetuning, all datasets' Feature Homophily scores are low. Since Feature Homophily is essentially the pearson correlation between the head and tail node feature embeddings over edges, this indicates that the correlation is very weak and thus hard to capture and be learnt by GNNs, resulting in low GNN performance for downstream graph tasks. However, after TouchUp-G, all datasets exhibits a more than 2× increase in feature homophily score. Thus, GNNs are more likely to perform well on downstream graph tasks using features from TouchUp-G.

6 CONCLUSION

We have presented TouchUp-G, a simple touch-up feature enhancement technique for general pretrained models. TouchUp-G has the following advantages:

- **General**: can be applied to any downstream graph tasks
- **Multi-modal**: can improve raw features that come from any modality (e.g. images, texts);
- **Principled**: a novel metric, feature homophily is proposed to measure the correlation between node features and graph structure
- **Effective**: outperforms baselines on 4 real datasets, with up to 2× performance improvement across various tasks and modalities.

We envision that the finetuning part of TouchUp-G can be done more efficiently using linear probing or delta finetuning, which only tunes a gradient of the pretrained model [10]. We leave this as the next step for this work.

Acknowledgements We thank Shengyi Qian on the helpful advice of vision transformers. We also thank Mark Newman, Andrew Owens, Yongyi Yang, Kaize Ding, Wei Jin, Puja Trivedi, Ang Cao, Mohamed El Banani on the helpful discussions of the paper.

REFERENCES

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

[2] Shaked Brody, Uri Alon, and Eran Yahav. 2021. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491* (2021).

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Yihong Chen, Pushkar Mishra, Luca Franceschi, Pasquale Minervini, Pontus Lars Erik Saito Stenetorp, and Sebastian Riedel. 2022. Refactor gnns: Revisiting factorisation-based models from a message-passing perspective. *Advances in Neural Information Processing Systems* 35 (2022), 16138–16150.

[5] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2021. Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction. *arXiv preprint arXiv:2111.00064* (2021).

[6] Hejie Cui, Zijie Lu, Pan Li, and Carl Yang. 2022. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3898–3902.

[7] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891* (2018).

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904* (2022).

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[13] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964* (2020).

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

[15] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

[16] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2022. Measuring and improving the use of graph information in graph neural networks. *arXiv preprint arXiv:2206.13170* (2022).

[17] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430* (2021).

[18] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[19] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* (2019).

[20] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.

[21] Vassilis N Ioannidis, Xiang Song, Da Zheng, Houyu Zhang, Jun Ma, Yi Xu, Belinda Zeng, Trishul Chilimbi, and George Karypis. 2022. Efficient and effective training of language and graph neural network models. *arXiv preprint arXiv:2206.10781* (2022).

[22] Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. 2022. Heterformer: A Transformer Architecture for Node Representation Learning on Heterogeneous Text-Rich Networks. *arXiv preprint arXiv:2205.10282* (2022).

[23] Di Jin, Xiangchen Song, Zhizhi Yu, Ziyang Liu, Heling Zhang, Zhaomeng Cheng, and Jiawei Han. 2021. Bite-gcn: A new GCN architecture via bidirectional convolution of topology and features on text-rich networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 157–165.

[24] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)* 54, 10s (2022), 1–41.

[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[26] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

[28] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 223–232.

[29] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2020. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International conference on artificial intelligence and statistics*. PMLR, 4313–4324.

[30] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*. 556–559.

[31] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[34] Mark EJ Newman. 2002. Assortative mixing in networks. *Physical review letters* 89, 20 (2002), 208701.

[35] Mark EJ Newman. 2003. Mixing patterns in networks. *Physical review E* 67, 2 (2003), 026126.

[36] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

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

[38] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12179–12188.

[39] Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. 2022. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In *Learning on Graphs Conference*. PMLR, 11–1.

[40] Tara Safavi. 2022. Augmenting Structure with Text for Improved Graph Learning. *PhD Thesis* (2022).

[41] Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. 2021. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176* (2021).

[42] Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*. PMLR, 9448–9457.

[43] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082* (2019).

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[46] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems*. 86–94.

[47] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. *arXiv preprint arXiv:1905.13416* (2019).

[48] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019).

[49] Yongyi Yang, Tang Liu, Yangkun Wang, Zengfeng Huang, and David Wipf. 2021. Implicit vs unfolded graph neural networks. *arXiv preprint arXiv:2111.06592* (2021).

[50] Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf. 2021. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*. PMLR, 11773–11783.

[51] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Linkbert: Pretraining language models with document links. *arXiv preprint arXiv:2203.15827* (2022).

[52] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).

[53] Muhan Zhang and Yixin Chen. 2019. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058* (2019).

[54] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2021. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *Advances in Neural Information Processing Systems* 34 (2021), 9061–9073.

[55] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on Large-scale Text-attributed Graphs via Variational Inference. *arXiv preprint arXiv:2210.14709* (2022).

[56] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[57] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*. 2848–2857.

[58] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.

[59] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems* 34 (2021), 29476–29490.