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
dkoutra@umich.edu

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

ABSTRACT

How can we enhance the node features acquired from Pretrained Models (PMs) to better suit downstream graph learning tasks? Graph Neural Networks (GNNs) have become the state-of-the-art approach for many high-impact, real-world graph applications. For feature-rich graphs, a prevalent practice involves directly utilizing a PM to generate features. Nevertheless, this practice is suboptimal as the node features extracted from PMs are graph-agnostic and prevent GNNs from fully utilizing the potential correlations between the graph structure and node features, leading to a decline in GNN performance. In this work, we seek to improve the node features obtained from a PM for graph tasks and introduce TouchUp-G, a "Detect & Correct" approach for refining node features extracted from PMs. TouchUp-G **detects** the alignment using a novel feature homophily metric and **corrects** the misalignment through a simple touchup on the PM. It is (a) **General**: applicable to any downstream graph task; (b) **Multi-modal**: able to improve raw features of any modality; (c) **Principled**: it is closely related to a novel metric, feature homophily, which we propose to quantify the alignment between the graph structure and node features; (d) **Effective**: achieving state-of-the-art results on four real-world datasets spanning different tasks and modalities.

CCS CONCEPTS

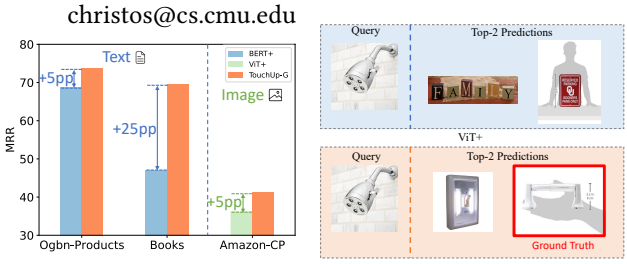
• **Information systems** → **Information retrieval**.

KEYWORDS

graph-centric finetuning; feature homophily; graph neural network

1 INTRODUCTION

Graphs or networks serve as fundamental representations for relational structures, and their analysis is useful in many scientific and industrial applications. Various tasks, ranging from recommendation and molecule property prediction to knowledge graph completion, can be formulated as graph learning endeavors.



(a) Quantitative Result (b) Qualitative Result

Figure 1: TouchUp-G wins: (a) Compared with features obtained directly from PMs (BERT [6] or ViT [8]), TouchUp-G improves the quantitative performance by more than 25% across datasets and modalities. (b) Examples from the Amazon co-purchasing graph (`Amazon-CP`) show that TouchUp-G correctly predicts the ground truth while ViT+ fails.

Pretrained Models (PMs), such as BERT, GPT, and ViT [3, 6, 8] have demonstrated remarkable performance across a range of natural language and computer vision tasks, including question answering and image classification. These models have become the foundations of modern ML systems. In real-world applications, it is common practice to utilize pretrained models for generating node features, and subsequently integrating these derived features into GNNs for graph learning tasks, encompassing link prediction and node classification [4, 12, 13, 18, 40]. Nonetheless, employing node features from PMs without any domain adaptation is suboptimal, as features generated from PMs are unaware of the graph structure and prevents GNNs from fully exploiting the relation between node features and the graph structure.

An illustrative real-world example is shown in Fig. 2, where we leverage a co-purchasing graph to predict products frequently purchased together. Utilizing image features directly from PMs results in distinct feature representations for each product, causing GNNs to falter in predicting products that tend to be purchased together. This is a widespread issue that persists whenever a PM is leveraged to generate node features and the pretraining objective does not take graph structure into account [4]. To overcome this issue, Patton proposes language-specific pretraining [17], while GLEM proposes alternative training of language models and GNNs [40]. These methods have two drawbacks: First, they do not measure the quality of node features from PMs, which decides whether finetuning is necessary. Second, they use text-specific features and only

* Work done during an internship at Amazon.

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07…$15.00
https://doi.org/10.1145/3626772.3657978

Table 1: Qualitative comparison of finetuning frameworks. TouchUp-G is General–it can be applied on any graph tasks–, Multi-modal (applicable to features from any modality), Principled and Effective.

Property \ Method	GLEM [40]	GIANT [4]	Patton [17]	BERT+ [6]	TouchUp-G (ours)
General-NC	✓	✓	✓	✓	✓
General-LP			✓	✓	✓
Multi-modal					✓
Principled		✓			✓
Effective	✓	✓	✓	?	✓

work on text-rich graphs. To address these issues, we introduce a *principled* and *general* solution, which improves node features obtained from *any* PM, so that GNNs can achieve better performance in *any* downstream graph task.

Specifically, we propose TouchUp-G, a "Detect & Correct" approach for refining node features extracted from PMs. To **detect** the alignment between the node features and the graph structure, we introduce a novel feature homophily metric that quantifies the potential correlations between graph structure and node features. Our proposed feature homophily metric accommodates vectorized features and is bounded in the range of [-1,1] so as to allow comparison between graphs. To **correct** the node features obtained from PMs, we propose graph-centric finetuning, a simple TouchUp enhancement technique that improves Graph's node features from any PM.

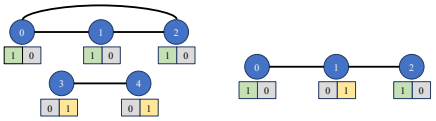
TouchUp-G is simple, adaptable to a variety of PMs in multiple domains. The node features obtained from TouchUp-G show a strong alignment with the graph structure and are able to attain state-of-the-art performance when employed in GNNs for downstream graph tasks. A summary of our results is shown in Fig. 1. Our contributions are summarized as follows:

Figure 2: Why pretrained features may fail: We show a subgraph of Amazon Co-purchasing graph (Amazon-CP). Products possessing disparate visual features are often bought together.

- **General**:TouchUp-G can be applied to a variety of graph tasks.
- **Multi-modal**: TouchUp-G can be applied to any PM from any modality, e.g., texts, images. To the best of our knowledge, we are the first to propose finetuning vision transformers for graph tasks.
- **Principled**: We propose a novel metric called feature homophily, (Eq. (1), § 3.2), to measure the correlation between node features and graph structure.
- **Effective**: TouchUp-G achieves state-of-the-art performance on four real datasets across various tasks and modalities.

2 RELATED WORK

Pretrained Models as Feature Embeddings. Pretrained models, have been widely used to acquire broad language and visual representations [6, 8, 20–23, 28, 30, 37]. For transformers in the language and vision domain, we refer to the survey for more details [19]. The typical approach of learning text-rich graphs adopts a "cascaded architecture" where features are extracted via a PM and incorporated into GNNs. But this approach is suboptimal and several works have been proposed to better adapt features from PMs to GNNs [4, 16, 17, 34, 40]. A comparative analysis between TouchUp-G and previous methods is outlined Table 1. Here, we aim to design a *general* approach that works for any source of modality, such as images, and any graph downstream task. To the best of our knowledge, we are the first to propose finetuning visual representations for graph tasks. Moreover, we propose to measure how misaligned the node features and graph structure are, which is important to decide whether enhancement is needed.



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

Figure 3: Example graphs that exhibit (a) strong positive and (b) strong negative correlation between features and structure. (a): All linked nodes have the same features; (b): All linked nodes have complementary features.

Feature Homophily in GNNs. Existing GNN models heavily rely on the feature homophily assumption. Yoo *et al.* highlight the inability of GNN models to effectively leverage the graph structure in the presence of noisy node features, primarily due to their strong dependence on the feature matrix [38]. Yang *et al.* show that GNNs penalize deviations between the embeddings of two nodes sharing an edge [35, 36]. However, what is a good metric for feature homophily remains unclear. Newman introduced assortativity as a measure to assess the similarity of scalar features along edges [25, 26], but it cannot be applied to vectorized features. Feature smoothness is another candidate metric [11], but it is unbounded in magnitude, making it hard to compare against graphs with varying sizes. In contrast, our proposed feature homophily score accommodates vectorized features and allows comparison across graphs of different sizes.

3 TOUCHUP-G

3.1 Preliminaries

Graphs. We consider a **graph** $G = (V, E, S)$, where V is the set of vertices, E is the set of edges, S is the set of raw node features, (e.g. raw text, images), and $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix.

Node Features X from PMs. Denote $\mathbf{T}$ as a pretained model of any modality, and $\mathbf{X} = \mathbf{T}(S)$. $\mathbf{X} \in \mathbb{R}^{|V|\times d}$ is the extracted d-dimensional node feature embedding from the pretrained model $\mathbf{T}$. We use $\mathbf{X} \in \mathbb{R}^{|V|\times d}$ as features in GNNs.

Link Prediction. Predict whether there will be a future link e_{ij} between a pair of nodes i and j, where $i, j \in V$ and $e_{ij} \notin E$.

Node Classification. Given the labels of l nodes in G, where $l \ll |V|$, predict the unknown classes of $|V| - l$ test nodes.

3.2 Detection: Proposed Homophily Measure

How can we detect the discrepancy between features and structure? An intuitive approach is to use cosine similarity between nodes

or measure the feature smoothness over nodes. However, these methods have several drawbacks, which we outline next.

Intuitive Measures & their Limitations. The most intuitive way to measure node similarity over connected edges is to measure the *cosine similarity* between nodes, which is not ideal. As shown in Figure 3(b), the graph's cosine similarity is 0 and fails to differentiate random features from negatively correlated features.

Another candidate measure is *feature smoothness* [11]. It is defined as $\lambda_f = \frac{||\sum_{v\in V}(\sum_{v'\in N_v}(x_v - x_{v'}))^2||_1}{|E|\cdot d}$, where N_v are neighbors of node v and $\lambda_f \in [0, +\infty)$. However, this metric suffers from two drawbacks. First, the magnitude of λ_f is unbounded and depends on the number of edges. Thus, feature smoothness cannot be compared across graphs of varying sizes. Second, similar to cosine similarity, feature smoothness fails to differentiate between random and negatively correlated features, as both generate a large λ_f.

Proposed Feature Homophily Measure. Motivated by the limitations of the above-mentioned intuitive metrics, we now introduce a new feature homophily measure h_f, which can effectively quantify the alignment between the node features and graph structure.

Definition 1 (Feature homophily). *Given a graph $G = (V, E)$ and the feature embeddings x_i, x_j of nodes i,j where $e_{ij} \in E$, the feature homophily ratio h_f is defined as follows:*

$$h_f = \frac{\sum_{e_{ij}\in E}(x_i - \bar{x})\cdot(x_j - \bar{x})}{\sqrt{\sum_{e_{ij}\in E}(x_i - \bar{x})\cdot(x_i - \bar{x})}\cdot\sqrt{\sum_{e_{ij}\in E}(x_j - \bar{x})\cdot(x_j - \bar{x})}} \quad (1)$$

where $\bar{x} = \frac{\sum_{e_{ij}\in E}(x_i + x_j)}{2|E|}$.

The feature homophily score h_f effectively measures the correlation between nodes over edges. Its range is confined between $[-1, 1]$. In various real-world cases, most graphs exhibit a feature homophily where $h > 0$. Figure 3 illustrates that when $h = 1$, every connected node pair possesses the *exact same* feature, and when $h = -1$, each connected node pair exhibits the *exact opposite* feature. When $h = 0$, the linked nodes showcase random features with no correlation with the graph connectivity.

Difference between Feature Homophily vs. Label Homophily. In the GNN literature, the most popular notion of homophily is *label* homophily [24, 41, 42]. While label homophily is related to feature homophily, they are inherently different. Label homophily captures the label similarity between nodes and their immediate neighbors, so it is contingent on the availability of node labels, a scenario commonly seen in node classification tasks. Nonetheless, even in such cases, due to the cost of labeling, access to node labels is primarily limited to a small subset prior to inference. In this work, we tackle different graph learning tasks, including but not limited to link prediction, wherein nodes lack class labels.

3.3 Correction: Graph-Centric Finetuning

How can we correct the features? We propose graph-centric finetuning, TouchUp-G, a simple TouchUp enhancement that refines node features from any PM. Formally, **given**:

- an undirected graph $G = (V, E)$ and its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$;
- the set S of raw node features for all nodes $n \in V$; and
- a pretrained model $\mathbf{T}$ that transforms raw node features to node feature embeddings, $\mathbf{X} = \mathbf{T}(S)$, $\mathbf{X} \in \mathbb{R}^{|V|\times d}$;

We **fine-tune** $\mathbf{T}$ to minimize L_{struct} using negative sampling:

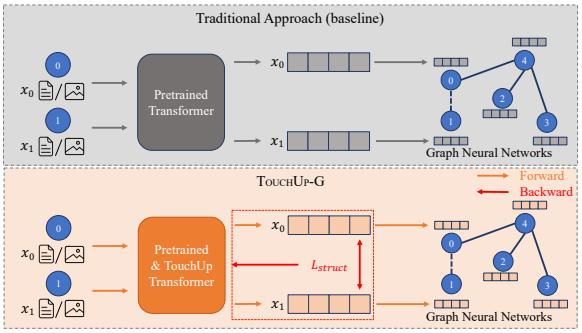


Figure 4: Overview of TouchUp-G. [Top, ■] A PM [6, 8] is used to extract features from raw text or images, and then GNNs are trained upon the extracted features. [Bottom, ■] We propose graph-centric finetuning on PMs to correct the discrepancy between features and the structure.

Table 2: Datasets used in TouchUp-G LP = Link Prediction. NC = Node Classification.

Name	Nodes	Edges	Node Features	Pretrained Model	Downstream Task
Ogb-Products[13]	2,449,029	61,859,140	Text	BERT [6]	LP & NC
Books [31, 32]	1,098,672	33,619,434	Text	BERT [6]	LP
Amazon-CP [27]	379,770	4,102,444	Image	ViT [8]	LP
Ogb-Arxiv [13]	169,343	1,166,243	Text	SciBERT [1]	NC

$$L_{\text{struct}} = -\frac{1}{|E|}\sum_{(u,v)\in E}(\log(x_u \cdot x_v) + \log(1 - x_u \cdot x_{v'})), \quad (2)$$

where $v' \in V$ is a randomly sampled negative and $(u, v) \notin E$.

An illustrative example is provided in Fig. 4. We note that, for link prediction, edges used for validation and testing are unobserved, and thus are not used for finetuning.

Non-specific Loss for PMs. Unlike previous works that hinge on the domain-specific knowledge of text [17], TouchUp-G does not rely on prior assumptions about the pretrained models or the source of node features. It can be readily adapted to any PM from any modality, including language models for text, vision transformers for images. In our experiments, we demonstrate that TouchUp-G operates efficiently on both text and image-rich graphs.

4 EXPERIMENTS

We aim to answer the following research questions:

- (**RQ1**) **Multi-modal**: Can TouchUp-G handle any modality such as text, images on link prediction?
- (**RQ2**) **General**: Can TouchUp-G handle other downstream tasks such as node classification?
- (**RQ3**) **Principled**: According to the feature homophily score h_f, how correlated are the features from PMs vs. the feature from TouchUp-G?

Data. We use four public datasets: `Ogb-Products`, `Books`, `Amazon-CP`, `Ogb-Arxiv`. The details of these datasets are shown in Table 2. `Books` is constructed following [29]. `Amazon-CP` is constructed from the Amazon-Review [27] by extracting the co-purchasing information from the metadata and using each product's high resolution image as the raw node features. Nodes with missing images or insufficient density (i.e., degree< 5) are eliminated.

Pretrained Models. For `Ogb-Products` and `Books` , we use BERT [6]. For `Ogb-Arxiv`, we use SciBERT [1]. For BERT and SciBERT, the last layer is dropped when generating representations. For `Amazon-CP`, we adapt the ImageNet pretrained ViT [8]. We drop the last layer and add a linear layer to project the embeddings to 256 dimensions.

Table 3: TouchUp-G is multi-modal: Link Prediction performance on text-rich graphs. TouchUp-G has the best overall performance across all datasets and modalities. We do not report Patton [17] on Amazon-CP as it only works for text features. OOM = Out of Memory.

Datasets	Methods	SAGE		GATv2	
		MRR ↑	H@10 ↑	MRR ↑	H@10 ↑
Books	Degree+	15.58 ± 0.60	29.45 ± 0.93	OOM	OOM
	Deepwalk+	19.52 ± 0.11	30.28 ± 0.20	OOM	OOM
	BERT+	47.05 ± 0.43	73.30 ± 0.38	OOM	OOM
	Patton	65.55 ± 0.08	86.00 ± 0.02	OOM	OOM
	TouchUp-G	69.61 ± 0.38	89.09 ± 0.61	OOM	OOM
Ogb-Products	Degree+	10.37 ± 0.42	22.62 ± 1.16	OOM	OOM
	Deepwalk+	21.14 ± 0.15	32.90 ± 0.06	OOM	OOM
	BERT+	68.65 ± 0.14	76.73 ± 0.02	OOM	OOM
	Patton	71.70 ± 0.01	77.66 ± 0.04	OOM	OOM
	TouchUp-G	73.66 ± 0.31	80.40 ± 0.76	OOM	OOM
Amazon-CP	Degree+	30.03 ± 0.53	74.15 ± 0.01	30.71 ± 0.82	72.89 ± 0.53
	Deepwalk+	36.05 ± 0.01	96.41 ± 0.01	35.99 ± 0.31	89.57 ± 0.01
	ViT+	36.08 ± 0.01	96.41 ± 0.01	40.14 ± 0.09	97.81 ± 0.01
	TouchUp-G	41.19 ± 0.07	97.35 ± 0.01	42.84 ± 0.08	97.05 ± 0.07

TouchUp-G Variants. As shown in Table 4, we detect that, for all four datasets, h_f is close to 0 when using PMs directly, thus we do correction on all the datasets. We consider two GNN backbones: SAGE [9] and GATv2 [2].

Metrics. We evaluate the performance on link prediction and node classification. For link prediction, we report MRR, Hits@10, the two most commonly-used evaluation metrics [13, 39]. For node classification, we report accuracy, following [12]. For all evaluation metrics, the higher the number is, the better. We perform hyperparameter tuning using grid search and choose the best performing ones on validation sets. Results are reported on test sets.

Baselines. As in [5], we use Degree+ and Deepwalk+ as embedding-based baselines. For PMs, we mainly consider BERT+[6], and ViT+[8]. We also report Patton [17], the most recent framework that captures the dependency between textual attributes and structure. For node classification,we compare against Ogb+: the original features [13], BERT+, SciBERT+and DeBERTa+ [10]. We also report results for two methods that finetune LLMs, GIANT and GLEM [4, 40].

4.1 (RQ1) Multi-modal: Link Prediction Results

Setup & Evaluation. We first evaluate TouchUp-G's effectiveness on link prediction for feature-rich graphs. Here, we consider text and image—due to the lack of data, we leave other modalities for future study. Link prediction has been recognized as building block for other tasks and can be leveraged to solve the insufficient label problem in other tasks [14, 15]. We report the performance of two widely-used GNN backbones: SAGE and GATv2.

Results. The quantitative results are shown in Table 3. TouchUp-G achieves state-of-the-art performance at most times with different GNN backbones, even compared with Patton. Especially, there is more than 20% MRR boost compared with BERT+. This indicates that structure-fused features are more effective than any feature or structure alone. In Fig. 1b, we also provide a qualitative co-purchasing example in Amazon-CP during test. Given the query "shower head", our goal is to predict "bath towel hanger" (ground truth) as a co-purchased item. While ViT+ fails, TouchUp-G correctly predicts the ground truth in the top-2 predictions. Moreover, the "tissue hanger" that TouchUp-G predicts is also related to bathroom equipment and is likely to be purchased together with shower heads. This qualitative example shows that TouchUp-G yields more meaningful co-purchasing predictions compared with ViT+ alone, and showcases TouchUp-G's multimodal ability.

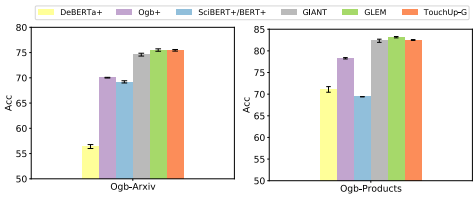


Figure 5: TouchUp-G is General: Node classification Results. *TouchUp-G does not use any node label information during training.* However, we obtain comparable performance compared with baselines explicitly finetuned on node labels [40].

Table 4: Principled: TouchUp-G wins. Feature homophily h_f on various methods. TouchUp-G gives more than 2× increase in feature homophily and indicates better correlation between node features and graph structure. Gain↑ denotes the improvements of TouchUp-G over the same PMs.

Dataset	BERT+ [6]	SciBERT+ [1]	ViT+ [8]	Patton [17]	TouchUp-G	Gain ↑
Books	0.137	-	-	0.2373	**0.579** (4.2x)	47%
Ogb-Products	0.223	-	-	0.4595	**0.762** (3.4x)	7%
Amazon-CP	-	-	0.173	-	**0.622** (3.6x)	14%
Ogb-Arxiv	-	0.194	-	-	**0.408** (2.1x)	9%

4.2 (RQ2) General: Node Classification Results

Setup & Evaluation. We report the node classification performance on Ogb-Arxiv and Ogb-Products. Results for DeBERTa+, GIANT, GLEM are directly adapted from [40], and the Ogb+results are directly adapted from [33].

Results. The results are shown in Fig. 5. TouchUp-G is comparable with GIANT and GLEM. This observation is noteworthy as we do not use any node label information in TouchUp-G. Our method surpasses all SciBERT+ and BERT+, indicating the generality of TouchUp-G across backbones. Moreover, Ogb+ consistently outperforms PM baselines. This further supports our argument that directly utilizing contextualized features from PMs without domain adaptation can negatively impact the performance when the contextualization is irrelevant, and highlights the necessity of TouchUp-G.

4.3 (RQ3) Principled: Feature Homophily Score

Setup & Evaluation. For each dataset in Table 2, we compute h_f using the representations derived from both the PMs and TouchUp-G, before training a GNN.

Results. The results are shown in Table 4. All datasets exhibit low feature homophily scores prior to fine-tuning. This suggests a lack of alignment between the node features and graph structure across all datasets. However, upon applying TouchUp-G, all datasets witness more than 2× increase in h_f, and the state-of-the-art performance on downstream graph tasks. Patton also shows increase in h_f when compared with BERT+, which suggests that GNNs' performance can be improved by increasing h_f.

5 CONCLUSION

We have presented TouchUp-G, a simple "Detect & Correct" approach for refining node features extracted from PMs. TouchUp-G is **General**, **Multi-modal**, **Principled** and **Effective**. As future work, we envision that the finetuning part of TouchUp-G can be done more efficiently using delta finetuning [7].

REFERENCES

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

[2] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? *ICLR* (2022).

[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. *NeurIPS* 33 (2020), 1877–1901.

[4] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2022. Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction. *ICLR* (2022).

[5] 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 *CIKM*. 3898–3902.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT* (2019).

[7] 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).

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR* (2021).

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS* 30 (2017).

[10] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *ICLR* (2021).

[11] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2020. Measuring and improving the use of graph information in graph neural networks. *ICLR* (2020).

[12] 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. *NeurIPS* (2021).

[13] 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. *NeurIPS* 33 (2020), 22118–22133.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. *ICLR* (2020).

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*. 1857–1867.

[16] 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).

[17] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. 2023. Patton: Language Model Pretraining on Text-Rich Networks. *ACL* (2023).

[18] Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. 2023. Heterformer: A Transformer Architecture for Node Representation Learning on Heterogeneous Text-Rich Networks. *KDD* (2023).

[19] 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.

[20] 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).

[21] 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. *ACM Computing Surveys(CSUR)* 55, 9 (2023), 1–35.

[22] 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).

[23] 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 *CVPR*. 10012–10022.

[24] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134* (2021).

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

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

[27] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*. 188–197.

[28] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision transformers for dense prediction. In *CVPR*. 12179–12188.

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

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

[31] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys*. 86–94.

[32] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. *ACL* (2019).

[33] 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).

[34] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. 2023. Graph-Aware Language Model Pre-Training on a Large Graph Corpus Can Help Multiple Graph Applications. *KDD* (2023).

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

[36] 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 *ICML*. PMLR, 11773–11783.

[37] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Linkbert: Pretraining language models with document links. *ACL* (2022).

[38] Jaemin Yoo, Meng-Chieh Lee, Shubhranshu Shekhar, and Christos Faloutsos. 2023. Less is More: SlimG for Accurate, Robust, and Interpretable Graph Mining. In *KDD*. 3128–3139.

[39] 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. *NeurIPS* 34 (2021), 9061–9073.

[40] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on Large-scale Text-attributed Graphs via Variational Inference. *ICLR* (2023).

[41] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *AAAI*, Vol. 35. 11168–11176.

[42] 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. *NeurIPS* 33 (2020), 7793–7804.