A Fused Gromov-Wasserstein Approach to Subgraph Contrastive Learning

Anonymous authors
Paper under double-blind review

Abstract

Self-supervised learning has become a key method for training deep learning models when labeled data is scarce or unavailable. While graph machine learning holds great promise across various domains, the design of effective pretext tasks for self-supervised graph representation learning remains challenging. Contrastive learning, a popular approach in graph self-supervised learning, leverages positive and negative pairs to compute a contrastive loss function. However, current graph contrastive learning methods often struggle to fully use structural patterns and node similarities. To address these issues, we present a new method called **F**used Gr**o**mov Wa**s**serstein **S**ubgraph Contrast**i**ve **L**earning (FOSSIL). Our method integrates node-level and subgraph-level contrastive learning, seamlessly combining a standard node-level contrastive loss with the Fused Gromov-Wasserstein distance. This combination helps our method capture both node features and graph structure together. Importantly, our approach works well with both homophilic and heterophilic graphs and can dynamically create views for generating positive and negative pairs. Through extensive experiments on benchmark graph datasets, we show that FOSSIL outperforms or achieves competitive performance compared to current state-of-the-art methods.

1 Introduction

Self-Supervised Learning (SSL) is a methodology employed in machine learning to extract meaningful and transferable representations from unlabeled data. SSL has been successfully used in many areas like computer vision (Tomasev et al., 2022), speech processing (Baevski et al., 2020), and natural language processing (Devlin et al., 2019; Lewis et al., 2019). SSL is particularly important when labeled data is impractical or costly to obtain. Graph Representation Learning (GRL) stands out as a domain where SSL excels due to the complexity involved in annotating graphs, which often demands specialized knowledge and cannot be easily crowd-sourced. For instance, in biochemistry (Zang et al., 2023), annotating molecular graphs requires specialized knowledge, significantly increasing the labeling effort compared to other data modalities.

In SSL, Contrastive Learning (CL) is a prominent approach (Chen et al., 2020). CL methods typically involve sampling an anchor from the dataset, generating an augmentation (positive pair), and contrasting it with negative pairs using a CL loss function. Unlike certain modalities, such as images, the generation of positive pairs in GRL presents unique challenges. Various methods have been proposed, including those involving node-level or subgraph-level contrastive loss functions (Zhu et al., 2020b; Yuan et al., 2023; Han et al., 2022; Jiao et al., 2020). Recent studies have shown that subgraph-level approaches outperform node-level ones, as they can capture the structural similarities between the anchor and positive graph pair (Han et al., 2022).

Prior subgraph-level CL methods often use readout functions to extract subgraph representations (Jiao et al., 2020; Velickovic et al., 2019). Alternatively, some approaches rely on Optimal Transport (OT) distances for computing the CL loss (Han et al., 2022). OT-based techniques exhibit superior performance compared to readout-based methods in subgraph-level CL. Notably, the Generative Subgraph Contrast (GSC) method (Han et al., 2022) leverages a combination of the well-established Wasserstein Distance (WD) (Villani et al., 2009) and Gromov-Wasserstein Distance (GWD) (Mémoli, 2011) from OT. However, existing OT-based methodologies face two challenges: separately computing the WD and GWD does not capture both structural and feature information simultaneously, and the GNN encoder struggles with heterophilic datasets.

Motivated by the limitations of previous subgraph CL methodologies, we introduce a new GNN approach employing the Fused Gromov-Wasserstein Distance (FGWD) (Vayer et al., 2018; Brogat-Motte et al., 2022) from OT. The FGWD captures both feature and structural similarities jointly. Our model, termed **F**used Gr**o**mov Wa**s**serstein **S**ubgraph Contrast**i**ve **L**earning (FOSSIL), is designed to be robust against variations in homophily levels within the graph dataset. The proposed method uses a decoupled scheme to process heterophilic data more effectively. Furthermore, FOSSIL leverages the FGWD to simultaneously encode the structural and feature characteristics of our subgraphs for calculating the CL loss. We conduct a comprehensive comparative analysis of FOSSIL against state-of-the-art models across various homophilic and heterophilic datasets for self-supervised node classification tasks. Our findings demonstrate that FOSSIL either outperforms existing methodologies or achieves competitive results, thereby showcasing its efficacy in graph SSL tasks. The main contributions of our paper can be summarized as follows:

- We introduce a novel approach, named FOSSIL, that effectively captures both feature and structural similarities simultaneously with the FGWD, addressing the limitations of previous OT-based SSL methods in GRL.
- We design a robust architecture capable of accommodating varying levels of homophily within graph datasets. By leveraging a decoupled scheme for the GNN encoder, FOSSIL performs consistently well across homophilic and heterophilic datasets. This robustness enhances the applicability of the proposed method in real-world scenarios.
- We conduct an extensive empirical evaluation to validate the effectiveness of FOSSIL against state-of-the-art models in self-supervised node classification tasks across diverse datasets. We also thoroughly validate each design choice through a series of ablation studies to validate each component of FOSSIL.

2 Related Work

2.1 Graph Neural Networks

GNNs are neural architectures designed for graphs (Wu et al., 2021). They usually extract information via message-passing mechanisms between nodes. This process can be divided into two steps: message aggregation from the neighborhood, and updating of the node embedding (Gilmer et al., 2017). The key difference between state-of-the-art GNN architectures lies in the aggregation functions. For example, Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) perform a weighted average of the messages inside the 1-hop neighborhood, where weights are fixed and defined with respect to the node degrees. Graph Attention Networks (GATs) (Veličković et al., 2018) compute learnable attention weights thanks to interactions inside the 1-hop neighborhood. Nevertheless, most of the GNN architectures are built under the homophily assumption. This assumption is pretty useful in graph datasets where nodes have similar characteristics. Therefore, these GNNs have poor node classification performance in heterophilic graph datasets.

Previous research works have tried to address the heterophilic limitation of classical message-passing neural networks. For example, GPR-GNN (Chien et al., 2021) uses a polynomial graph filter with coefficients learned altogether with the linear projection layer used in GNNs. H2GCN (Zhu et al., 2020a) proposes a set of designs to improve the node classification performance of GNN on heterophilic graphs: feature-structure embedding separation, aggregating in higher-order neighborhoods, and combination of intermediate representations (which is equivalent to adding skip connections between GNN layers). From a graph signal processing perspective (Isufi et al., 2024), GCNs are considered low-pass filters (NT & Maehara, 2019). Therefore, ACM (Luan et al., 2022) adaptively combines information extracted by a graph low-pass filter, high-pass filter, and an MLP to address heterophilic networks.

2.2 Self-supervised Learning on Graphs

The key idea of SSL on graphs is to design and solve some pretext tasks that do not require label information. Depending on how the pretext task is defined, SSL methods for graphs can be divided into two categories:

predictive and *contrastive* methods. Predictive methods aim to learn meaningful representations where the output is typically a perturbed version of the input graph. For example, BGRL (Thakoor et al., 2021) learns node representations by encoding two perturbed versions of the input graph with an online encoder and a target encoder. The online encoder is trained to predict the representation of the target encoder, and the target encoder is updated as an exponential moving average of the online encoder. BNLL (Liu et al., 2024) improves the BGRL model by introducing a few noisy-ground truth positive node pairs. Based on the homophily assumption that neighboring nodes often share the same label, they include additional candidate positive pairs to the objective of BGRL. Specifically, for each node, BNLL adds the cosine similarity of its online representation and the target representations of its neighbors weighted by its attention weights with its neighbors. VGAE (Kipf & Welling, 2016) uses a variational autoencoder approach in graphs to predict the same graph and feature as in the input. Finally, GPT-GNN (Hu et al., 2020b) performs an autoregressive reconstruction of the graph.

Contrastive methods, which are also the focus of this paper, have shown better performance than predictive methods for SSL on graphs in general. Such methods can be divided into three categories depending on how the data pairs are defined: node-to-node, graph-to-graph, and node-to-graph. For example, GRACE (Zhu et al., 2020b) generates two perturbed views of the original graph and performs contrastive learning between nodes of these views. MUSE (Yuan et al., 2023) extracts node-wise semantic, contextual, and fused embeddings to perform node-to-node contrastive learning for each of these embeddings. Nevertheless, node-level contrastive learning approaches are suboptimal since they cannot easily capture global information about the structure of the graphs.

Regarding subgraph level contrast, DGI (Velickovic et al., 2019) performs node-to-graph contrast by extracting the node embeddings of the original graph and a perturbed version, then it increases or decreases the agreement between the original or perturbed node embeddings using a readout function. Though spectral polynomial filters such as GPR-GNN Chien et al. (2021) and ChebNetII He et al. (2022) are more expressive than GCNs and can adapt to arbitrary homophily levels, they underperform GCNs when used as an encoder for traditional SSL methods. PolyGCL Chen et al. (2024) tries to solve this problem by restricting the expressiveness of the polynomial filters from a spectral perspective to construct the low-pass and high-pass views and introduces a simple linear combination strategy to construct the optimization objective. Specifically, they follow a similar approach to DGI (Velickovic et al., 2019) with the difference that they do it with both high/low-frequency embeddings extracted with high-pass and low-pass polynomial filters sharing weights. Subg-Con (Jiao et al., 2020) follows up the work of DGI by doing the same contrast but at a subgraph level. It starts by sampling a set of anchor nodes, followed by the extraction of subgraphs utilizing the personalized PageRank algorithm. Overall, the model aims to adjust the agreement level between an anchor node and the sampled subgraph for the positive and negative pairs. Methods such as DGI and Subg-Con, however, use a readout embedding to characterize graphs, which ignores the structure of the graph. GSC (Han et al., 2022) addresses this issue by performing subgraph-level contrast using WD and GWD from OT as the measure of subgraph similarity.

FOSSIL is a combined subgraph-level and node-level contrastive learning approach. It uses OT distance metrics to measure the dissimilarity between subgraphs while further incorporating node-level contrast to capture differences among nodes. While sharing common features, our method differs from GSC in three key aspects: (i) we jointly capture structural and feature information in the OT distance using the FGWD, (ii) we also perform node-level contrast to emphasize the node's difference, and (iii) we have a specialized architecture that is robust to heterophilic and homophilic datasets.

3 Preliminaries

3.1 Mathematical Notation

In this paper, matrices are represented with uppercase bold letters like $\mathbf{A}$, while lowercase bold letters like $\mathbf{x}$ denote vectors. The symbol $\langle \cdot, \cdot \rangle$ denotes the matrix scalar product associated with the Frobenius norm. For an element x of a measured space, δ_x represents the Dirac measure at x. The vector $\mathbb{1}_n \in \mathbb{R}^n$ denotes the vector of all ones of dimension n. We denote the simplex histogram with n bins as $\Sigma_n = \{\mathbf{h} \in \mathbb{R}^n_+ \mid$

$\sum_{i=1}^{n} \mathbf{h}_i = 1\}$. Let $\underline{\mathbf{L}}$ be a 4-way tensor and $\mathbf{B}$ be a matrix. The tensor-matrix multiplication between $\underline{\mathbf{L}}$ and $\mathbf{B}$ is denoted as the matrix $\underline{\mathbf{L}} \otimes \mathbf{B}$, where $(\underline{\mathbf{L}} \otimes \mathbf{B})_{i,j} = \sum_{k,l} \underline{\mathbf{L}}_{i,j,k,l} \mathbf{B}_{k,l}$. The element-wise (Hadamard) product operator between two matrices is denoted as $\odot$. The fraction $\frac{\mathbf{u}}{\mathbf{v}}$ denotes the element-wise division of vectors $\mathbf{u}$ and $\mathbf{v}$.

3.2 Graph Neural Networks

We consider attributed undirected graphs, denoted as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents the adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N \times F}$ denotes the feature matrix, and N and F respectively denote the number of nodes and the dimensionality of each node feature vector. A GNN is a function $f : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times D}$, where $\mathbf{H} = f(\mathbf{A}, \mathbf{X}) \in \mathbb{R}^{N \times D}$ represents the node embeddings matrix of $\mathcal{G}$, with each row $\mathbf{H}_i$ corresponding to the embedding of node v_i (Wu et al., 2021). Additionally, for a vector $\mathbf{v} \in \mathbb{R}^N$, $\operatorname{diag}(\mathbf{v}) \in \mathbb{R}^{N \times N}$ denotes the diagonal matrix with diagonal elements from $\mathbf{v}$. Let $S \subseteq \{1, \ldots, N\}$ be a set of indices; $\mathbf{A}[S; S]$ represents the adjacency matrix of the subgraph of $\mathcal{G}$ restricted to nodes indexed by S, and $\mathbf{X}[S]$ denotes its corresponding feature matrix.

3.3 Fused Gromov-Wasserstein Distance

The main purpose of OT is to find the optimal way to shift objects from one location to another in a metric space. This is done by finding a minimum cost soft matching between two probability distributions, one on objects of each location. Let $\boldsymbol{\mu} \in \Sigma_n$ and $\boldsymbol{\nu} \in \Sigma_m$ be two probability distributions, respectively, over the objects of the first and the second location. The WD (Villani et al., 2009) uses a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ of transportation costs (or pair-wise distances) between location 1 and 2 to find a soft matching that minimizes the overall transportation cost. However, locations 1 and 2 may not lie in the same space, preventing us from defining the transportation cost $\mathbf{M}$. GWD (Mémoli, 2011) solves this problem by using internal transportation cost matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ in each space separately to find the soft matching. The purpose is to match objects that have similar structural behaviors inside their metric spaces.

The main downside of WD and GWD is that they fail to capture both structural and feature information (Vayer et al., 2018) when used to compare graphs. In fact, WD only uses node features to define its cost matrix $\mathbf{M}$ while GWD only uses each graph's adjacency matrix to define its internal costs matrices $\mathbf{C}_1$ and $\mathbf{C}_2$. To alleviate this issue, Vayer et al. (2018) proposed the Fused Gromov-Wasserstein Distance (FGWD). The idea is to use a coefficient $\alpha \in [0, 1]$ that trades off between WD and GWD. The FGWD between the two locations is defined as follows:

$$\mathrm{FGWD}_{\alpha, \mathbf{M}, \mathbf{C}_1, \mathbf{C}_2}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \min_{\mathbf{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \langle \alpha \mathbf{M} + (1 - \alpha) \underline{\mathbf{L}}(\mathbf{C}_1, \mathbf{C}_2) \otimes \mathbf{P}, \mathbf{P} \rangle, \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{n \times m}$ is a soft matching (or joint probability distribution) between objects of location 1 and 2, $\underline{\mathbf{L}}(\mathbf{C}_1, \mathbf{C}_2)_{i,j,k,l} = |\mathbf{C}_1[i; k] - \mathbf{C}_2[j; l]|^2$, and $\boldsymbol{\Pi}(\boldsymbol{\mu}, \boldsymbol{\nu})$ is the set of joint probability distributions with marginals $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. FGWD is a generalization of WD and GWD. We recover WD for $\alpha = 1$ and GWD for $\alpha = 0$.

3.4 Graph Contrastive Learning (GCL)

GCL (You et al., 2020; Xie et al., 2021) is built upon the Mutual Information Maximization principle (MIM) Linsker (1988). Given a graph $\mathcal{G}$ sampled from a graph distribution $\mathbb{P}_{\mathcal{G}}$, the goal is to learn an encoder $f(\cdot)$ by maximizing the mutual information between $\mathcal{G}$ and its encoded version $f(\mathcal{G})$. The encoder f should be able to distinguish different graphs. Hence we solve:

$$\max \mathrm{MI}(\mathcal{G}; f(\mathcal{G})), \text{ where } \mathcal{G} \sim \mathbb{P}_{\mathcal{G}}. \tag{2}$$

For computational efficiency, in practice, we maximize an estimated lower bound $\widehat{\mathrm{MI}}$ of the mutual information. These lower bounds are used to maximize the similarity between positive pairs and minimize the similarity between negative ones. Formally, given a graph $\mathcal{G} \sim \mathbb{P}_{\mathcal{G}}$, and a graph view generation operator $T(\mathcal{G})$ that creates different but semantically similar versions $t(\mathcal{G})$ of $\mathcal{G}$, the problem writes:

$$\max \widehat{\mathrm{MI}}(f(\mathcal{G}), f(t(\mathcal{G}))), \text{ where } \mathcal{G} \sim \mathbb{P}_{\mathcal{G}} \text{ and } t(\mathcal{G}) \sim T(\mathcal{G}). \tag{3}$$

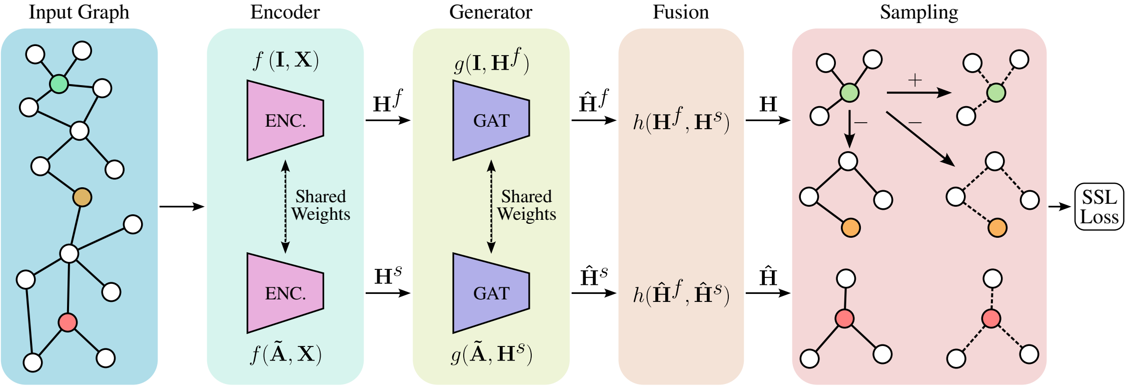
Input Graph
Encoder
Generator
Fusion
Sampling
$f(\mathbf{I}, \mathbf{X})$
ENC.
Shared Weights
ENC.
$f(\tilde{\mathbf{A}}, \mathbf{X})$
$\mathbf{H}^f$
$\mathbf{H}^s$
$g(\mathbf{I}, \mathbf{H}^f)$
GAT
Shared Weights
GAT
$g(\tilde{\mathbf{A}}, \mathbf{H}^s)$
$\hat{\mathbf{H}}^f$
$\hat{\mathbf{H}}^s$
$h(\mathbf{H}^f, \mathbf{H}^s)$
$h(\hat{\mathbf{H}}^f, \hat{\mathbf{H}}^s)$
$\mathbf{H}$
$\hat{\mathbf{H}}$
+
−
−
SSL Loss

Figure 1: Pipeline of FOSSIL for self-supervised node classification tasks. The proposed method integrates encoding, view generation, fusion, subgraph sampling, and CL in attributed graphs. FOSSIL employs shared-weight encoder and generator, with GNNs to extract informative representations, subsequently fused for enhanced learning. Subgraph sampling facilitates subgraph-level CL loss computation based on the FGWD.

In this work, we use the Jensen-Shannon estimator (Nowozin et al., 2016) for our subgraph-level contrastive learning and the InfoNCE (van den Oord et al., 2018) for our node-level contrastive learning.

The learned representation $f(\mathcal{G})$ can then be transferred to a graph-level or node-level downstream task by plugging a carefully designed decoder:

$$\arg\min_{\boldsymbol{\theta}_{ds}} \mathcal{L}_{\text{downstream}}(\text{Dec}(f(\mathcal{G})), \mathcal{Y}), \tag{4}$$

where $\mathcal{Y}$, $\text{Dec}(\cdot)$, $\boldsymbol{\theta}_{ds}$, and $\mathcal{L}_{\text{downstream}}$ are respectively the labels, the decoder (or prediction head), the parameters, and the loss for the downstream task.

4 Fused Gromov-Wasserstein Subgraph Contrastive Learning (FOSSIL)

4.1 Overview

Figure 1 illustrates the pipeline of our proposed method, referred to as FOSSIL. FOSSIL comprises five primary stages: (i) *encoding*, (ii) view *generation*, (iii) *fusion* layer, (iv) subgraph *sampling*, and (v) *SSL loss* computation based on OT for subgraph-level plus a node-level contrastive loss. FOSSIL takes an input graph $\mathcal{G}$ and employs two neural networks with shared weights as encoders. The first neural network performs linear projection, while the second implements a GNN (Kipf & Welling, 2017), both equipped with the same learnable parameters $\mathbf{W}$. Subsequently, view generation follows a decoupled strategy similar to the encoder, utilizing a GAT (Veličković et al., 2018). The fusion layer combines the outputs from the encoder and generator using a linear combination. Finally, subgraphs are sampled at the encoder and generator levels to compute the subgraph-level CL loss using the FGWD as the distance metric and a node-level contrastive loss is additionally computed.

4.2 Encoder

Several GNNs for self-supervised node classification are built under the homophily assumption, meaning that they assume the nodes in the neighborhood share similar labels (Han et al., 2022). As a consequence, they typically perform badly when evaluated on heterophilic graph datasets. To address this issue, we propose a decoupled encoding, where we use a GNN and a linear projection with shared weights $\mathbf{W} \in \mathbb{R}^{F \times D}$. The neural network encoder $f(\mathbf{S}, \mathbf{X})$ is thus given by:

$$\mathbf{H} = \sigma(\mathbf{S}\mathbf{X}\mathbf{W}), \tag{5}$$

where σ is a non-linearity and $\mathbf{S}$ is an operator characterized either by a normalized form of the adjacency matrix or the identity. Next, we elaborate more on the two parts of FOSSIL's encoder, namely embedding features and graph structure.

4.2.1 Feature embedding

We use the identity matrix $\mathbf{I} \in \mathbb{R}^{N \times N}$ as $\mathbf{S}$ in equation 5 to independently encode each node without considering the information from neighboring nodes. In other words, we extract the feature information embedded in the node features. This is formally given by:

$$\mathbf{H}^f = f(\mathbf{I}, \mathbf{X}) = \sigma(\mathbf{IXW}), \tag{6}$$

where we omit the bias term for simplicity. The intuition in equation 6 is to extract embeddings that completely ignore the graph structure. Please notice that by replacing $\mathbf{S}$ by $\mathbf{I}$ in equation 5, this is equivalent to applying a Multilayer Perceptron (MLP) to node features.

4.2.2 Structure embedding

We use the normalized adjacency matrix $\tilde{\mathbf{A}}$ to extract the structural information in the encoder as:

$$\mathbf{H}^s = f(\tilde{\mathbf{A}}, \mathbf{X}) = \sigma(\tilde{\mathbf{A}}\mathbf{XW}), \tag{7}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbb{1}_N) \in \mathbb{R}^{N \times N}$. Here, we derive additional structural insights complementing the feature information extracted in equation 6. Please notice that in equation 7, contrary to GCN (Kipf & Welling, 2016), we use the normalized adjacency matrix without self-loops.

Intuitively, the decoupling strategy proves effective due to the presence of two distinct sources of information in an attributed graph: node features and graph structure. The relevance of each information source can vary depending on the downstream task at hand. For instance, in the context of a social network, individual user features often play a primary role in defining users, while friendship connections offer supplementary insights into their interactions. Conversely, in a citation network, it could be more crucial to examine the documents cited by an article to predict its category effectively. In essence, on homophilic graphs, traditional GNN encoding typically outperforms MLPs, whereas, on heterophilic graphs, MLPs tend to perform better Zhu et al. (2020a). The decoupled approach enables us to better adapt the SSL model to the homophily level of the graph.

4.3 Generator

Generating positive pairs in SSL for graphs is still an open problem of research. A common approach here involves perturbing the original graph to preserve the underlying semantics (You et al., 2020; Qiu et al., 2020). However, determining these perturbations for graphs is not straightforward. State-of-the-art methods usually apply hand-crafted perturbations like random node feature masking and random edge dropping. Such perturbations lack control and can degrade the intrinsic properties of the original graph—hence increasing the risk of false positives (Wang et al., 2022). Moreover, the performance of these hand-crafted perturbations heavily depends on the characteristics of the particular graph dataset. Therefore, an additional step is required to manually identify the appropriate perturbation method tailored to each dataset.

In FOSSIL, we adopt an end-to-end perturbation strategy, meaning that we learn the correct perturbation from the data. We use a similar strategy as in the encoder to separately generate node feature perturbations and graph structure perturbations. Given the feature embeddings $\mathbf{H}^f$ and structure embeddings $\mathbf{H}^s$ from the encoder, we generate perturbations as $\hat{\mathbf{H}}^f = g(\mathbf{I}, \mathbf{H}^f)$, $\hat{\mathbf{H}}^s = g(\tilde{\mathbf{A}}, \mathbf{H}^s)$, where $g : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times D}$ corresponds to a GAT model (Veličković et al., 2018). Intuitively, we leverage the attention weights of the GAT to adaptively generate perturbations. At the feature level, this is equivalent to applying adaptive scaling to the node features, while at the graph structure level, this is equivalent to adaptively dropping some edges.

4.4 Fusion Layer

We combine the feature and structure information from the encoder and generator to obtain the final node embeddings. To this end, we apply adaptive node-wise fusion layers with shared weights $\boldsymbol{\theta}$. This stems from the fact that nodes do not necessarily require the same amount of structural information. At the encoder level, for each node v_i we combine the semantic and structural information as follows:

$$\boldsymbol{\lambda}_i = \psi(\mathbf{H}_i^f, \mathbf{H}_i^s, (\mathbf{A}\mathbb{1}_N)_i; \boldsymbol{\theta}), \tag{8}$$

$$\mathbf{H}_i = h(\mathbf{H}_i^f, \mathbf{H}_i^s) = \mathbf{H}_i^f + \boldsymbol{\lambda}_i \mathbf{H}_i^s, \tag{9}$$

where $\psi : \mathbb{R}^D \times \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}$ is an MLP (with sigmoid as the last activation function) computing how much structural information each node needs, and $\boldsymbol{\lambda} \in [0,1]^N$ is the vector of fusion coefficients of the graph nodes. Similarly, for the generator we have:

$$\hat{\boldsymbol{\lambda}}_i = \psi(\hat{\mathbf{H}}_i^f, \hat{\mathbf{H}}_i^s, (\mathbf{A}\mathbb{1}_N)_i; \theta), \tag{10}$$

$$\hat{\mathbf{H}}_i = h(\hat{\mathbf{H}}_i^f, \hat{\mathbf{H}}_i^s) = \hat{\mathbf{H}}_i^f + \hat{\boldsymbol{\lambda}}_i \hat{\mathbf{H}}_i^s. \tag{11}$$

We incorporate node degrees $\mathbf{A}\mathbb{1}_N$ into the fusion module. The rationale behind this choice is to leverage node degree centralities to account for the relative importance of each node within the entire graph. By doing so, we assist the function $\psi(\cdot)$ in computing fusion coefficients $\boldsymbol{\lambda}$ that are more sensitive to the graph's structure.

4.5 Subgraph Sampling

Using the original node embeddings $\mathbf{H}$ and the perturbed node embeddings $\hat{\mathbf{H}}$, we sample the subgraphs that will be subsequently used as contrastive samples. To this end, we first sample without replacement a set of anchor-node indices $S \subseteq \{1, \dots, N\}$. From each anchor node $v_i \in S$, we sample a set of node indices S_i of size k (including v_i) using breadth-first sampling in the 2-hop neighborhood of v_i. If we do not get the required size k, then the subgraph is not included in the subgraph-level contrast. Therefore, we create two subgraphs using each S_i. The first subgraph is given by $\mathcal{G}_i^s = (\mathbf{A}[S_i; S_i], \mathbf{H}[S_i])$. The second subgraph is given by $\hat{\mathcal{G}}_i^s = (\hat{\mathbf{A}}[S_i; S_i], \hat{\mathbf{H}}[S_i])$, where $\hat{\mathbf{A}}_{i,j} = s(\hat{\mathbf{h}}_i, \hat{\mathbf{h}}_j) = \frac{\hat{\mathbf{h}}_{s_i}^\top \hat{\mathbf{h}}_{s_j}}{\|\hat{\mathbf{h}}_{s_i}\|_2 \|\hat{\mathbf{h}}_{s_j}\|_2}$. These subgraphs are used in our CL loss function using the FGWD. A perturbed subgraph corresponds to the original subgraph contextualized by the GAT we are using for adaptive graph perturbation. The intuition is that this contextualized subgraph further captures the similarities between nodes inside the original subgraph, hence it should be similar to the original subgraph while adding additional context information to the nodes.

4.6 Self-supervised Loss Function

The SSL loss function in FOSSIL is given by two terms: a contrastive loss $\mathcal{L}_{\text{contrast}}$, and a regularization term $\mathcal{L}_\theta$ to guide the learning of the fusion layer. The contrastive loss is at the same time divided into an OT loss $\mathcal{L}_{\text{ot}}$ and a node-level contrastive loss $\mathcal{L}_{\text{node}}$, so that $\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{ot}} + \mathcal{L}_{\text{node}}$. Our final loss is thus given by $\mathcal{L} = \mathcal{L}_{\text{contrast}} + \mathcal{L}_\theta$.

4.6.1 OT distance-based subgraph-level contrastive loss $\mathcal{L}_{\text{ot}}$

In FOSSIL, we consider graphs as measured spaces by endowing them with probability distributions over their nodes. The probability mass on a given node represents the node's relative importance in the graph. With no prior knowledge of the graphs, we assume a uniform probability distribution over its nodes.

Let $\mathcal{G}_1^s = (\mathbf{A}_1^s, \mathbf{H}_1^s)$ and $\mathcal{G}_2^s = (\mathbf{A}_2^s, \mathbf{H}_2^s)$ be two subgraphs. We measure the difference between $\mathcal{G}_1^s$ and $\mathcal{G}_2^s$ using the FGWD as follows:

$$D_{\text{fgw}}(\mathcal{G}_1^s, \mathcal{G}_2^s) = \text{FGWD}_{\alpha, \mathbf{C}^{1,2}, \mathbf{C}^1, \mathbf{C}^2}(\boldsymbol{\mu}, \boldsymbol{\nu}), \tag{12}$$

where μ is the uniform distribution over nodes of $\mathcal{G}_1^s$, ν is the uniform distribution on the nodes of $\mathcal{G}_2^s$, $\mathbf{C}^1 = \exp(-\mathbf{A}_1^s/\tau)$, $\mathbf{C}^2 = \exp(-\mathbf{A}_2^s/\tau)$ and $\mathbf{C}^{1,2} = \exp(-\mathbf{H}_1^s(\mathbf{H}_2^s)^\top/\tau)$ is the matrix of pairwise transportation costs between their nodes (the exponentials on matrices are applied element-wise). We use FGWD to jointly exploit both features and structure information inside the two graphs to compute their distances. In this work, we solve the optimization problem associated with FGWD using the Bregman Alternated Projected Gradient (BAPG) method (Li et al., 2023) given in Alg. 1, App. A. Our subgraph-level contrastive loss can be thus written as:

$$\begin{aligned}\mathcal{L}_{\text{ot}} = -\frac{1}{|S|(M+1)} \sum_{i \in S} \Big[&\log \Big(\sigma(\exp(-D_{\text{fgw}}(\mathcal{G}_i^s, \hat{\mathcal{G}}_i^s)/\tau)) \Big) \\ &+ \sum_{j=1}^{M} \log \big(1 - \sigma(\exp(-D_{\text{fgw}}(\mathcal{G}_i^s, \mathcal{G}_i^{sn_j})/\tau)) \big) \Big].\end{aligned} \tag{13}$$

where S is the set of sampled anchor-node indices defined in Sec. 4.5, $\mathcal{G}_i^s$ is a sampled subgraph and $\mathcal{G}_i^{sn_j}$ is its j-th negative view used in the contrastive loss. For each node, we define M negative views in equation **??**. In practice, we set $M = 2$ and define $\mathcal{G}_i^{sn_1} = \mathcal{G}_j^s$ and $\mathcal{G}_i^{sn_2} = \hat{\mathcal{G}}_j^s$ with j randomly picked in $\{j \in S : j \neq i\}$. In the contrastive loss $\mathcal{L}_{\text{ot}}$, $\mathcal{L}_{\text{align}}$ aims to increase the agreement between positive pairs while $\mathcal{L}_{\text{reg}}$ aims to decrease the agreement between negative pairs.

4.6.2 Node-level contrastive loss $\mathcal{L}_{\text{node}}$

We also incorporate a node-level contrastive loss as follows:

$$l(\mathbf{h}_i, \hat{\mathbf{h}}_i) = -\log \frac{\exp(s(\mathbf{h}_i, \hat{\mathbf{h}}_i)/\tau)}{\sum_{j \neq i} \exp(s(\mathbf{h}_i, \mathbf{h}_j)/\tau) + \sum_{j=1}^{N} \exp(s(\mathbf{h}_i, \hat{\mathbf{h}}_j)/\tau)} \tag{14}$$

$$\mathcal{L}_{\text{node}} = \frac{1}{2N} \sum_{i=1}^{N} \Big(l(\mathbf{h}_i, \hat{\mathbf{h}}_i) + l(\hat{\mathbf{h}}_i, \mathbf{h}_i) \Big). \tag{15}$$

4.6.3 Embedding fusion loss $\mathcal{L}_\theta$

The embedding fusion loss guides the fusion layer learning behavior. The main motivation for this loss is to prevent the structure and feature embeddings from the encoder from being too similar. To this end, when the structure and feature embedding are very similar, we assume that we should emphasize more the feature embedding. Therefore, we promote low values in the vector $\boldsymbol{\lambda}$ for the concerned nodes. We also couple the $\mathcal{L}_\theta$ and the OT loss through the parameter α. Notice that $1-\alpha$ in the definition of the FGWD in equation 1 represents the relative weight of structure similarity when comparing subgraphs with OT. Therefore, we promote the values of $\boldsymbol{\lambda}$ to align with $1-\alpha$ on average, so that $\boldsymbol{\lambda}$ aligns with the OT distances. Additionally, we constrain the variance of $\boldsymbol{\lambda}$ with L_2 regularization to put more control on the way we combine embeddings. Our fusion layer loss function is thus given as:

$$\mathcal{L}_\theta = \sum_{i=1}^{N} \boldsymbol{\lambda}_i s(\mathbf{h}_i^s, \mathbf{h}_i^c) + \beta_1 \|\boldsymbol{\lambda}\|_2 + \beta_2 \left| \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\lambda}_i - (1-\alpha) \right|, \tag{16}$$

where β_1 and β_2 are regularization parameters.

4.7 Computational Complexity of the Framework

4.7.1 Time complexity

Here, we analyze the computational complexity of our subgraph-level contrastive loss in equation 12. To compute FGWD, we use the BAPG method described in Alg. 1 of App. A, which is a single-loop method for solving the non-convex quadratic optimization problem of FGWD. The tensor product $\underline{\mathbf{L}}(\mathbf{C}_1, \mathbf{C}_2) \otimes \mathbf{P}$ can be computed with complexity $\mathcal{O}(n^2 m + nm^2)$. So the gradients computed at lines 4 and 8 of Alg. 1 can be computed with complexity $\mathcal{O}(n^2m + nm^2)$. The other operations of Alg. 1 have complexity $\mathcal{O}(nm)$. Hence, the time complexity of BAPG is $\mathcal{O}(T(n^2m + nm^2))$ overall, where n and m are, respectively, the order of the first and second graph and T is the number of iterations that have been performed. Now we need to compute the cost matrices $\mathbf{M}$, $\mathbf{C}_1$, and $\mathbf{C}_2$ that are inputs of the BAPG algorithm. Those are computed with an overall complexity $\mathcal{O}(nmD)$, where D is the dimension of the embedding vectors. Finally solving the optimization problem in equation 1 with Alg. 1 has complexity $\mathcal{O}(T(n^2m + nm^2) + nmD)$.

At each iteration of our method, we sample $|S|$ subgraphs, each with k nodes (so we are in the case where $n = m = k$). Then for each subgraph, we compute its FWGDs with its positive view and its M negative views, which costs $\mathcal{O}((M+1)(Tk^3 + k^2D))$ in time. Hence, solving all the optimization problems costs $\mathcal{O}(|S|(M+1)(Tk^3 + k^2D))$ in time. Finally, we need to compute the contrastive loss with the FGWDs, which is done in $\mathcal{O}(|S|(M+1))$. Overall the time complexity of computing $\mathcal{O}(|S|(M+1)(Tk^3 + k^2D))$.

It is worth noting that this time complexity only depends on the hyperparameters of our method. In particular, the cubic term k^3 is not a real concern in practice, as k is chosen such that $k \ll N$, with N the number of nodes in the graph. Therefore, the complexity of computing $\mathcal{L}_{\text{ot}}$ does not have scalability issues with the size of the input graph as it remains constant w.r.t. N.

In our framework, we also compute a node-level contrastive loss $\mathcal{L}_{\text{node}}$. This loss is computed with complexity $\mathcal{O}(N^2D)$ theoretically. However, in practice, as computing the pair-wise cosine similarities between nodes inside the graph is parallelized thanks to matrix multiplication on the GPU, the complexity is roughly constant depending on the number of cores and memory available on the GPU.

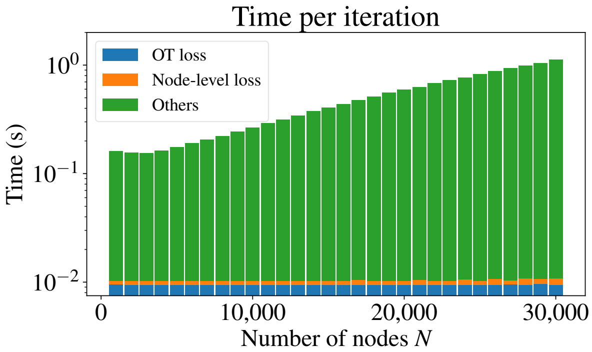

Figure 2: Time per iteration of the different components of FOSSIL in a GPU A40 48G.

To experimentally validate this theoretical analysis, we evaluate the time required for computing each loss with synthetic graphs generated with the contextual stochastic block model (cSBM) Chien et al. (2021). We vary the number of nodes N from $1,000$ to $30,000$, and for each value of N, we measure the times on 5 synthetic graphs with arbitrary homophily levels. We fix $k = 12$, $D = 512$, $|S| = 300$, and $M = 2$ for this experiment. Fig. 2 shows that the time required to compute the OT loss and node-level contrastive losses remains roughly constant for any value of N. Overall, on Fig. 2 we observe that our method scales reasonably well with the size of the graph. The empirical time complexity depicted by the green bars does not significantly increase from a graph with $1,000$ nodes to one with $30,000$ nodes.

4.7.2 Memory complexity

Computing the FGWD with Alg. 1 requires storing all the cost matrices and the intermediate matrices used inside the algorithm. Hence, this entails a memory complexity of $\mathcal{O}(n^2 + m^2 + nm)$ for two graphs of order n and m. In our case, $n = m = k$, and we compute the FGWD of all the $|S|(M+1)$ subgraph pairs together inside the same batch. Therefore, we have a memory complexity of $\mathcal{O}(k^2|S|(M+1))$ for $\mathcal{L}_{ot}$. The memory complexity of the optimal transport component of FOSSIL is thus constant w.r.t. the size of the graph N.

5 Experimental Framework and Results

We perform a set of experiments to compare our model with several approaches in the literature. We compare FOSSIL with nine state-of-the-art methods including: GSC (Han et al., 2022), DGI (Velickovic et al., 2019), DSSL (Xiao et al., 2022), BGRL (Thakoor et al., 2021), GRACE (Zhu et al., 2020b), MUSE (Yuan et al., 2023), Subg-Con (Jiao et al., 2020), PolyGCL Chen et al. (2024), and BNLL Liu et al. (2024). In addition, we compare FOSSIL with GSC using as the encoder the Chebysev graph convolution (Defferrard et al., 2016) (GSC+Cheb) instead of the regular GCN, to make the GSC framework more robust to heterophilic datasets. We evaluate all methods in four homophilic datasets Cora (McCallum et al., 2000), CiteSeer (Sen et al.,

Table 1: Statistics of the datasets used in this work.

	Cora	**CiteSeer**	**PubMed**	**CoAuthor-CS**	**OGBN-Arxiv**	**Actor**	**Chameleon**	**Squirrel**
$H(G)$	0.81	0.74	0.80	0.81	0.66	0.22	0.24	0.22
Nodes	$2,708$	$3,327$	$19,717$	$18,333$	$169,343$	$7,600$	$2,277$	5201
Edges	$10,556$	$9,104$	$88,648$	$163,788$	$1,166,243$	$30,019$	$36,101$	$217,073$
Features	$1,433$	$3,703$	500	6805	128	932	$2,325$	$2,089$
Classes	7	6	3	15	40	5	5	5

2008), PubMed (Namata et al., 2012), and CoAuthor-CS (Shchur et al., 2018); and in three heterophilic datasets Actor (Tang et al., 2009), Chameleon, and Squirrel (Rozemberczki et al., 2021). Table 1 shows the statistics of the datasets tested in this work, where $H(G)$ is the homophily of the graph as defined in (Pei et al., 2020). Finally, we also run a set of ablation studies to analyze different aspects of FOSSIL.

We use a standard downstream logistic regression accuracy with the self-supervised node embeddings to evaluate all compared methods. We split the data into a development set (80%) and a test set (20%) once, ensuring that the test set is not used during the hyperparameter optimization process. Therefore, we split the development set into training and validation sets for hyperparameter tuning. This is done by taking 20 nodes per class for the training set and the rest into the validation set for Cora, CiteSeer, and PubMed. For the other datasets, the training set gets 60% of the nodes and the validation set gets the other 40% in the development set. Finally, we test each method using 100 test seeds to randomly split the dataset into the train and validation, while keeping the original test set. We train and evaluate the encoder on those splits and report the average performances accompanied by 95% confidence intervals calculated by bootstrapping with $1,000$ samples. Unlike other works in SSL for graphs, we do not perform model selection during SSL training by looking at the downstream test/validation performance. Finally, we take the encoder at the end of the SSL training to extract the node embeddings for the downstream evaluation. All experiments are conducted on A40 48GB and P100 16GB GPUs.

5.1 Implementation Details

We implement all methods using PyTorch (Ketkar, 2017) and PyG (Fey & Lenssen, 2019). We optimize the hyperparameters with the framework Optuna (Akiba et al., 2019). The search spaces for hyperparameters are defined as follows: 1) learning rate $lr \in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$; 2) learning rate of fusion module $lr_f \in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$; 3) $\alpha \in \{[0 : 0.1 : 1]\}$; 4) the FGWD regularizer $\beta \in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}, 5\times10^{-1}, 1, 1.5, 2\}$; 5) the number of nodes in each sampled subgraph $k \in [10, 30]$; 6) the temperature parameter $\tau \in \{0.2, 0.5, 0.8, 1.0, 1.5, 2.0, 2.5, 3.0\}$; 7) the GNN dropout parameter $p \in \{0.1, 0.2, 0.3, 0.4\}$, and 8) the fusion MLP dropout $p_f \in \{0.1, 0.2, 0.3, 0.4\}$. The value of β_2 is fixed to 1. We tune the hyperparameters by performing 100 trials in the development set. Finally, for each trial, we sample the values from the search space using the Tree-structured Parzen estimator (Watanabe, 2023). The source code will be made publicly available upon publication[1].

5.2 Results and Discussion

Table 2 shows the results of the comparison of FOSSIL against previous state-of-the-art models for self-supervised node classification. We evaluate all methods with our setting to have a fair comparison. Our encoder is a GCN without self-loops; thus, we also evaluate GCN in a fully supervised setting, which we refer to as FS-GCN. We also evaluate a fully-supervised MLP (FS-MLP) for reference.

In Table 2, we observe that FOSSIL outperforms FS-MLP and FS-GCN, suggesting that pre-trained models using self-supervision and then fine-tuned on a downstream task can outperform end-to-end supervised model. In general, FOSSIL ranks either as the best or second best method among the baseline models used while being the best on average across both homophilic and heterophilic datasets. We also observe that FOSSIL is better than MUSE and GRACE on average, showing that node-level only contrast is suboptimal compared to including subgraph-level contrast for node classification.

[1] https://anonymous.4open.science/r/FOSSIL-72CE

Table 2: Test accuracy on the node classification task for different baseline models and FOSSIL. The best results are highlighted in **bold**, while the second best-performing methods are underlined.

Method	Cora	CiteSeer	PubMed	CoAuthor CS	Actor	Chameleon	Squirrel	*Overall*
FS-GCN	80.04$_{\pm 0.26}$	69.05$_{\pm 0.26}$	77.33$_{\pm 0.41}$	92.50$_{\pm 0.02}$	29.69$_{\pm 0.14}$	45.48$_{\pm 0.25}$	30.26$_{\pm 0.18}$	60.62$_{\pm 0.22}$
FS-MLP	57.28$_{\pm 0.39}$	56.37$_{\pm 0.37}$	69.20$_{\pm 0.43}$	92.05$_{\pm 0.06}$	30.08$_{\pm 0.20}$	48.68$_{\pm 0.33}$	37.14$_{\pm 0.16}$	55.83$_{\pm 0.28}$
GSC	79.29$_{\pm 0.20}$	54.56$_{\pm 0.74}$	76.98$_{\pm 0.50}$	90.16$_{\pm 0.06}$	28.43$_{\pm 0.16}$	42.90$_{\pm 0.40}$	28.98$_{\pm 0.22}$	57, 33$_{\pm 0.33}$
CSC+Cheb	76.84$_{\pm 0.26}$	68.11$_{\pm 0.26}$	75.75$_{\pm 0.45}$	89.20$_{\pm 0.14}$	32.46$_{\pm 0.13}$	43.99$_{\pm 0.34}$	35.03$_{\pm 0.24}$	47.45$_{\pm 0.26}$
MUSE	62.52$_{\pm 0.38}$	58.05$_{\pm 0.42}$	79.08$_{\pm 0.31}$	54.32$_{\pm 1.76}$	**36.61**$_{\pm 0.15}$	52.11$_{\pm 0.50}$	33.57$_{\pm 0.17}$	53.75$_{\pm 0.53}$
DGI	79.90$_{\pm 0.23}$	70.22$_{\pm 0.32}$	77.80$_{\pm 0.47}$	87.38$_{\pm 0.06}$	29.11$_{\pm 0.14}$	36.24$_{\pm 0.24}$	29.58$_{\pm 0.16}$	58.60$_{\pm 0.23}$
DSSL	72.39$_{\pm 0.56}$	54.16$_{\pm 0.64}$	69.98$_{\pm 0.69}$	87.73$_{\pm 0.08}$	26.60$_{\pm 0.14}$	48.38$_{\pm 0.35}$	34.99$_{\pm 0.21}$	56.32$_{\pm 0.38}$
BGRL	62.95$_{\pm 0.87}$	38.02$_{\pm 1.46}$	72.93$_{\pm 0.44}$	91.81$_{\pm 0.22}$	27.93$_{\pm 0.17}$	36.90$_{\pm 0.49}$	28.91$_{\pm 0.25}$	51.35$_{\pm 0.56}$
GRACE	80.86$_{\pm 0.29}$	65.86$_{\pm 0.40}$	**79.84**$_{\pm 0.33}$	93.01$_{\pm 0.03}$	28.91$_{\pm 0.18}$	43.87$_{\pm 0.34}$	30.60$_{\pm 0.22}$	60.42$_{\pm 0.26}$
Subg-Con	44.32$_{\pm 0.02}$	30.07$_{\pm 0.91}$	67.35$_{\pm 0.85}$	82.29$_{\pm 0.54}$	27.60$_{\pm 0.18}$	45.60$_{\pm 0.44}$	**37.16**$_{\pm 0.43}$	47.77$_{\pm 0.48}$
PolyGCL	**80.95**$_{\pm 0.26}$	**71.38**$_{\pm 0.21}$	76.39$_{\pm 0.41}$	OOM	26.52$_{\pm 0.07}$	32.75$_{\pm 0.19}$	34.25$_{\pm 0.13}$	–
BNLL	66.46$_{\pm 0.83}$	41.57$_{\pm 1.17}$	70.45$_{\pm 0.92}$	93.60$_{\pm 0.03}$	25.33$_{\pm 0.34}$	50.18$_{\pm 0.90}$	36.63$_{\pm 0.50}$	54.89$_{\pm 0.67}$
FOSSIL	80.02$_{\pm 0.26}$	68.24$_{\pm 0.34}$	79.76$_{\pm 0.48}$	**94.49**$_{\pm 0.04}$	35.61$_{\pm 0.15}$	**53.03**$_{\pm 0.33}$	37.13$_{\pm 0.24}$	**64.04**$_{\pm 0.26}$

Table 3: Impact of subgraph similarity metric in node classification.

Metric	Cora	CiteSeer	PubMed	CoAuthor CS	Actor	Chameleon	Squirrel	*Overall*
WD	61.82$_{\pm 0.44}$	59.06$_{\pm 0.39}$	71.94$_{\pm 0.68}$	91.22$_{\pm 1.83}$	**35.64**$_{\pm 0.18}$	44.20$_{\pm 0.36}$	33.09$_{\pm 0.26}$	56, 71$_{\pm 0.59}$
GWD	19.51$_{\pm 0.82}$	22.33$_{\pm 0.34}$	39.44$_{\pm 0.65}$	64.93$_{\pm 2.42}$	34.71$_{\pm 0.16}$	26.50$_{\pm 1.63}$	24.27$_{\pm 1.44}$	33.10$_{\pm 1.07}$
Readout+Cosim	66.04$_{\pm 0.54}$	61.32$_{\pm 0.44}$	70.13$_{\pm 0.63}$	84.33$_{\pm 0.28}$	34.27$_{\pm 0.17}$	42.43$_{\pm 0.29}$	33.35$_{\pm 0.29}$	55.98$_{\pm 0.38}$
FGWD	**80.02**$_{\pm 0.26}$	**68.24**$_{\pm 0.34}$	**79.76**$_{\pm 0.48}$	**94.49**$_{\pm 0.04}$	35.61$_{\pm 0.15}$	**53.03**$_{\pm 0.33}$	**37.13**$_{\pm 0.24}$	**64.04**$_{\pm 0.26}$

It is important to highlight further that FOSSIL is better than DGI and Subg-Con, which use a readout function to characterize a graph. This suggests that a simple readout function does not properly capture the structure of the graph. Besides, these methods perform node-graph contrast. Maximizing the agreement of an anchor node and its sampled subgraph is not optimal for node classification since this subgraph can contain a majority of dissimilar nodes (particularly in heterophilic graphs)—hence its readout representation will be a false positive.

FOSSIL works better than the previous OT-based method GSC, showing that a joint feature-structure subgraph similarity metric altogether with a decoupled embedding extraction has more potential to benefit from the structural patterns inside a graph. Therefore, FOSSIL has the ability to induce more relevant perturbations hence a more accurate contrastive training, and to adapt to varying levels of homophily.

5.3 Ablation Studies

We validate the effectiveness of each component of FOSSIL through a series of ablation studies. First, we validate the use of the FGWD that captures both feature and graph structure similarities between subgraphs. Then, we investigate whether the decoupling strategy is necessary for self-supervised node classification. Finally, we explore other strategies to generate graph perturbations within our framework.

5.3.1 Impact of decoupled node embeddings

We replace the FGWD with other subgraph similarity metrics to evaluate its effectiveness. We replace FGWD with other OT distances like WD and GWD. Similarly, we also use a simple average pooling of the subgraphs' node features followed by a cosine similarity (referred to as Readout+Cosim). Table 3 shows the results of this ablation study in all datasets. We first notice that node features are more important than pure graph connectivity for node classification tasks. Indeed, GWD, which relies solely on subgraph structure similarity, performs very poorly compared to WD and Readout+Cosim. Secondly, we observe that WD performs better than Readout+Cosim. This suggests that OT distances are better for self-supervised node classification than simple readout functions. Nevertheless, both WD and Readout+Cosim, ignore the underlying graph structure. Finally, we observe the superior performance of using the FGWD that jointly captures the feature and structure similarities between subgraphs.

Table 4: Ablation study on the encoder method.

Encoder	Cora	CiteSeer	PubMed	CoAuthor CS	Actor	Chameleon	Squirrel	*Overall*
Standard GCN	74.72$_{\pm0.49}$	61.35$_{\pm0.45}$	74.11$_{\pm0.60}$	90.12$_{\pm0.06}$	13.54$_{\pm0.94}$	48.68$_{\pm0.52}$	25.34$_{\pm0.28}$	55.41$_{\pm0.48}$
Decoupled GCN	**80.02**$_{\pm0.26}$	**68.24**$_{\pm0.34}$	**79.76**$_{\pm0.48}$	**94.49**$_{\pm0.04}$	**35.61**$_{\pm0.15}$	**53.03**$_{\pm0.33}$	**37.13**$_{\pm0.24}$	**64.04**$_{\pm0.26}$

Table 5: Graph perturbation ablation study results on node classification.

Generator	Cora	CiteSeer	PubMed	CoAuthor CS	Actor	Chameleon	Squirrel	*Overall*
Random	78.28$_{\pm0.35}$	65.85$_{\pm0.39}$	76.54$_{\pm0.42}$	89.40$_{\pm0.27}$	28.95$_{\pm1.58}$	**57.29**$_{\pm1.34}$	38.22$_{\pm0.96}$	62.08$_{\pm0.76}$
GCN	76.95$_{\pm0.32}$	67.70$_{\pm0.34}$	79.44$_{\pm0.50}$	93.49$_{\pm1.93}$	34.33$_{\pm0.62}$	46.83$_{\pm0.77}$	**39.43**$_{\pm0.23}$	62.60$_{\pm0.67}$
GAT	**80.02**$_{\pm0.26}$	**68.24**$_{\pm0.34}$	**79.76**$_{\pm0.48}$	**94.49**$_{\pm0.04}$	**35.61**$_{\pm0.15}$	53.03$_{\pm0.33}$	37.13$_{\pm0.24}$	**64.04**$_{\pm0.26}$

5.3.2 Impact of decoupled node embeddings

We analyze the effectiveness of our encoder by replacing it with a simple GCN. As depicted in Table 4, the decoupled approach outperforms the standard GCN in the heterophilic datasets. We also note a significant improvement in the performance of the homophilic datasets with the decoupled approach, emphasizing its importance in the FOSSIL framework.

5.3.3 Impact of the view generation process

We analyze the design of the generator of FOSSIL by replacing the GAT component with a GCN. We also replace GAT with random feature masking ($\sim 10\%$) and random edge dropping ($\sim 15\%$) to assess the benefit of a learnable view generation (denoted as Random). Table 5 shows the results of this study. First, we notice that both the GCN and GAT models outperform the random perturbation. Even though random perturbation presents competitive results, it requires careful fine-tuning of the hyperparameters. Finally, we observe that the attention weights of the GAT provide better views than a GCN. On the contrary, since the weights of a GCN are rather fixed, they do not adapt to the dynamics of contrastive learning to provide better views.

Table 6: Comparison between FOSSIL and FOSSILv2.

Method	Cora	CiteSeer	PubMed	CoAuthor CS	OGBN-Arxiv	Actor	Chameleon	Squirrel
FOSSIL	80.02$_{\pm0.26}$	68.24$_{\pm0.34}$	79.76$_{\pm0.48}$	94.49$_{\pm0.04}$	OOM	35.61$_{\pm0.15}$	53.03$_{\pm0.33}$	37.13$_{\pm0.24}$
FOSSILv2	78.94$_{\pm0.26}$	68.16$_{\pm0.23}$	79.09$_{\pm0.39}$	95.04$_{\pm0.03}$	56.22$_{\pm0.12}$	35.41$_{\pm0.18}$	53.14$_{\pm1.24}$	34.79$_{\pm0.21}$

6 Limitations

Our method presents a memory bottleneck that comes from the node-level contrastive loss $\mathcal{L}_{\text{node}}$ that requires storing a $N \times N$ matrix for the pairwise cosine similarities of all the nodes of the graph, hence $\mathcal{O}(N^2)$ of memory complexity. This limits the scalability of FOSSIL to large-scale graphs. To circumvent this issue, we propose a straightforward solution called FOSSILv2. This method only considers the union of nodes inside the sampled subgraphs rather than all nodes of the graph in $\mathcal{L}_{\text{node}}$. Therefore, the new node-level contrastive loss becomes:

$$\mathcal{L}_{\text{node-v2}} = \frac{1}{2\left|\bigcup_{j\in S} S_j\right|} \sum_{i\in\bigcup_{j\in S} S_j} \Big(l(\mathbf{h}_i, \hat{\mathbf{h}}_i) + l(\hat{\mathbf{h}}_i, \mathbf{h}_i) \Big) . \tag{17}$$

Therefore, we have a memory complexity of $\mathcal{O}((|S|k)^2D)$ for FOSSILv2. Results in Table 6 show that FOSSILv2 can be applied on the large-scale OGBN-Arxiv graph (Hu et al., 2020a), with a limited decrease in performance in the other datasets.

7 Conclusion

In this paper, we introduced a novel subgraph-based contrastive learning framework utilizing the Fused Gromov-Wasserstein distance to measure subgraph similarities. Drawing inspiration from recent advancements in addressing heterophily with GNNs, we implemented a decoupled embedding extraction strategy, rendering our method agnostic to the graph's homophily level. This approach allowed us to independently generate feature and structure perturbations adaptively. Our extensive experimental evaluation demonstrated that our method outperforms or has competitive performance against state-of-the-art frameworks across multiple benchmark datasets with varying levels of homophily. Our findings also highlight the challenges posed by heterophilic datasets. We suggest future research avenues, such as integrating a high-pass feature extractor to enhance performance in such scenarios.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Alexei Baevski, Henry Zhou, Abdel rahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, 2020.

Luc Brogat-Motte, Rémi Flamary, Céline Brouard, Juho Rousu, and Florence d'Alché Buc. Learning to predict graphs with fused Gromov-Wasserstein barycenters. In *International Conference on Machine Learning*, 2022.

Jingyu Chen, Runlin Lei, and Zhewei Wei. PolyGCL: Graph contrastive learning via learnable spectral polynomial filters. In *International Conference on Learning Representations*, 2024.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *International Conference on Learning Representations*, 2021.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations - Workshops*, 2019.

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

Yuehui Han, Le Hui, Haobo Jiang, Jianjun Qian, and Jin Xie. Generative subgraph contrast for self-supervised graph representation learning. In *European Conference on Computer Vision*, 2022.

Mingguo He, Zhewei Wei, and Ji-Rong Wen. Convolutional neural networks on graphs with Chebyshev approximation, revisited. In *Advances in Neural Information Processing Systems*, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: Generative pre-training of graph neural networks. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020b.

Elvin Isufi, Fernando Gama, David I Shuman, and Santiago Segarra. Graph filters for signal processing and machine learning on graphs. *IEEE Transactions on Signal Processing*, 2024.

Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. *IEEE International Conference on Data Mining*, 2020.

Nikhil S. Ketkar. Deep learning with Python. In *Apress*, 2017.

Thomas Kipf and Max Welling. Variational graph auto-encoders. In *Advances in Neural Information Processing Systems - Workshops*, 2016.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Association for Computational Linguistics*, 2019.

Jiajin Li, Jianheng Tang, Lemin Kong, Huikang Liu, Jia Li, Anthony Man-Cho So, and José H. Blanchet. A convergent single-loop algorithm for relaxation of Gromov-Wasserstein in graph data. In *International Conference on Learning Representations*, 2023.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21:105–117, 1988.

Yunhui Liu, Huaisong Zhang, Tieke He, Tao Zheng, and Jianhua Zhao. Bootstrap latents of nodes and neighbors for graph self-supervised learning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2024.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiaoming Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances Neural Information Processing Systems*, 2022.

Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Facundo Mémoli. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11:417–487, 2011.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, 2012.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *ArXiv*, abs/1606.00709, 2016.

Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *ArXiv*, abs/1905.09550, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. volume 29, pp. 93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Advances Neural Information Processing Systems - Workshops*, 2018.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Velivckovi'c, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2021.

Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised ResNets: Can we outperform supervised learning without labels on ImageNet? In *International Conference on Machine Learning - Workshops*, 2022.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Titouan Vayer, Nicolas Courty, Romain Tavenard, Laetitia Chapel, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, 2018.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Lio', Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

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

Haonan Wang, Jieyu Zhang, Qi Zhu, and Weitao Huang. Augmentation-free graph contrastive learning. *ArXiv*, abs/2204.04874, 2022.

Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *ArXiv*, abs/2304.11127, 2023.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24, 2021.

Teng Xiao, Zhengyu Chen, Zhimeng Guo, Zeyang Zhuang, and Suhang Wang. Decoupled self-supervised learning for non-homophilous graphs. In *Advances in Neural Information Processing Systems*, 2022.

Yaochen Xie, Zhao Xu, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:2412–2429, 2021.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, 2020.

Mengyi Yuan, Minjie Chen, and Xiangci Li. MUSE: Multi-view contrastive learning for heterophilic graphs. *ACM International Conference on Information and Knowledge Management*, 2023.

Xuan Zang, Xianbing Zhao, and Buzhou Tang. Hierarchical molecular graph self-supervised learning for property prediction. *Communications Chemistry*, 6(1):34, 2023.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: current limitations and effective designs. In *Advances in Neural Information Processing Systems*, 2020a.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *International Conference on Machine Learning - Workshops*, 2020b.

A Algorithm for FGWD

Algorithm 1 Bregman Alternated Projected Gradient (BAPG) for FGWD

```
$\mathbf{P}^{(0)} \leftarrow \boldsymbol{\mu}\boldsymbol{\nu}^\top$
$\Delta\mathbf{P} \leftarrow \infty$
$i \leftarrow 1$
while $(i < T)$ And $(\Delta\mathbf{P} > \epsilon)$ do
  // Row updating
  $\mathbf{G} \leftarrow \alpha\mathbf{M} + 2(1-\alpha)\underline{\mathbf{L}}(\mathbf{C}_1, \mathbf{C}_2) \otimes \mathbf{P}^{(i-1)}$ // Gradient of the objective function of Eq. (1) in $\mathbf{P}^{(i-1)}$
  $\tilde{\mathbf{P}} \leftarrow \mathbf{P}^{(i-1)} \odot \exp(-\mathbf{G}/\beta)$
  $\tilde{\mathbf{P}} \leftarrow \mathrm{diag}(\frac{\mu}{\tilde{\mathbf{P}}\mathbb{1}_m})\tilde{\mathbf{P}}$

  // Column updating
  $\mathbf{G} \leftarrow \alpha\mathbf{M} + 2(1-\alpha)\underline{\mathbf{L}}(\mathbf{C}_1, \mathbf{C}_2) \otimes \tilde{\mathbf{P}}$ // Gradient of the objective function of Eq. (1) in $\tilde{\mathbf{P}}$
  $\mathbf{P}^{(i)} \leftarrow \tilde{\mathbf{P}} \odot \exp(-\mathbf{G}/\beta)$
  $\mathbf{P}^{(i)} \leftarrow \mathbf{P}^{(i)}\,\mathrm{diag}(\frac{\mu}{\mathbf{P}^{(i)\top}\mathbb{1}_n})$
  $\Delta\mathbf{P} \leftarrow \|\mathbf{P}^{(i)} - \mathbf{P}^{(i-1)}\|$
  $i \leftarrow i + 1$
end while
```

B Sensibility Analysis

One of the most important hyperparameters of FOSSIL is α, which dictates how much our model should focus on each source of information of the input graph, namely node features and graph structure. To assess how sensitive our framework is to the value of α, we vary its value and see how the performance changes. Specifically for each real-world dataset in our experiments, we fix the other hyperparameters and vary α within the set $\{[0 : 0.1 : 1]\}$. For each value of α, we evaluate our method on 5 seeds and report the mean and standard deviation. Fig. 3 shows the sensibility analysis results. We notice that the performance of our method can significantly change depending on the value of α. Overall for homophilic graphs, the best value of α tends to be close to 0.5. This suggests that node features and graph structure are equally important for the encoder to characterize the graph. However, for heterophilic graphs, the best value of α tends to be close to 1. A plausible explanation is that even though the graph structure is important, the GCN encoder not being able to correctly exploit the structure of heterophilic graphs due to its intrinsic homophily assumption, our framework will rely more on node features. Hence, our encoder will converge to an MLP with a value of α close to 1. This further supports our idea to include a feature extractor that is more suited to heterophilic graphs for future research.

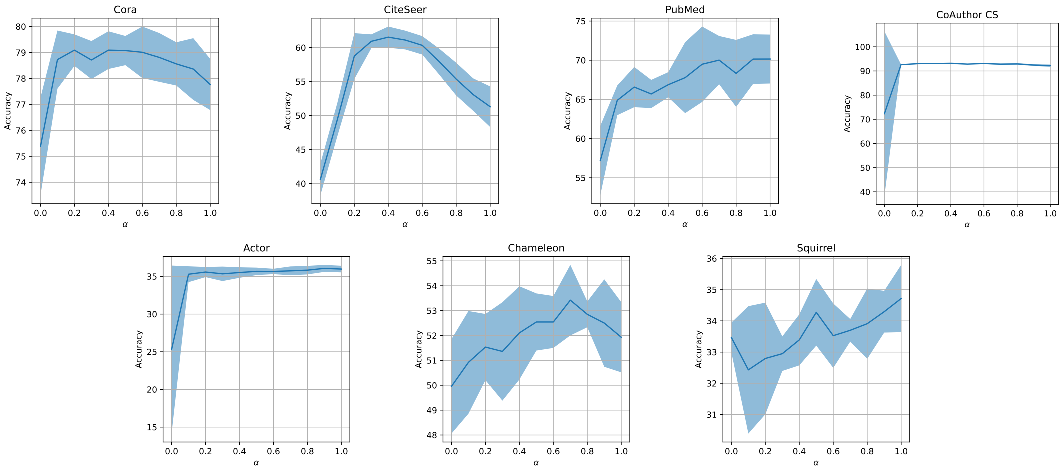

Figure 3: Variation of the accuracy for several datasets w.r.t. α.