Graph Homomorphism Convolution

Hoang NT 1,2  Takanori Maehara 1

Abstract

In this paper, we study the graph classification problem from the graph homomorphism perspective. We consider the homomorphisms from $F$ to $G$, where $G$ is a graph of interest (e.g., molecules or social networks) and $F$ belongs to some family of graphs (e.g., paths or non-isomorphic trees). We prove that graph homomorphism numbers provide a natural universally invariant (isomorphism invariant) embedding maps which can be used for graph classifications. In practice, by choosing $F$ to have bounded tree-width, we show that the homomorphism method is not only competitive in classification accuracy but also run much faster than other state-of-the-art methods. Finally, based on our theoretical analysis, we propose the Graph Homomorphism Convolution module which has promising performance in the graph classification task.

1. Introduction

1.1. Background

In many fields of science, objects of interest often exhibit irregular structures. For example, in biology or chemistry, molecules and protein interactions are often modeled as graphs (Milo et al., 2002; Benson et al., 2016). In multi-physics numerical analyses, methods such as the finite element methods discretize the sample under study by 2D/3D-meshes (Mezentsev, 2004; Fey et al., 2018). In social studies, interactions between people are presented as a social network (Barabási et al., 2016). Understanding these irregular non-Euclidean structures have yielded valuable scientific and engineering insights. With recent successful developments of machine learning on regular Euclidean data such as images, a natural extension challenge arises: How do we learn non-Euclidean data such as graphs or meshes?

Geometric (deep) learning (Bronstein et al., 2017) is an important extension of machine learning as it generalizes learning methods from Euclidean data to non-Euclidean data. This branch of machine learning not only deals with learning irregular data but also provides a proper mean to combine meta-data with their underlying structure. Therefore, geometric learning methods have enabled the application of machine learning to real-world problems: From categorizing complex social interactions to generating new chemical molecules. Among these methods, graph-learning models for the classification task have been the most important subject of study.

Let $\mathcal{X}$ be the space of features (e.g., $\mathcal{X} = \mathbb{R}^d$ for some positive integer $d$), $\mathcal{Y}$ be the space of outcomes (e.g., $\mathcal{Y} = \{0, 1\}$), and $G = (V(G), E(G))$ be a graph with a vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. The graph classification problem is stated follow$^1$.

Problem 1 (Graph Classification Problem). We are given a set of tuples $\{(G_i, x_i, y_i) : i = 1, \ldots, N\}$ of graphs $G_i = (V(G_i), E(G_i))$, vertex features $x_i : V(G_i) \to \mathcal{X}$, and outcomes $y_i \in \mathcal{Y}$. The task is to learn a hypothesis $^2$ $h$ such that $h((G_i, x_i)) \approx y_i$.

Problem 1 and its solutions have been studied both theoretically and empirically. Regarding the theoretical graph classification models, universality properties to some targeted classes of function are often discussed. While we can identify the function classes which these theoretical models can approximate, practical implementations pose many challenges. For instance, the tensorized model proposed by (Keriven & Peyré, 2019) is universal in the space of continuous functions on bounded size graphs, but it is impractical to implement such model. On the other hand, little is known about the class of function which can be estimated by some practical state-of-the-art$^3$ models. To address these disadvantages of both theoretical models and practical models, we need a practical graph classification model whose approximation capability can be parameterized. Such model is not only effective in practice as we can introduce inductive bias to the design by the aforementioned parameterization but also useful in theory as a framework to study the graph classification problem.

---

1RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
2Tokyo Institute of Technology, Tokyo, Japan.
3On benchmark datasets such as the TU Dortmund datasets

Work in progress. Copyright 2020 by the author(s).

---

1This setting also includes the regression problem.
2In practice, $h$ can be a machine learning model.
3On benchmark datasets such as the TU Dortmund datasets
We found that using homomorphism numbers as an invariant we are often interested in a hypothesis $h$. A common approach to Problem 1 is to design an embedding map for the graph classification problem? Question 2. How to design an efficient and invariant embedding map for the graph classification problem?

1.2. Homomorphism Numbers as a Classifier

A common approach to Problem 1 is to design an embedding $\rho: (G, x) \rightarrow \rho((G, x)) \in \mathbb{R}^p$, which maps graphs to vectors, where $p$ is the dimensionality of the representation. Such an embedding can be used to represent a hypothesis for graphs as $h((G, x)) = g(\rho((G, x)))$ by some hypothesis $g: \mathbb{R}^p \rightarrow \mathcal{Y}$ for vectors. Because the learning problem on vectors is a well-studied problem, we can focus on designing and understanding the graph embedding.

We found that using homomorphism numbers as an invariant embedding is not only theoretically valid but also extremely efficient in practice. In a nutshell, the embedding for a graph $G$ is given by selecting $k$ pattern graphs to form a fixed set $\mathcal{F}$, then computing the homomorphism numbers from each $F \in \mathcal{F}$ to $G$. The classification capability of the homomorphism embedding is parameterized by $\mathcal{F}$. We develop rigorous analyses for this idea in Section 2 (without vertex features) and Section 3 (with vertex features).

Our contribution is summarized as follows:

- Introduce and analyze the usage of weighted graph homomorphism numbers with a general choice of $\mathcal{F}$. The choice of $\mathcal{F}$ is a novel way to parameterize the capability of graph learning models compared to choosing the tensorization order in other related works.

- Prove the universality of our methods. Our proof technique is simpler than existing works by unifying under the Stone-Weierstrass theorem.

- Empirically demonstrate our theoretical findings with synthetic and benchmark datasets. Notably, we show that our methods perform well in settings where other methods fail.

In this paper, we focus on the simple undirected graphs without edge weights for simplicity. The extension of all our results to directed and/or weighted graphs is left as the future work.

1.3. Related Works

There are two main approaches to construct an embedding. The first one is the graph kernels and the second one is the graph neural networks. We introduce some of the most popular methods which directly related to our work. For a more comprehensive view of the literature, we refer to the surveys on graph neural networks (Wu et al., 2019) and graph kernels (Gärtner, 2003; Kriege et al., 2019).

1.3.1. Graph Kernels

The kernel method first defines a kernel function on the space, which implicitly defines an embedding $\rho$ such that the inner product of the embedding vectors gives a kernel function. The advantage of this approach is that the computational complexity is independent of the dimensionality of the embedding vector. Hence, we can learn a complicated model as a linear model in a high-dimensional (possibly infinite-dimensional) space. The disadvantage of this approach is that the computational complexity typically depends quadratically on the number of samples.

The graph kernel method is the most popular approach to study graph embedding maps. Since designing a kernel which uniquely represents graphs up to isomorphisms is as hard as solving graph isomorphism (Gärtner et al., 2003), many previous studies on graph kernels have focused on proposing a solution to the trade-off between computational efficiency and representability. A natural idea is to compute subgraph frequencies (Gärtner et al., 2003) to use as graph embeddings. However, the subgraph counting problem is a #W[1]-hard problem (Flum & Grohe, 2006) and even counting induced subgraphs is an NP-hard problem (more precisely it is an #A[1]-hard problem (Flum & Grohe, 2006)). Therefore, methods like the tree kernel (Collins & Duffy, 2002) or the random walk kernel (Gärtner et al., 2003; Borgwardt et al., 2005) restrict the subgraph family to be some computationally efficient graphs. Interestingly, (Gärtner et al., 2003) also mentioned a relaxation which is similar to homomorphism counting. However, they only studied the walk kernel.

More recently, the graphlet kernel (Shervashidze et al., 2009; Pržulj et al., 2004) and the Weisfeiler-Lehman kernel (Shervashidze et al., 2011; Kriege et al., 2016) set the state-of-the-art for benchmark datasets (Kersting et al., 2016). Other similar kernels with novel modifications to the distance function

\[ 4 \text{Not to be confused with “vertex embedding”} \]
are also proposed. For instance, the Wasserstein Weisfeiler-Lehman kernel (Togninalli et al., 2019) combined ideas from the transportation theory with the Weisfeiler-Lehman algorithm to create a graph kernel which works well with continuous vertex features. While these kernels are effective for benchmark datasets, some are known to be not universal (Xu et al., 2019; Keriven & Peyré, 2019) and it is difficult to know which class of graphs they would fail to classify.

1.3.2. Graph Neural Networks

Graph Neural Networks refers to a new class of graph classification model in which the embedding map \( \rho \) is implemented by a neural network. In general, the mapping \( \rho \) follows an aggregation-readout scheme (Hamilton et al., 2017; Gilmer et al., 2017; Xu et al., 2019; Du et al., 2019) where vertex features are aggregated from their neighbors and then read-out to obtain the graph embedding. Empirically, especially on social network datasets, these neural networks have shown better accuracy and inference time than graph kernels. However, there exist some challenging cases where these practical neural networks fail such as the Circular Skip Links synthetic data (Murphy et al., 2019) or bipartite classification (Section 4).

Theoretical analysis of graph neural networks is an active topic of study. The capability of a graph neural network has been recently linked to the Weisfeiler-Lehman isomorphism test (Morris et al., 2019; Xu et al., 2019). Since (Morris et al., 2019) and (Xu et al., 2019) proved that the aggregation-readout scheme is bounded by the one-dimensional Weisfeiler-Lehman test, much work has been done to quantify and improve the capability of graph neural networks via the tensorization order. Another important aspect of graph neural networks is their property to approximate equivariant or invariant functions for some symmetry group (Maron et al., 2018; 2019; Keriven & Peyré, 2019). Interestingly, isomorphism testing and function approximation are equivalent (Chen et al., 2019).

The advantage of tensorized graph neural networks lies in their expressible power. However, the disadvantage is that the tensorization order makes it difficult to have an intuitive view of the functions which need to be approximated. Furthermore, the empirical performance of these models might heavily depend on initialization (Chen et al., 2019).

2. Graphs without Features

We first establish our theoretical framework for graphs without vertex features. This type of feature-less graphs are often social network data in which only structure (hyperlinks, friendships, etc.) is captured. The main result of this section is to show that using the homomorphism numbers with some polynomial not only yield a universal invariant approximator but we can also select the pattern set \( \mathcal{F} \) for some targeted application.

2.1. Definition

An (undirected) graph \( G = (V(G), E(G)) \) is simple if it has neither self-loops nor parallel edges. We denote by \( \mathcal{G} \) the set of all simple graphs.

Let \( G \) be a graph. For a finite set \( U \) and a bijection \( \sigma : V(G) \to U \), we denote by \( G^\sigma \) the graph defined by \( V(G^\sigma) = U \) and \( E(G^\sigma) = \{(\sigma(u), \sigma(v)) : (u, v) \in E(G)\} \). Two graphs \( G_1 \) and \( G_2 \) are isomorphic if \( G_1^\sigma = G_2 \) for some bijection \( \sigma : V(G_1) \to V(G_2) \).

2.2. Homomorphism Numbers

Here, we introduce the homomorphism number. This is a well-studied concept in graph theory (Hell & Nesetril, 2004; Lovász, 2012) and plays a key role in our framework. Let \( F \) and \( G \) be undirected graphs. A homomorphism from \( F \) to \( G \) is a function \( \pi : V(F) \to V(G) \) that preserves the existence of edges, i.e., \( (u, v) \in E(F) \) implies \( (\pi(u), \pi(v)) \in E(G) \). We denote by \( \text{Hom}(F, G) \) the set of all homomorphisms from \( F \) to \( G \).

The homomorphism number \( \text{hom}(F, G) \) is the cardinality of the homomorphisms, i.e., \( \text{hom}(F, G) = |\text{Hom}(F, G)| \).

We consider the homomorphism density \( t(F, G) \). This is a normalized version of the homomorphism number:

\[
t(F, G) = \frac{\text{hom}(F, G)}{|V(G)|^{|V(F)|}} = \sum_{\pi: V(F) \to V(G)} \prod_{u \in V(F)} \frac{1}{|V(G)|} \times \prod_{(u, v) \in E(F)} 1[(\pi(u), \pi(v)) \in E(G)],
\]

where \([\cdot]\) is the Iverson bracket such that \( 1[P] = 1 \) if the condition \( P \) is true. Eq. (2) can be seen as the probability that randomly sampled \( |V(F)| \) vertices of \( V(G) \) preserves the edges of \( E(F) \). Intuitively, a homomorphism number \( \text{hom}(F, G) \) aggregates local connectivity information of \( G \) using a pattern graph \( F \).

Example 3. Let \( \circ \) be a single vertex, we have \( \text{hom}(\circ, G) = |V(G)| \) and \( \text{hom}(\circ \circ, G) = 2|V(E)| \).

Example 4. Let \( S_k \) be the star graph of size \( k + 1 \). Then, \( \text{hom}(S_k, G) \propto \sum_{v \in V(G)} d(v)^k \), where \( d(v) \) is the degree of vertex \( v \).

Example 5. We have: \( \text{hom}(C_k, G) \propto \text{tr}(A^k) \), where \( C_k \) is a length \( k \) cycle and \( A \) is the adjacency matrix of \( G \).

It is trivial to see that the homomorphism number is invariant under isomorphism. Surprisingly, the converse holds as the
homomorphism numbers identify the isomorphism class of a graph. Formally, we have the following theorem.

**Theorem 6** ([Lovász, 1967]). Two graphs $G_1$ and $G_2$ are isomorphic if and only if $\text{hom}(F,G_1) = \text{hom}(F,G_2)$ for all simple graphs $F$. In addition, if $|V(G_1)|, |V(G_2)| \leq n$ then we only have to examine $F$ with $|V(F)| \leq n$.

### 2.3. Homomorphism Numbers as Embeddings

The invariance of the homomorphism numbers motivates us to use them as the embedding vectors for a graph. Because examining all graphs will be impractical (i.e. $F = \mathcal{G}$), we select a subset $F \subseteq \mathcal{G}$ as a parameter for the graph embedding. We obtain the embedding vector of a graph $G$ by stacking the the homomorphism numbers from $F \in F$:

$$\text{hom}(F,G) = [\text{hom}(F,G) : F \in F] \in \mathbb{R}^{|F|}.$$  

We evaluate this embedding both theoretically and experimentally. The experimental evaluation will be presented in Section 4; thus, we only consider a theoretical evaluation here. We focus on the following two criteria: Expressive capability and computational efficiency. Similar to the trade-off between kernel representability and efficiency, a more expressive homomorphism embedding map is usually less efficient and vice versa.

Two graphs $G_1$ and $G_2$ are $\mathcal{F}$-indistinguishable if $\text{hom}(F,G_1) = \text{hom}(F,G_2)$ for all $F \in \mathcal{F}$ (Böker et al., 2019). Theorem 6 implies that the $\mathcal{F}$-indistinguishability generalizes graph isomorphism. For several classes $\mathcal{F}$, the interpretation of $\mathcal{F}$-indistinguishability is studied; the results are summarized in Table 1. The most interesting result is the case when $\mathcal{F}$ is the trees of size at most $k$ where $\mathcal{F}$-indistinguishability coincides with the $k$-dimensional Weisfeiler–Lehman isomorphism test (Dell et al., 2018).

A function $f : \mathcal{G} \rightarrow \mathbb{R}$ is $\mathcal{F}$-invariant if $f(G_1) = f(G_2)$ for all $\mathcal{F}$-indistinguishable $G_1$ and $G_2$; therefore, if we use the $\mathcal{F}$-homomorphism as an embedding, we can only represent $\mathcal{F}$-invariant functions. In practice, $\mathcal{F}$ should be chosen as small as possible such that the target hypothesis can be assumed to be $\mathcal{F}$-invariant.

In the next section, we show that any continuous $\mathcal{F}$-invariant function is arbitrary accurately approximated by a function of the $\mathcal{F}$-homomorphism embedding.

### 2.4. Expressive Power: Universality Theorem

We first characterize the class of functions that is represented by $\mathcal{F}$-homomorphism numbers. We obtain the following two results.

**Theorem 7.** Let $f$ be an $\mathcal{F}$-invariant function. For any positive integer $N$, there exists a degree $N$ polynomial $h_N$ of $\text{hom}(F,G)$ such that $f(G) \approx h_N(G)$ for all $G$ with $|V(G)| \leq N$.

| $\mathcal{F}$ | $\mathcal{F}$-indistinguishable |
|---------------|-------------------------------|
| single vertex | graphs have the same number of vertices (Example 3) |
| single edge   | graphs have the same number of edges (Example 3) |
| all stars     | graphs have the same degree sequence (Example 4) |
| all cycles    | adjacency matrices have the same eigenvalues (Example 5) |
| all graphs of treewidth $k$ | graphs cannot be distinguished by the $k$-dimensional Weisfeiler–Lehman test (Dell et al., 2018) |
| all simple graphs | isomorphic graphs (Lovász, 1967) |

**Theorem 8.** Let $f$ be a continuous $\mathcal{F}$-invariant function. There exists a degree $N$ polynomial $h_N$ of $\text{hom}(F,G)$ ($F \in \mathcal{F}$) such that $f(G) \approx h_N(G)$ for all $G$.

Theorem 7 is the universal approximation theorem for bounded size graphs. This holds without any assumption of the target function $f$. It is worth mentioning that the invariant/equivariant universality results of tensorized graph neural networks on this bounded size setting were proven by (Keriven & Peyré, 2019); the unbounded case remains an open problem. Theorem 8 is the universal approximation for all graphs (unbounded). This is an improvement to the previous works. However, our theorem only holds for continuous functions, where the topology of the space has to satisfy the conditions of the Stone-Weierstrass theorem.

Let $X$ be a set of points (e.g., graphs). A set of functions $A$ separates $X$ if for any two different points $G_1, G_2 \in X$, there exists a function $h \in A$ such that $h(G_1) \neq h(G_2)$.

**Theorem 9** (Stone–Weierstrass Theorem (Hart et al., 2003)). Let $X$ be a compact Hausdorff space and $C(X)$ be the set of continuous functions from $X$ to $\mathbb{R}$. If a subset $A \subseteq C(X)$ separates $X$ then the set of polynomials of $A$ is dense in $C(X)$ w.r.t. the topology of uniform convergence.

**Proof of Theorem 7** First, we consider the case that the number of vertices of the input graphs is bounded by $N$. Hence, the graph space contains a finite number of points; therefore, under the discrete topology, the space is compact Hausdorff. The separability follows from the definition. Therefore, by applying the Stone–Weierstrass theorem (Theorem 9), we conclude the proof.

Next, we consider the situation when the number of the
vertices in the input graph is unbounded. In this case, the input space contains infinitely many graphs; therefore, it is not compact under the discrete topology. Hence, we cannot directly apply the Stone–Weierstrass theorem as in the bounded case.

To obtain a stronger result, we have to complete the set of all graphs, and prove the completed space is compact Hausdorff. Since it is non-trivial to work directly with discrete graphs here, we find that the graphons theory (Lovász, 2012) fits our purpose.

A sequence of graphs $G_1, G_2, \ldots$ is a convergence if the homomorphism density, $t(F, G_i)$, is a convergence for all simple graph $F$. A limit of a convergence is called a graphon, and the space obtained by adding the limits of the convergences is called the graphon space, which is denoted by $\mathcal{G}$. See (Lovász, 2012) for the detail of this construction. The following theorem is one of the most important results in graphon theory.

**Theorem 10** (Compactness Theorem (Lovász, 2012; Lovász & Szegedy, 2006)). The graphon space $\mathcal{G}$ with the cut distance $\delta_{\mathcal{G}}$ is compact Hausdorff.

Now we can prove a graphon version of Theorem 8.

**Theorem 11.** Any continuous $\mathcal{F}$-invariant function $f: \mathcal{G} \to \mathbb{R}$ is arbitrarily accurately approximated by a polynomial of $\{t(F, \cdot): F \in \mathcal{F}\}$.

**Proof.** The $\mathcal{F}$-indistinguishability forms a closed equivalence relation on $\mathcal{G}$, where the homomorphism density is used instead of the homomorphism number. Let $\mathcal{G}/\mathcal{F}$ be the quotient space of this equivalence relation, which is compact Hausdorff in the quotient topology.

By the definition of the quotient topology, any continuous $\mathcal{F}$-invariant function is identified as a continuous function on $\mathcal{G}/\mathcal{F}$. Also, by the definition, the set of $\mathcal{F}$-homomorphisms separates the quotient space. Therefore, the conditions of the Stone–Weierstrass theorem (Theorem 9) are fulfilled.

### 2.5. Computational Complexity: Bounded Treewidth

Computing homomorphism numbers is, in general, an $\text{NP}$-hard problem (Díaz et al., 2002). However, if the pattern graph $F$ has bounded treewidth, homomorphism numbers can be computed in polynomial time.

A tree-decomposition (Robertson & Seymour, 1986) of a graph $F$ is a tree $T = (V(T), E(T))$ with mapping $B: V(T) \to 2^{V(F)}$ such that (1) $\bigcup_{t \in V(T)} B(t) = V(F)$, (2) for any $(u, v) \in E(F)$ there exists $t \in V(T)$ such that $(u, v) \subseteq B(t)$, and (3) for any $u \in V(F)$ the set $\{t \in V(T) : u \in B(t)\}$ is connected in $T$. The treewidth (abbreviated as “tw”) of $F$ is the minimum of $\max_{t \in V(T)} |B(t)| - 1|$ for all tree-decomposition $T$ of $F$.

### Algorithm 1 Compute $\text{hom}(F, (G, x))$

**Input:** target graph $G$, pattern graph $F$, vertex features $x$

**function** recursion(current, visited)

\[ \text{hom}_x \leftarrow x \]

for $y$ in $F$.neighbors(current) do

if $y \neq$ visited then

\[ \text{hom}_y \leftarrow \text{recursion}(y, \text{current}) \]

aux $\leftarrow \left\lfloor \sum \text{hom}_y[F.\text{neighbors}(i) \text{ for } i \in V(G)] \right\rfloor$

\[ \text{hom}_x \leftarrow \text{hom}_x \ast \text{aux} \text{ (element-wise mult.)} \]

end if

end for

**return** $\text{hom}_x$

**Output:** $\sum \text{recursion}(0, -1)$

### Theorem 12 ((Díaz et al., 2002)). For any graphs $F$ and $G$, the homomorphism number $\text{hom}(F, G)$ is computable in $O(|V(G)|^{tw(F) + 1})$ time.

The most useful case will be when $F$ is the set of trees of size at most $k$. The number of trees of size $k$ is a known integer sequence\(^6\). There are 106 non-isomorphic trees of size $k = 10$, which is computationally tractable in practice. Also, in this case, the algorithm for computing $\text{hom}(F, G)$ is easily implemented by dynamic programming with recursion as in Algorithm 1. This algorithm runs in $O(|V(G)| + |E(G)|)$ time. For the non-featured case, we sets $x_r(u) = 1 \forall u \in V(G)$. The simplicity of Algorithm 1 comes from the fact that if $F$ is a tree then we only need to keep track of a vertex’s immediate ancestor when we process that vertex by the visited argument in the function recursion.

### 3. Graphs with Features

Many datasets contain metadata such as vertex features. This type of data are often biological/chemical data and studied extensively in the graph kernel literature (Kersting et al., 2016). To extend our framework to featured data, we develop theoretical results for graphs with features.

#### 3.1. Definition

A vertex-featured graph is a pair $(G, x)$ of a graph $G$ and a function $x: V(G) \to \mathcal{X}$, where $\mathcal{X} = [0, 1]^p$.

Let $(G, x)$ be a vertex-featured graph. For a finite set $U$ and a bijection $\sigma: V(G) \to U$, we denote by $x^\sigma$ the feature vector on $G^\sigma$ such that $x^\sigma(\sigma(u)) = x(u)$. Two vertex-featured graphs $(G_1, x_1)$ and $(G_2, x_2)$ are isomorphic if $G_1^\sigma = G_2$ and $x_1^\sigma = x_2$ for some bijection $\sigma: V(G_1) \to V(G_2)$.

\(^6\text{https://oeis.org/A000055} \)
3.2. Weighted Homomorphism Numbers

We first consider the case that the features are non-negative real numbers. To clarify this setting, we denote by $x(u)$ the feature of vertex $u$. The weighted homomorphism number is defined as

$$\text{hom}(F,(G,x)) = \sum_{\pi \in \text{Hom}(F,G)} \prod_{u \in V(F)} x(\pi(u)), \tag{3}$$

and the weighted homomorphism density is defined by $t(F,(G,x)) = \text{hom}(F,(G,x))$, where $x = x(u)/\sum_{v \in V(G)} x(v)$. This definition coincides with the homomorphism number and density if $x(u) = 1$ for all $u \in V(G)$.

The weighted version of the Lovász theorem holds as follows. We say that two vertices $u, v \in V(G)$ are twins if the neighborhood of $u$ and $v$ are the same. The twin-reduction is a procedure that iteratively selects twins $u$ and $v$, contract them to create new vertex $uv$, and assign $x(uv) = x(u) + x(v)$ as a new weight. Note that the result of the process is independent of the choice of the order of the contraction.

**Theorem 13** (Freedman et al., 2007), (Cai & Govorov, 2019). Two graphs $(G_1, x_1)$ and $(G_2, x_2)$ are isomorphic after the twin-reduction and removing vertices of weight zero if and only if $\text{hom}(F,(G_1, x_1)) = \text{hom}(F,(G_2, x_2))$ for all simple graph $F$.

3.3. $(F, \phi)$-Homomorphism Number

Now we propose a method for the general feature case. Our idea is to reduce the high dimensional features into one-dimensional features by a function and use the weighted homomorphism numbers. Let $\phi: \mathbb{R}^p \rightarrow \mathbb{R}_{\geq 0}$ be a function. For a simple graph $F$, we define the $(F, \phi)$-homomorphism $\text{hom}(F, \phi, (G, x))$ as the weighted homomorphism $\text{hom}(F, (G, \phi(x)))$, i.e.,

$$\text{hom}(F, \phi, (G, x)) = \sum_{\pi \in \text{Hom}(F,G)} \prod_{u \in V(F)} \phi(x(\pi(u))),$$

and the $(F, \phi)$-homomorphism density by $t(F, \phi, (G, x)) = t(F, (G, \phi(x)))$. We prove the following generalization of the Lovász theorem.

**Theorem 14.** Two graphs $(G_1, x_1)$ and $(G_2, x_2)$ are isomorphic if and only if $\text{hom}(F, \phi, (G_1, x_1)) = \text{hom}(F, \phi, (G_2, x_2))$ for all simple graph $F$ and some continuous function $\phi$.

**Proof.** It is trivial to see that if $(G_1, x_1)$ and $(G_2, x_2)$ are isomorphic then they produce the same homomorphism numbers. Thus, we only have to prove the only-if part.

Suppose that the graphs are non-isomorphic. By setting $\phi = 1$, we have the same setting as the feature-less case; hence, by Theorem 6, we can detect the isomorphism classes of the underlying graphs.

Assuming $G_1$ and $G_2$ are isomorphic, we arrange the vertices of $V(G_1)$ in the increasing order of the features (compared with the lexicographical order). Then, we arrange the vertices of $V(G_2)$ lexicographically smallest while the corresponding subgraphs induced by some first vertices are isomorphic. Let us choose the first vertex $u \in V(G_1)$ whose feature is different to the feature of the corresponding vertex in $V(G_2)$. Then, we define

$$\phi(z) = \begin{cases} 1, & z \leq \text{lex} x_1(u), \\ 0, & \text{otherwise}, \end{cases}$$

where $\leq \text{lex}$ stands for the lexicographical order. Then, we have $\text{hom}(F,\phi, (G_1, x_1)) \neq \text{hom}(F,\phi, (G_2, x_2))$ as follows. Suppose that the equality holds. Then, by Theorem 13, the subgraphs induced by vertices whose features are lexicographically smaller than or equal to $x_1(u)$ are isomorphic. However, this contradicts the minimality of the ordering of $V(G_2)$. Finally, by taking a continuous approximation of $\phi$, we obtain the theorem.

3.4. $(F, \phi)$-Homomorphism Number as Embedding

Let $\Phi$ be a set of continuous functions. As same as the featureless case, we propose to use the $(F, \Phi)$-homomorphism numbers as an embedding.

We say that two featured graphs $(G_1, x_1)$ and $(G_2, x_2)$ are $(F, \Phi)$-indistinguishable if $\text{hom}(F, \phi, (G_1, x_1)) = \text{hom}(F, \phi, (G_2, x_2))$ for all $F \in \mathcal{F}$ and $\phi \in \Phi$. A function $f$ is $(F, \Phi)$-invariant if $f(G_1, x_1) = f(G_2, x_2)$ for all $(F, \Phi)$-indistinguishable $(G_1, x_1)$ and $(G_2, x_2)$.

3.5. Universality Theorem

The challenge in proving the universality theorem for the featured setting is similar to the featureless case, which is the difficulty of the topological space. We consider the quotient space of graphs with respect to $(F, \Phi)$-indistinguishability. Our goal is to prove this space is completed to a compact Hausdorff space.

With a slight abuse of notation, consider a function $\ell$ that maps a vertex featured graph $(G, x)$ to a $|\Phi|$-dimensional vector $[(G, \phi(x)) : \phi \in \Phi] \in (G/\mathcal{F})^\Phi$ where each coordinate is an equivalence class of $\mathcal{F}$-indistinguishable graphs. This space has a bijection to the quotient space by $(F, \Phi)$-indistinguishability.

Each coordinate of the $|\Phi|$-dimensional space is completed to a compact Hausdorff space (Borgs et al., 2008). Therefore, by the Tychonoff product theorem (Hart et al., 2003), the $|\Phi|$-dimensional space is compact. The bijection between the quotient space shows the quotient space is com-
Theorem 15. Any continuous \((\mathcal{F}, \Phi)-\)invariant function \(\mathcal{G} \to \mathbb{R}\) is arbitrary accurately approximated by a polynomial of \((G, x) \mapsto t(F_i(G, \phi(x))).\)

Proof. The space \(\mathcal{G}\) is compact by construction. The separability follows from the definition of \((\mathcal{F}, \Phi)-\)invariant. Therefore, by the Stone–Weierstrass theorem, we complete the proof. 

4. Experimental results

| Methods          | CSL  | BIPARTITE | PAULUS25 |
|------------------|------|-----------|----------|
| **Practical models** |      |           |          |
| GIN              | 10.00 ± 0.00 | 55.75 ± 7.91 | 71.14 ± 0.00 |
| GNTK             | 10.00 ± 0.00 | 58.03 ± 6.84 | 71.14 ± 0.00 |
| **Theory models** |      |           |          |
| Ring-GNN         | 10.00 ± 0.00 | 52.68 ± 7.15 | 71.14 ± 0.00 |
| GHC-Cycle       | 100.00 ± 0.00 | 100.00 ± 0.00 | 71.14 ± 0.00 |

(b) Benchmark datasets

| Methods          | MUTAG | IMDB-BIN | IMDB-MUL |
|------------------|-------|----------|----------|
| **Practical models** |      |          |          |
| GNTK             | 89.46 ± 7.03 | 75.61 ± 3.98 | 51.91 ± 3.56 |
| GIN              | 89.40 ± 5.60 | 70.70 ± 1.10 | 43.20 ± 2.00 |
| PATCHY-SAN       | 89.92 ± 4.50 | 71.00 ± 2.20 | 45.20 ± 2.80 |
| WL kernel        | 90.40 ± 5.70 | 73.80 ± 3.90 | 50.90 ± 3.80 |
| **Theory models** |      |          |          |
| Ring-GNN         | -     | 73.00 ± 5.40 | 48.20 ± 2.70 |
| GHC-Tree         | 89.28 ± 8.26 | 72.10 ± 2.62 | 48.60 ± 4.40 |
| GHC-Cycles       | 87.81 ± 7.46 | 70.93 ± 4.54 | 47.41 ± 3.67 |

4.1. Classification models

The realization of our ideas in Section 2 and Section 3 are called Graph Homomorphism Convolution (GHC-* models) due to their resemblance to the \(\mathcal{R}\)–convolution (Hausdorff, 1999). Here, we give specific formulations for two practical embedding maps: GHC-Tree and GHC-Cycle. These embedding maps are then used to train a classifier (Support Vector Machine). We report the 10-folds cross-validation accuracy scores and standard deviations in Table 2.

GHC-Tree We let \(\mathcal{F}_{\text{tree}(6)}\) to be all simple trees of size at most \(k = 6\). Algorithm 1 implements Equation 3 for this case. Given \(G\) and vertex features \(x\), the \(i\)-th dimension of the embedding vector is

\[
\text{GHC-Tree}(G)_i = \text{hom}(\mathcal{F}_{\text{tree}(6)}[i], (G, x)).
\]

GHC-Cycle We let \(\mathcal{F}_{\text{cycle}(8)}\) to be all simple cycles of size at most \(k = 8\). This variant of GHC cannot distinguish iso-spectral graphs. The \(i\)-th dimension of the embedding vector is

\[
\text{GHC-Cycle}(G)_i = \text{hom}(\mathcal{F}_{\text{cycle}(8)}[i], G).
\]

With this configuration, GHC-Tree\((G)\) has 13 dimensions and GHC-Cycle\((G)\) has 7 dimensions.

Other methods To compare our performance with other approaches, we selected some representative methods. GIN (Xu et al., 2019) and PATCHY-SAN (Niepert et al., 2016) are representative of neural-based methods. WL-kernel (Shervashidze et al., 2011) is a widely used efficient method for graph classifications. GNTK (Du et al., 2019) is a recent neural tangent approach to graph classification. We also include results for Ring-GNN (Chen et al., 2019) as this recent model used in theoretical studies performed well in the Circular Skip Links synthetic dataset (Murphy et al., 2019). Except for setting the number of epochs for GIN to be 50, we use the default hyperparameters provided by the original papers. More details for hyperparameters tuning and source code is available in the Supplementary Materials.

4.2. Synthetic Experiments

Bipartite classification We generate a binary classification problem consisting of 200 graphs, half of which are random bipartite graphs with density \(p = 0.2\) and the other half are Erdős-Rényi graphs with density \(p = 0.1\). These graphs have from 40 to 100 vertices. According to Table 1, GHC-Cycle should work well in this case while GHC-Tree cannot learn which graph is bipartite. More interestingly, as shown in Table 2, other practical models also cannot work with this simple classification problem due to their capability limitation (1-WL).

Circular Skip Links We adapt the synthetic dataset used by (Murphy et al., 2019) and (Chen et al., 2019) to demonstrate another case where GIN, Relational Pooling (Murphy et al., 2019), and Order 2 G-invariant (Maron et al., 2018) do not perform well. Circular Skip Links (CSL) graphs are undirected regular graphs with the same degree sequence (4’s). There are 150 graphs with 10 isomorphic groups used as labels in this dataset. Since these graphs are not cospectral, GHC-Cycle can easily learn them with 100% accuracy. In their paper, (Chen et al., 2019) mentioned that the performance of GNN models could vary due to randomness (accuracies ranging from 10% to 80%). However, it is not the case for GHC-Cycle. CSL classification results shows another benefit of using \(F\) patterns as an inductive bias to implement a strong classifier without the need of additional features like Ring-GNN-SVD (Chen et al., 2019).
Paulus graphs We prepare 14 non-isomorphic cospectral strongly regular graphs known as the Paulus graphs\(^7\) and create a dataset of 210 graphs belonging to 14 isomorphic groups. This is a hard example because these graphs have exactly the same degree sequence and spectrum. In our experiments, no method achieves accuracy higher than random guesses (7.14%). We believe further studies of this case could be fruitful to understand and improve our models.

4.3. Benchmark Experiments

We select 3 datasets from the TU Dortmund data collection (Kersting et al., 2016): MUTAG dataset (Debnath et al., 1991), IMDB-BINARY, and IMDB-MULTI (Yanardag & Vishwanathan, 2015). These datasets correspond with and without vertex features graph classification settings. We run and record the 10-folds cross-validation score for each experiment. We report the average accuracy and standard deviation of 10 experiments in Table 2. More experiments on other datasets in the TU Dortmund data collection, as well as the detail of each dataset, are provided in the Appendix.

4.4. Running time

Even though homomorphism counting is \#P-complete in general, polynomial and linear time algorithms exist under the bounded tree-width condition (Díaz et al., 2002). We show in Figure 1 that our method runs much faster than other practical models. The results are recorded from averaging total runtime in seconds for 10 experiments, each computes the 10-folds cross-validation accuracy score. Since the deviation of the runtime is very small, we only plot the mean value. Note that our GHC models only run on a single thread while others run on GPU and multiple threads. In principle, GHC can be linearly distributed to multiple processes to further reduce the computational time making it an ideal baseline model for future studies.

5. Conclusion

In this work we propose the study of graph classification models and datasets from the graph homomorphism perspective. We contribute an alternative approach to the question of quantifying a graph classification model’s capability beyond the tensorization order and the Weisfeiler-Lehman isomorphism test. In principle, tensorized graph neural networks can implement homomorphism numbers, hence our work is in coherence with prior works. However, we find that the homomorphism from \(F\) to \(G\) is a more “fine-grained” tool to analyze graph classification problems as studying \(F\) would be more intuitive (and graph-specific) than studying the tensorization order.

Interestingly, since GHC is a more restricted embedding compared to tensorized graph neural networks such as the model proposed by (Keriven & Peyré, 2019), the universality result of GHC can be translated to the universality result of any other model that has the capability to implement the homomorphism numbers.

Another note on the universality is about the proof for Theorem 8 (universality on unbounded graphs). In order to prove this result, we made an assumption about the topology of \(f\) and also about the graph of interest belongs to the graphon space. While the graphon space is natural in our application to prove the universality, there are a few concerns. First, we assumed that the graphons exist for graphs of interest. However, it might not be true in general. Second, graph limit theory is well-studied in dense graphs while sparse graph problems remain largely open.

In practice, GHC can be used to be a fast baseline for the graph classification problem. Furthermore, GHC with different choices of \(F\) (preferably with bounded treewidth) can be used for other studies beyond classification.

Future work Since the graph homomorphism number is a well-studied concept in graph theory, connecting results in graph theory and machine learning theory via graph homomorphism is a promising direction. For instance, a comprehensive analysis of the components of \(F\) is needed to further categorize the capability of the mapping \(\text{hom}(F, G)\). This question could be connected to the open question in (Chen et al., 2019) about the elements of the algebra. Such analysis will not only lead to a better understanding of graph classification and model designs but also contribute to the explainability of graph neural networks, which is of growing interest recently (Ying et al., 2019).

In practice, Paulus graphs dataset remains challenging for ours and state-of-the-art methods; identifying if there exists

\(^7\)https://www.distanceregular.org/graphs/paulus25.html
a computationally efficient set $\mathcal{F}$ to solve the Paulus classification (and cospectral classification in general) is an open problem we leave for the future. Another interesting practical application of our idea here is to use $(\mathcal{F}, \phi)$ in a neural network. Since $\phi$ is an arbitrary function, it is interesting to model $\phi$ as a neural network. Our Algorithm 1 provides a good starting point for such implementation with $\text{hom}_K$ set as a neural output.

**References**

Barabási, A.-L. et al. *Network science*. Cambridge university press, 2016.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295): 163–166, 2016.

Böker, J., Chen, Y., Grohe, M., and Rattan, G. The complexity of homomorphism indistinguishability. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS’19)*, 2019.

Borgs, C., Chayes, J. T., Lovász, L., Sós, V. T., and Vesztergombi, K. Convergent sequences of dense graphs i: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6):1801–1851, 2008.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Cai, J.-Y. and Govorov, A. On a theorem of lovász that $\text{hom}(\cdot, h)$ determines the isomorphism type of $h$. *arXiv preprint arXiv:1909.03695*, 2019.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gns. In *Advances in Neural Information Processing Systems*, pp. 15868–15876, 2019.

Collins, M. and Duffy, N. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems*, pp. 625–632, 2002.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 (2):786–797, 1991.

Dell, H., Grohe, M., and Rattan, G. Lovász meets weisfeiler and leman. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP’18)*, 2018.

Díaz, J., Serna, M., and Thilikos, D. M. Counting h-colorings of partial k-trees. *Theoretical Computer Science*, 281(1-2):291–309, 2002.

Du, S. S., Hou, K., Póczos, B., Salakhutdinov, R., Wang, R., and Xu, K. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *CoRR*, abs/1905.13192, 2019.

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Flum, J. and Grohe, M. *Parameterized Complexity Theory*. Springer, 2006.

Freedman, M., Lovász, L., and Schrijver, A. Reflection positivity, rank connectivity, and homomorphism of graphs. *Journal of the American Mathematical Society*, 20(1): 37–51, 2007.

Gärtner, T. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

Gärtner, T., Flach, P., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1263–1272. JMLR, 2017.

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

Hart, K. P., Nagata, J.-i., and Vaughan, J. E. *Encyclopedia of general topology*. Elsevier, 2003.

Hausdorff, D. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California, 1999.

Hell, P. and Nesetril, J. *Graphs and homomorphisms*. Oxford University Press, 2004.

Keriven, N. and Peyré, G. Universal invariant and equivariant graph neural networks. *arXiv preprint arXiv:1905.04943*, 2019.
Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. Benchmark data sets for graph kernels, 2016. URL http://graphkernels.cs.tu-dortmund.de.

Kriege, N. M., Giscard, P.-L., and Wilson, R. On valid optimal assignment kernels and applications to graph classification. In Advances in Neural Information Processing Systems, pp. 1623–1631, 2016.

Kriege, N. M., Johansson, F. D., and Morris, C. A survey on graph kernels. arXiv preprint arXiv:1903.11835, 2019.

Lovász, L. Operations with structures. Acta Mathematica Hungarica, 18(3-4):321–328, 1967.

Lovász, L. Large networks and graph limits, volume 60. American Mathematical Soc., 2012.

Lovász, L. and Szegedy, B. Limits of dense graph sequences. Journal of Combinatorial Theory, Series B, 96(6):933–957, 2006.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. arXiv preprint arXiv:1812.09902, 2018.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. arXiv preprint arXiv:1901.09342, 2019.

Mezentsev, A. A. A generalized graph-theoretic mesh optimization model. In IMR, pp. 255–264, 2004.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: simple building blocks of complex networks. Science, 298(5594):824–827, 2002.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 4602–4609, 2019.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In Chaudhuri, K. and Salakhutdinov, R. (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 4663–4673, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In International conference on machine learning, pp. 2014–2023, 2016.

Pržulj, N., Corneil, D. G., and Jurisica, I. Modeling interactome: scale-free or geometric? Bioinformatics, 20(18):3508–3515, 2004.

Robertson, N. and Seymour, P. D. Graph minors. ii. algorithmic aspects of tree-width. Journal of algorithms, 7 (3):309–322, 1986.

Sannai, A., Takai, Y., and Cordonnier, M. Universal approximations of permutation invariant/equivariant functions by deep neural networks. arXiv preprint arXiv:1903.01939, 2019.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In Artificial Intelligence and Statistics, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. Journal of Machine Learning Research, 12(Sep):2539–2561, 2011.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., and Borgwardt, K. Wasserstein weisfeiler-lehman graph kernels. arXiv preprint arXiv:1906.01277, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? International Conference on Learning Representations, 2019.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1365–1374, 2015.

Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnn explainer: A tool for post-hoc explanation of graph neural networks. arXiv preprint arXiv:1903.03894, 2019.
APPENDIX

We present additional information for the theoretical framework (Section 2 and Section 3) and experimental settings (Section 4) here. The Appendix is organized as follow:

- Section A gives details of the configurations for GHC-\(\star\) and other GNNs.
- Section B gives details of synthetic and real-world datasets used in this paper. We also provide some additional results on other real-world datasets.
- Section C provides additional proofs to support our results for GHC.

A. Implementation Details

The source code for GHC is provided with this supplementary document. The main implementation is in file `homomorphism.py`. Aside from Algorithm 1, which can be implemented directly with numpy and run with networkx, other types of homomorphism counting are implemented with C++ and called from `homomorphism.py`. The implementation for general homomorphism is called `homlib`. We include an instruction to install homlib in its README.md. All our experiments are run on a PC with the following specifications. Kernel: 5.3.11-arch1-1; CPU: Intel i7-8700K (12) 4.7 GHz; GPU: NVIDIA GEFORCE GTX 1080 Ti 11GB; Memory: 64 GB. Note that GPU is only used for training GIN (Xu et al., 2019).

Benchmark Experiments The main file to run experiment on real-world (benchmark) datasets is `tud.py`. This is a simple classification problem where each graph in a dataset belongs to a single class. While any other classifier can be used with GHC, we provide the implementation only for Support Vector Machines (scikit-learn) with One-Versus-All multi-class algorithm. We preprocess the data using the StandardScaler provided with scikit-learn. As described in the main part of this paper, we report the best 10-folds cross validation accuracy scores across different SVM configurations. The parameter settings for this experiments are:

- Homomorphism types: Tree, LabelTree (weighted homomorphism), and Cycle.
- Homomorphism size: 6 for trees and 8 for cycles. Theoretically, the homomorphism size increase implies performance increase, but in practice we observe no improvement in classification accuracy beyond size 6. The number of non-isomorphic trees of size \(k\) is presented in Table 3.
- SVC kernel: Radial Basis Function, Polynomial (max degree = 3).
- SVC regularization parameter (C): 20 values in the log-space from \(10^{-2}\) to \(10^{5}\).
- SVC kernel coefficient (gamma): ‘scale’ (1 / \((n\text{\_features} * X\text{\_var()})\))

For other GNNs, we use the default hyperparameters used by the original paper. To run GIN (Xu et al., 2019), we fix the number of epochs at 100 and enable the use degree as tags by default for all datasets. We limit the number of threads used by GNTK (Du et al., 2019) to 8.

Synthetic Experiments The main file to run experiment on real-world (benchmark) datasets is `synthetic.py` and the implementation of synthetic datasets can be found in `utils.py`. Since these experiments focus on the capability of GHC, we can achieve the best performance with just a simple classifier. The parameter settings for this experiments are:

- Homomorphism types: Tree and Cycle.
- Homomorphism size: 6 for trees and 8 for cycles.
- SVC kernel: Radial Basis Function.
- SVC regularization parameter (C): Fix at 1.0.
- SVC kernel coefficient (gamma): Fix at 1.0.

We provide in our source code helper functions which behave the same as the default dataloader used by GIN and GNTK implementations. The external loaders are provided in `externals.py`. Users can copy-paste (or import) these loader into the repository provided by GIN and GNTK to run our synthetic experiments. The settings for other models are set as in the benchmark experiments.

Timing Experiments We measure run-time using the `time` module provided with Python 3.7. The reported time in Figure 1 is the total run-time (in seconds) including homomorphism time and prediction time for our model as well as kernel learning time and prediction time for others.

B. Datasets

As other works in the literature, we use the TU Dortmund data collections (Kersting et al., 2016). The overview of

\[\text{Table 3. The number of non-isomorphic trees of size } k.\]

| \(k\) | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
|------|----|----|----|----|----|----|----|
| # trees | 1  | 1  | 2  | 3  | 6  | 11 | 23 |

| \(k\) | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|------|----|----|----|----|----|----|----|
| # trees | 47 | 106| 235| 551| 1301| 3159| 7741|

\[\text{Table 3. The number of non-isomorphic trees of size } k.\]
C. Homomorphism Convolution

Let \((G, x)\) be a vertex-featured graph. For a simple graph \(F\) and a function \(\phi: \mathbb{R}^p \to \mathbb{R}\), we define \((F, \phi)\)-convolution by

\[
\text{hom}(F, G, x; \phi) = \sum_{\pi \in \text{Hom}(F, G)} \prod_{u \in V(G)} \phi(x(u)).
\]  

(4)

The \((F, \phi)\)-convolution first transform the vertex features into real values by the encoding function \(\phi\). Then this aggregates the values by the pattern graph \(F\). The aggregation part has some similarity with the convolution in CNNs. Thus, we call this operation “convolution.”

Example 16. Let \(\circ\) be a singleton graph and \(\phi\) be the \(i\)-th component of the argument. Then,

\[
\text{hom}(F, G, x; \phi) = \sum_{u \in V(G)} x_i(u).
\]  

(5)

Example 17. Let \(\circ - \circ\) be a graph of one edge and \(\phi\) be the \(i\)-th component of the argument. Then,

\[
\text{hom}(\circ - \circ, G, x; \phi) = \sum_{(u, v) \in E(G)} x_i(u) x_i(v).
\]  

(6)

The \((F, \phi)\)-convolution is invariant under the isomorphism as follows.

Theorem 18. For a simple graph \(F\), a function \(\phi: \mathbb{R}^p \to \mathbb{R}\), a vertex-featured graph \((G, x)\), and a permutation \(\sigma\) on \(V(G)\), we have

\[
\text{hom}(F, G, x; \phi) = \text{hom}(F, G^\sigma, x^\sigma, \phi).
\]  

(7)

Proof. Clear from the definition.

By Theorem 18, for any \(F\) and \(\Phi\), the \((F, \Phi)\)-convolution is invariant under the isomorphism as follows.

The strength of an embedding is characterized by the separability.

Lemma 19. Let \(\mathcal{F}\) be the set of all simple graphs and \(\Phi\) be the set of all continuous functions from \([0, 1]^p\) to \([0, 1]\). Then, \((G, x) \mapsto \text{hom}(\mathcal{F}, G, x; \Phi)\) is injective.

C.1. Universality Theorem

The overview of the datasets in this paper. Here, \(N\) denotes total number of graphs, \(\overline{n}\) denotes the average number of nodes, \(|c|\) denotes number of classes, \(\mathcal{X}\) denotes if the dataset consists of vertex features, and \(\mathcal{T}\) denotes if the dataset consists of vertex tags (or types).

| DATASETS   | \(N\) | \(\overline{n}\) | \(|c|\) | \(\mathcal{X}\) | \(\mathcal{T}\) |
|------------|-------|-----------------|-------|-------------|-------------|
| MUTAG      | 188   | 17.9            | 2     | no          | yes         |
| PTC-MR     | 344   | 25.5            | 2     | no          | yes         |
| NCI1       | 4110  | 29.8            | 2     | no          | yes         |
| PROTEINS   | 1113  | 39.1            | 2     | yes         | yes         |
| D&D        | 1178  | 284.3           | 2     | yes         | yes         |
| BZR        | 405   | 35.7            | 2     | yes         | yes         |
| RDT-BIN    | 2000  | 429.6           | 5     | no          | no          |
| RDT-5K     | 5000  | 508.5           | 5     | no          | no          |
| RDT-12K    | 11929 | 391.4           | 11    | no          | no          |
| COLLAB     | 5000  | 74.5            | 3     | no          | no          |
| IMDB-BIN   | 1000  | 19.8            | 2     | no          | no          |
| IMDB-MUL   | 1500  | 13.0            | 3     | no          | no          |
| Bipartite  | 200   | 70.0            | 2     | no          | no          |
| CSL        | 150   | 41.0            | 10    | no          | no          |
| Paulus 25  | 210   | 25.0            | 14    | no          | no          |

Table 4. Overview of the datasets in this paper. Here, \(N\) denotes total number of graphs, \(\overline{n}\) denotes the average number of nodes, \(|c|\) denotes number of classes, \(\mathcal{X}\) denotes if the dataset consists of vertex features, and \(\mathcal{T}\) denotes if the dataset consists of vertex tags (or types).
Graph Homomorphism Convolution

Figure 3. Example of MUTAG data

Figure 4. Elements of \( \mathcal{F}_{\text{tree}(6)} \)

| Methods    | Datasets         |
|------------|------------------|
|            | RDT-BIN | RDT-5K | RDT-12K | COLLAB | IMDB-BIN | IMDB-MUL |
| **Our experiments** (Average over 10 runs of stratified 10-folds CV) |         |         |         |         |          |          |
| GHC-Tree   | 88.42 ± 2.05    | 52.98 ± 1.83 | 44.8 ± 1.00 | 75.23 ± 1.71 | 72.10 ± 2.62 | 48.60 ± 4.40 |
| GHC-Cycles | 87.61 ± 2.45    | 52.45 ± 1.24 | 40.9 ± 2.01 | 72.59 ± 2.02 | 70.93 ± 4.54 | 47.61 ± 3.67 |
| GIN        | 74.10 ± 2.34    | 46.74 ± 3.07 | 32.56 ± 5.33 | 75.90 ± 0.81 | 70.70 ± 1.10 | 43.20 ± 2.00 |
| GNTK       | -               | -          | -          | 83.70 ± 1.00 | 75.61 ± 3.98 | 51.91 ± 3.56 |
| **Literature** (One run of stratified 10-folds CV) |         |         |         |         |          |          |
| GIN        | 92.4 ± 2.5      | 57.5 ± 1.5  | -         | 80.2 ± 1.9 | 75.1 ± 5.1 | 52.3 ± 2.8  |
| PATCHY-SAN | 86.3 ± 1.6      | 49.1 ± 0.7  | -         | 72.6 ± 2.2 | 71.0 ± 2.2 | 45.2 ± 2.8  |
| WL kernel  | 80.8 ± 0.4      | -          | -         | 79.1 ± 0.1 | 73.12 ± 0.4 | -          |
| Graphlet kernel | 60.1 ± 0.2 | -          | 31.8      | 64.7 ± 0.1 | -          | -          |
| AWL kernel | 87.9 ± 2.5      | 54.7 ± 2.9  | -         | 73.9 ± 1.9 | 74.5 ± 5.9 | 51.5 ± 3.6  |
| WL-OA kernel | 89.3      | -          | -         | 80.7 ± 0.1 | -          | -          |
| WL-W kernel | -          | -          | -         | -        | 74.37 ± 0.83 | -          |
| GNTK       | -               | -          | -         | 83.6 ± 1.0 | 76.9 ± 3.6 | 52.8 ± 4.6  |

Table 5. Graph classification accuracy (percentage) on popular non-vertex-featured benchmark datasets. This table provides the results obtained by averaging 10 times the 10-folds cross-validation procedure. Note that the results reported in the literature are run for only one 10-folds cross-validation. "-" denotes the result is not available or the experiment runs for more than 2 days (48 hours).
Graph Homomorphism Convolution

| METHODS       | DATASETS |
|---------------|----------|
|               | MUTAG    | PTC-MR | NCI1 | PROTEINS | D&D | BZR |
| Our experiments | (Average over 10 runs of stratified 10-folds CV) |
| GHC-Tree      | 89.28 ± 8.26 | 52.98 ± 1.83 | 48.8 ± 1.00 | 75.23 ± 1.71 | 72.10 ± 2.62 | 48.60 ± 4.40 |
| GHC-Cycle     | 87.81 ± 7.46 | 50.97 ± 2.13 | 47.4 ± 1.02 | 74.30 ± 1.93 | 70.10 ± 2.49 | 47.20 ± 3.84 |
| GHC-LabelTree | 88.86 ± 4.82 | 59.68 ± 7.98 | 73.95 ± 1.99 | 73.27 ± 4.17 | 76.50 ± 3.15 | 82.82 ± 4.37 |
| GIN           | 74.10 ± 2.34 | 46.74 ± 3.07 | 76.67 ± 1.16 | 75.9 ± 0.81  | 70.70 ± 1.11 | 43.20 ± 2.00 |
| GNTK          | 89.65 ± 7.5  | 68.2 ± 5.8  | 85.0 ± 1.2  | 76.65 ± 5.02 | 75.61 ± 3.98 | 83.64 ± 2.95 |

| Literature (One run of stratified 10-folds CV) |
| GIN                              | 89.4 ± 5.6  |
| PATCHY-SAN                       | 92.5 ± 4.2  |
| WL kernel                        | 90.4 ± 5.7  |
| Graphlet kernel                  | 85.2 ± 0.9  |
| AWL kernel                       | 87.9 ± 9.8  |
| WL-OA kernel                     | 84.5 ± 0.17 |
| WL-W kernel                      | 87.27 ± 1.5 |
| GNTK                             | 90.00 ± 8.5 |

Table 6. Graph classification accuracy (percentage) on popular vertex-featured (vertex-labeled) benchmark datasets. This table provides the results obtained by averaging 10 times the 10-folds cross-validation procedure. "-" denotes the result is not available in the literature or the experiment runs for more than 2 days (48 hours).

![Elements of $F_{\text{cycles}(8)}$](image)

Figure 5. Elements of $F_{\text{cycles}(8)}$

**Proof.** Let $(G, x)$ and $(G', y)$ be two non-isomorphic vertex-featured graphs. We distinguish these graphs by the homomorphism convolution.

If $G$ and $G'$ are non-isomorphic, by (Lovász, 1967), $\text{hom}(F, G, x; 1) \neq \text{hom}(F, G', y; 1)$ where 1 is the function that takes one for any argument.

Now we consider the case that $G = G'$. Let $\{1, \ldots, n\}$ be the set of vertices of $G$. Without loss of generality, we assume $x(1) \leq x(2) \leq \ldots$ where $\leq$ is the lexicographical order. Now we find a permutation $\pi$ such that $G = G^\pi$ and $y(\pi(1)), y(\pi(2)), \ldots$ are lexicographically smallest. Let $u \in \{1, \ldots, n\}$ be the smallest index such that $x(u) \neq y(u)$. By the definition, $x(u) \leq y(u)$. We choose $\psi$ by

$$
\psi(x) = \begin{cases} 
1, & x \leq x(u), \\
0, & \text{otherwise} 
\end{cases}
$$

(9)

Then, there exists $F \in \mathcal{F}$ such that $\text{hom}(F, G, x; \psi) \neq \text{hom}(F, G, y; \psi)$ because the graphs induced by $\{1, \ldots, k\}$ and $\{\pi(1), \ldots, \pi(k)\}$ are non-isomorphic because of the choice of $\pi$.

Now we approximate $\psi$ by a continuous function $\phi$. Because $(F, \phi)$-convolution is continuous in the vertex weights (i.e., $\phi(x(u)))$, by choosing $\phi$ sufficiently close to $\psi$, we get $\text{hom}(F, G, x; \phi) \neq \text{hom}(F, G, y; \phi)$.

We say that a sequence $(G_i, x_i)$ $(i = 1, 2, \ldots)$ of featured graphs is an $(\mathcal{F}, \Phi)$-convergent if for each $F \in \mathcal{F}$ and $\phi \in \Phi$ the sequence $\text{hom}(F, G_i, x_i; \phi)$ $(i = 1, 2, \ldots)$ is a convergent in $\mathbb{R}$. A function $f : (G, x) \mapsto f(G, x)$ is $(\mathcal{F}, \Phi)$-continuous if for any $(\mathcal{F}, \Phi)$-convergent $(G_i, x_i)$ $(i = 1, 2, \ldots)$, the limit $\lim_{i \to \infty} f(G_i, x_i)$ of the function exists and its only depends on the limits $\lim_{i \to \infty} \text{hom}(F, G_i, x_i, \phi)$ of the homomorphism convolutions for all $F \in \mathcal{F}$ and $\phi \in \Phi$.

Now we prove the universality theorem. Let $\mathcal{H}$ be a dense subset of the set of continuous functions, e.g., the set of polynomials or the set of functions represented by a deep neural network. Let $\mathcal{G}$ be a set of graphs. We define $\mathcal{H}(\mathcal{G}; \mathcal{F}, \Phi)$ by

$$
\mathcal{H}(\mathcal{G}; \mathcal{F}, \Phi) = \left\{ \sum_{F \in \mathcal{F}, \phi \in \Phi} h_{F, \phi}(\text{hom}(F_i; \phi)) : h_{F, \phi} \in \mathcal{H} \right\}
$$

(10)

where the argument of the function is restricted to $\mathcal{G}$. This is the set of functions obtained by combining universal approximators in $\mathcal{H}$ and the homomorphism convolutions $\text{hom}(F, G, x, \phi)$ for some $F \in \mathcal{F}$ and $\phi \in \Phi$. Let $\mathcal{G}$ be
a set of graphs, and let \( C(\mathcal{G}; \mathcal{F}, \Phi) \) be the set of \((\mathcal{F}, \Phi)\)-continuous functions defined on \( \mathcal{G} \). Then, we obtain the following theorem.

**Theorem 20** (Universal Approximation Theorem). Let \( \mathcal{G} \) be a compact set of graphs whose number of vertices are bounded by a constant. Then, \( \mathcal{H}(\mathcal{G}; \mathcal{F}, \Phi) \) is dense in \( C(\mathcal{G}; \mathcal{F}, \Phi) \).

**Proof.** Because the number of vertices are bounded, the space of converging sequences is identified as \( \mathcal{G} \). Therefore, this space is compact Hausdorff. The separability is proved in Theorem 19. Hence, we can use the Stone–Weierstrass theorem to conclude this result. \( \square \)

**Remark 21.** We conjecture that the above result can be extended to the infinite graphs (say, graphons).

### C.2. Choice of \( \mathcal{F} \) and \( \Phi \)

In an application, we have to choose \( \mathcal{F} \) and \( \Phi \) appropriately. The criteria of choosing them will be the following.

- **Representability.** The \((\mathcal{F}, \Phi)\)-convolutions can separate the graphs in which we are interested in.
- **Efficiency.** The \((\mathcal{F}, \Phi)\)-convolutions can be efficiently computable. This trivially limits both \( \mathcal{F} \) and \( \Phi \) as finite sets.

The choice of \( \Phi \) will depend on the property of the vertex features. We will include the constant function 1 if the topology of the graph is important. We will also include the \( i \)-th component of the arguments. If we know some interaction between the features is important, we can also include the cross-terms. We can also use the dimensionality reduction. This part does not affect the efficiency.

The choice of \( \mathcal{F} \) relates with the topology of the graphs of interest. If \( \Phi = \{1\} \) where 1 is the constant function, the homomorphism convolution coincides with the homomorphism number; therefore its separability is well-studied. Some results are shown in Table 1.

Here, we focus on the efficiency. In general, computing \( \text{hom}(F, G, x, \phi) \) is \#P-hard. However, it is computable in polynomial time if \( F \) has a bounded tree The treewidth of a graph \( F \), denoted by \( \text{tw}(F) \), is a graph parameter that measures the tree-likeness of the graph. The treewidth is one if and only if it is a tree, and the treewidth is two if and only if its biconnected components are series-parallel graphs. The following result holds.

**Theorem 22.** \( \text{hom}(F, G, x, \phi) \) is computable in \( |V(G)|^{\text{tw}(F)+1} \) time, where \( \text{tw}(F) \) is the tree-width of \( F \).

For vertex-featured graph \((G, x)\), we define the \( i \)-th component graph \((G, x)^{(i)}\) by the vertex-weighted graph obtained by removing the vertices having zero \( i \)-th component.

**Theorem 23.** If \( \text{hom}(F, (G_1, x_1)) = \text{hom}(F, (G_2, x_2)) \) then the graphs of the \( i \)-th components \((G_1, x_1)^{(i)}\) and \((G_2, x_2)^{(i)}\) are isomorphic.

**Proof.** See Lovasz theorem paper. \( \square \)

We say that a sequence \([(G_i, x_i) : i = 1, 2, \ldots] \) of vertex-featured graphs are positively \( F \)-convergent if (1) all the weights \( x_i \) are strictly positive, (2) there is no dominant vertex weight, i.e., \( \max x(u)/\sum x(u) \to 0 \), and (3) for all \( F \in \mathcal{F} \), the sequence \( \{\text{hom}(F, (G_i, x_i)) / |V(G_i)|^{\text{tw}(F)} : i = 1, 2, \ldots\} \) is a convergent in \( \mathbb{R} \). Two \( F \)-convergents \([(G_i, x_i) : i = 1, 2, \ldots] \) and \([(G'_i, x'_i) : i = 1, 2, \ldots] \) converges to the same point if the corresponding sequences converges to the same value for all \( F \in \mathcal{F} \) converges to the same point. A function \( f \) is continuous if \( \lim_{i \to \infty} f((G_i, x_i)) \) have the same value for all \( F \)-convergent sequence converges to the same point.

Now we state our main theorem.

**Theorem 24** (Universal Approximation Theorem of \( \mathcal{F} \)-Invariant Functions). Any continuous \( \mathcal{F} \)-invariant function is arbitrarily accurately approximated by the form

\[
\sum_{F \in \mathcal{F}} a_F \cos(b_F \text{hom}(F, (G, x)))
\]

where \( a_F, b_F \in \mathbb{R} \).

We can replace the cosine function in the above theorem by arbitrary class of functions that can arbitrary accurately approximate the cosine function. For example, we can use the sigmoid function, the ReLU function, the polynomials, and the radial-basis functions can be used instead of the cosine function.

**Proof.** Proof outline:

Step 1: Prove the lemma if \( \mathcal{F} \) is the set of all graphs. This is basically done in our preprint (feature part is new). How to cite? or merge? *** I aware that this is not so easy. Convergence of the feature values must be considered. ***

Step 2: Prove the lemma for general \( \mathcal{F} \) as follows. Let \( \mathcal{G} \) be the set of graphons with vertex features. We prove that this space is compact Hausdorff (in Step 1).

We first notice that the \( \mathcal{F} \)-equivalence is an equivalence relation. Let us consider the quotient space \( \mathcal{G}/\mathcal{F} \) by this equivalence relation. Then, \( \mathcal{G}/\mathcal{F} \) is a compact Hausdorff space because it is a quotient space of a compact Hausdorff space. Also, by the definition of the quotient topology, we
can identify a continuous $\mathcal{F}$-invariant function in $\mathcal{G}$ and a continuous function in $\mathcal{G}_{\mathcal{F}}$.

Then, the remaining part of the proof is similar to Step 1. We use the Stone–Weierstrass theorem. Then we obtain the lemma. \qed