Expectation-Complete Graph Representations with Homomorphisms

Pascal Welke [*1 2] **Maximilian Thiessen** [*2] **Fabian Jogl** [2 3] **Thomas Gärtner** [2]

Abstract

We investigate novel random graph embeddings that can be computed in expected polynomial time and that are able to distinguish all non-isomorphic graphs *in expectation*. Previous graph embeddings have limited expressiveness and either cannot distinguish all graphs or cannot be computed efficiently for every graph. To be able to approximate arbitrary functions on graphs, we are interested in efficient alternatives that become arbitrarily expressive with increasing resources. Our approach is based on Lovász' characterisation of graph isomorphism through an infinite dimensional vector of homomorphism counts. Our empirical evaluation shows competitive results on several benchmark graph learning tasks.

1. Introduction

We study novel efficient and expressive graph embeddings motivated by Lovász' characterisation of graph isomorphism through homomorphism counts. While most graph embeddings drop *completeness*—the ability to distinguish all pairs of non-isomorphic graphs—in favour of runtime, we devise efficient embeddings that retain completeness *in expectation*. The specific way in which we sample a fixed number of pattern graphs guarantees an expectation-complete embedding in expected polynomial time. In this way, repeated sampling will eventually allow us to distinguish all pairs of non-isomorphic graphs, a property that no efficiently computable deterministic embedding can guarantee. In comparison, most recent graph neural networks are inherently limited by the expressiveness of some k-dimensional Weisfeiler-Leman isomorphism test (Morris et al., 2019; Xu et al., 2019).

[*]Equal contribution [1]Machine Learning and Artificial Intelligence Lab, University of Bonn, Germany [2]Research Unit Machine Learning, TU Wien, Austria [3]Center for Artificial Intelligence and Machine Learning, TU Wien, Austria. Correspondence to: Pascal Welke <pascal.welke@tuwien.ac.at>, Maximilian Thiessen <maximilian.thiessen@tuwien.ac.at>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Our approach to achieve an expectation-complete graph embedding is based on homomorphism counts. These are known to determine various properties of graphs important for learning, such as the degree sequence or the eigenspectrum (Hoang & Maehara, 2020). Furthermore, homomorphism counts are related to the Weisfeiler-Leman hierarchy (Dvořák, 2010; Dell et al., 2018), which is the standard measure for expressiveness on graphs (Morris et al., 2019). They also determine subgraph counts (Curticapean et al., 2017) and the distance induced by the homomophism counts is asymptotically equivalent to the *cut distance*, which Grohe (2020) and Klopp & Verzelen (2019) motivated as an appropriate graph similarity for graph learning tasks.

In Section 2 we introduce the required concepts. In Section 3 we discuss that general expectation-complete embeddings can eventually distinguish all pairs of non-isomorphic graphs (Lemma 3), which leads to a universal representation (Theorem 4). Then we propose our expectation-complete embedding based on sampling entries from the Lovász vector (Theorem 7) and bound the number of samples required to provably get as close as desired to the full Lovász vector (Theorem 8). In Section 4, we show how to compute our embedding efficiently in expected polynomial time (Theorem 14). In Section 5, we show how to combine our embedding with graph neural networks. Finally, we discuss related work in Section 6 and show competitive results on benchmark datasets in Section 7 before Section 8 concludes.

2. Background and Notation

We start by defining the required concepts and notation. A *graph* $G = (V, E)$ consists of a set $V = V(G)$ of *vertices* and a set $E = E(G) \subseteq \{e \subseteq V \mid |e| = 2\}$ of *edges*. In this work we only consider undirected graphs. The *size* $v(G)$ of a graph G is the number of its vertices and by $\mathcal{G}_n$ we denote the set of all graphs with size at most $n \in \mathbb{N}$. In the following F and G denote graphs, where F represents a *pattern* graph and G a graph in our training set. A *homomorphism* $\Phi : V(F) \to V(G)$ is a map that preserves edges, i.e. $\{v, w\} \in E(F) \Rightarrow \{\Phi(v), \Phi(w)\} \in E(G)$. Note that homomorphisms, unlike *subgraph isomorphisms*, allow non-injectivity: multiple vertices of F can be mapped to the same vertex of G, see Figure 1. Let $\hom(F, G)$ denote the number of homomorphisms from F to G and let $\varphi_{\mathcal{G}}(G) = (\hom(F, G))_{F \in \mathcal{G}}$ denote the vector of homomorphism

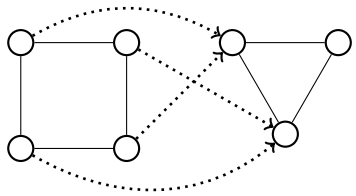

Figure 1. Example homomorphism: mapping a 4-cycle to an edge.

counts from each graph of a family of graphs $\mathcal{G}$ to G. We define the shorthand $\varphi_n(G) = \varphi_{\mathcal{G}_n}(\mathrm{G})$. We also define the *homomorphism density* $t(F, G) = \hom(F, G)/v(G)^{v(F)}$, corresponding to the probability that a mapping from $V(F)$ to $V(G)$ drawn uniformly at random is a homomorphism. Similarly to φ, we define $\psi_{\mathcal{G}}(G) = t(\mathcal{G}, G) = (t(F, G))_{F \in \mathcal{G}}$ and $\psi_n(G) = \psi_{\mathcal{G}_n}$. An *isomorphism* between two graphs G and G' is a bijection $I : V(G) \to V(G')$ such that $\{v, w\} \in E(G)$ if and only if $\{I(v), I(w)\} \in E(G')$. If there is an isomorphism between G and G', we say they are *isomorphic* and denote it as $G \simeq G'$. We say that a probability distribution $\mathcal{D}$ over a countable domain $\mathcal{X}$ has *full support* if each $x \in \mathcal{X}$ has nonzero probability $\Pr_{X \sim \mathcal{D}}(X = x) > 0$.

2.1. Complete Graph Embeddings

Classical graph kernel and recent (neural) graph representation methods perform learning on graphs by (potentially implicitly) embedding them into a real vector space $\mathcal{H}$. A *graph embedding* is a map $\varphi : \mathcal{G} \to \mathcal{H}$ defined on a set of graphs $\mathcal{G}$. A graph embedding φ is called *permutation-invariant* if for all $G \simeq G' \in \mathcal{G}$ it holds that $\varphi(G) = \varphi(G')$. All common graph kernels (Kriege et al., 2020) and standard message-passing neural networks (Xu et al., 2019) are permutation-invariant. Now we define *completeness*, which requires the opposite direction of the implication.

Definition 1. *A permutation-invariant graph embedding $\varphi : \mathcal{G} \to V$ is complete (on $\mathcal{G}$) if $\varphi(G) \neq \varphi(G')$ for all non-isomorphic $G, G' \in \mathcal{G}$.*

Completeness is necessary if we want to be *universal*, that is, be able to approximate any permutation-invariant function $f : \mathcal{G} \to \mathbb{R}$. In particular we would not be able to approximate a function f with $f(G) \neq f(G')$ for two non-isomorphic graphs G and G' with $\varphi(G) = \varphi(G')$.

Complete graph embeddings allow to determine whether two graphs are isomorphic, as $G \simeq G'$ if and only if $\varphi(G) = \varphi(G')$. Deciding graph isomorphism is a classical problem in graph theory whose computational complexity is a major open problem (Babai, 2016). While the problem is in NP, neither a polynomial-time algorithm is known nor it is known whether the problem is NP-complete. Thus, we always face a trade-off between efficiency and expressiveness: complete graph embeddings are unlikely to be computable in polynomial time (Gärtner et al., 2003) and hence most graph representations drop completeness in favour of polynomial runtime.

If $\mathcal{H}$ is a (real) Hilbert space with inner product $\langle \cdot, \cdot \rangle \to \mathbb{R}$, and not just a vector space, we can define a *graph kernel* $k_\varphi(G, G') = \langle \varphi(G), \varphi(G') \rangle$ using any permutation-invariant graph embedding $\varphi : \mathcal{G} \to \mathcal{H}$. We call k_φ *complete* if φ is complete. Note that

$$k_\varphi(G, G) - 2k_\varphi(G, G') + k_\varphi(G', G') = \|\varphi(G) - \varphi(G')\|^2 \tag{1}$$

which for a complete kernel is 0 if and only if $G \simeq G'$. Thus, evaluating a complete graph kernel is at least as hard as deciding graph isomorphism, even if φ is not known or computed explicitly (Gärtner et al., 2003).

In this work, we avoid the previously mentioned trade-off by using random graph embeddings than can be computed in expected polynomial time. While dropping completeness, this allows us to keep a slightly weaker yet still desirable property: *completeness in expectation*.

3. Expectation-Complete Graph Embeddings

In the remainder of this work we will consider *random graph embeddings*. These are graph embeddings $\varphi_X : \mathcal{G} \to \mathcal{H}$ that are parameterised by a random variable X. Algorithmically, we can think of $\varphi_X(G)$ as first sampling a random variable $X \sim \mathcal{D}$ from a distribution $\mathcal{D}$ and then computing $\varphi_X(G)$. If the expectation $\mathbb{E}_{X \sim \mathcal{D}}[\varphi_X(G)]$ is defined for all $G \in \mathcal{G}$, we can define a (deterministic) graph embedding $\mathbb{E}_{X \sim \mathcal{D}}[\varphi_X(\cdot)] : \mathcal{G} \to \mathcal{H}$. This leads us to the central notion of this paper.

Definition 2. *A random graph embedding φ_X is expectation-complete if the graph embedding $\mathbb{E}_X[\varphi_X(\cdot)]$ is complete. The corresponding kernel $k_X(G, G') = \langle \varphi_X(G), \varphi_X(G') \rangle$ is expectation-complete if φ_X is expectation-complete.*

Expectation-complete graph embeddings satisfy a useful property, which no non-complete deterministic graph embedding can satisfy: they eventually will be complete if we sample often enough.

Lemma 3. *Let $\varphi_X : \mathcal{G} \to \mathcal{H}$ be a expectation-complete graph embedding and $G, G' \in \mathcal{G}$ which are not isomorphic. For any $\delta > 0$, there exists $L \in \mathbb{N}$ such that for all $\ell \geq L$*

$$(\varphi_{X_1}(G), \ldots, \varphi_{X_\ell}(G)) \neq (\varphi_{X_1}(G'), \ldots, \varphi_{X_\ell}(G'))$$

with probability $1 - \delta$, where $X_1, \ldots, X_\ell \sim \mathcal{D}$ i.i.d.

Proof. Let G, G' be non-isomorphic graphs. Since φ_X is expectation-complete, it must hold that $\mathbb{E}[\varphi_X(G)] \neq$

$\mathbb{E}[\varphi_X(G')]$, which in particular means that there exists a set $A_{G,G'}$ of outcomes of X with $\Pr(X \in A_{G,G'}) = p_{G,G'} > 0$ such that for all $a \in A_{G,G'}$ it holds that $\varphi_a(G) \neq \varphi_a(G')$. We need $\Pr(\exists i \in \{1, \dots, \ell\} : X_i \in A_{G,G'}) \geq 1 - \delta$, hence $1 - (1 - p_{G,G'})^\ell \geq 1 - \delta$ must hold. Solving for ℓ we see that $\ell \geq L = \left\lceil \frac{\log(1/\delta)}{\log(\frac{1}{1-p_{G,G'}})} \right\rceil$ is sufficient to guarantee that there will be at least one X_i in A with probability at least $1 - \delta$, implying $\varphi_{X_i}(G) \neq \varphi_{X_i}(G')$. □

This leads to the following result, that sampling eventually yields universality.

Theorem 4. *Let $n \in \mathbb{N}$, $\varphi_X : \mathcal{G}_n \to \mathbb{R}^d$ be a finite-dimensional expectation-complete graph embedding and $f : \mathcal{G}_n \to \mathbb{R}$ a permutation-invariant function. For any $\varepsilon > 0$ and $\delta > 0$ there exists an $\ell \in \mathbb{N}$ and a multi-layer-perceptron $g : \mathbb{R}^{d\ell} \to \mathbb{R}$ such that*

$$|f(G) - g(\varphi_{X_1}(G), \dots, \varphi_{X_\ell}(G))| < \varepsilon$$

for all $G \in \mathcal{G}_n$ with probability at least $1 - \delta$, where $X_1, \dots, X_\ell \sim \mathcal{D}$ i.i.d.

Proof. Let $N = |\mathcal{G}_n|$, $\mathcal{G}_n = \{G_1, \dots, G_N\}$ and $f(G_i) = y_i$ for all i. As in the proof of *Lemma* 3 we know that for each pair $G, G' \in \mathcal{G}$ of non-isomorphic graphs there exists an event $A_{G,G'}$ with non-zero probability $p_{G,G'}$ guaranteeing that $\varphi_X(G) \neq \varphi_X(G')$. Let $p = \min_{G,G'} p_{G,G'} > 0$. We have to satisfy this for all pairs of non-isomorphic graphs simultaneously. By applying a union bound on the complement (meaning at least one $A_{G,G'}$ does not happen) and bounding each of these terms through Lemma 3, we see that

$$\ell \geq \frac{\log|\mathcal{G}_n| + \log(1/\delta)}{\log(\frac{1}{1-p})}$$

samples are sufficient to guarantee that the embedding $\varphi_\ell(G) = (\varphi_{X_1}(G), \dots, \varphi_{X_\ell}(G))$ is complete with probability $1 - \delta$. Note that $\log|\mathcal{G}_n| \leq n^2$, meaning that if we treat p and δ as constants then $\mathcal{O}(n^2)$ many samples suffice.

It remains to show that there is an MLP g which can approximate the points $(\varphi_\ell(G_1), y_1), \dots, (\varphi_\ell(G_N), y_N)$. It is clear that there exists a multivariate polynomial exactly fitting all the points. Then we can apply universal function approximation to the bounded region spanned by the N points and approximate the polynomial. □

3.1. Expectation-Completeness Through Graph Homomorphisms

We now present one way to achieve expectation-completeness. We use the classical result of Lovász (1967) that all homomorphism counts up to $n = \max\{v(G), v(G')\}$ determine if G and G' are isomorphic.

Theorem 5 (Lovász (1967)[1])**.** *Two graphs $G, G' \in \mathcal{G}_n$ are isomorphic if and only if $\varphi_n(G) = \varphi_n(G')$.*

This provides a powerful graph embedding for learning tasks on graphs (Dell et al., 2018; Hoang & Maehara, 2020; Barceló et al., 2021). We can define a simple kernel on $\mathcal{G}_n$ with the canonical inner product using φ_n.

Definition 6 (Complete Lovász kernel)**.** *Let $k_{\varphi_n}(G, G') = \langle \varphi_n(G), \varphi_n(G') \rangle$.*

Note that φ_n and k_{φ_n} are both complete on $\mathcal{G}_n$, and hence can be used to distinguish non-isomorphic graphs of size up to n. We can use the Lovász vector embedding φ_n to devise graph embeddings that are expectation-complete. For that let $e_F \in \mathbb{R}^{\mathcal{G}_n}$ be the 'Fth' standard basis unit-vector of $\mathbb{R}^{\mathcal{G}_n}$. For a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ define the graph embedding $\varphi_F(G) = \hom(F, G) e_F$ with $F \sim \mathcal{D}$.

Theorem 7. *For a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ and $F \sim \mathcal{D}$, the random embedding $\varphi_F(\cdot)$ and the corresponding kernel are expectation-complete on $\mathcal{G}_n$.*

Proof. Let φ_F with $F \sim \mathcal{D}$ be as stated and $G \in \mathcal{G}_n$. Then

$$g = \mathbb{E}_F[\varphi_F(G)] = \sum_{F' \in \mathcal{G}_n} \Pr\left(F = F'\right) \hom(F', G) e_{F'} \,.$$

The vector g has the entries $(g)_{F'} = \Pr\left(F = F'\right) \hom(F', G)$. Let G' be a graph that is non-isomorphic to G and let $g' = \mathbb{E}_F[\varphi_F(G')]$ accordingly. By Theorem 5 we know that $\varphi_n(G) \neq \varphi_n(G')$. Thus, there is an F' such that $\hom(F', G) \neq \hom(F', G')$. By definition of $\mathcal{D}$ we have that $\Pr(F = F') > 0$ and hence $\Pr(F = F') \hom(F', G) \neq \Pr(F = F') \hom(F', G')$ which implies $g \neq g'$. That shows that $\mathbb{E}_F[\varphi_F(\cdot)]$ is complete and concludes the proof. □

We now analyse how close we are to the actual Lovász kernel, if we sample ℓ patterns $\mathcal{F} = (F_1, \dots, F_\ell)$ i.i.d. from $\mathcal{D}$. We consider $\varphi_{\mathcal{F}} = \sum_{F \in \mathcal{F}} \varphi_F$ and the kernel $k_{\mathcal{F}}(G, G') = \langle \varphi_{\mathcal{F}}(G), \varphi_{\mathcal{F}}(G') \rangle$. While formally working in $\mathbb{R}^{\mathcal{G}_n}$, we can restrict the analysis (and practical computation) to $\mathbb{R}^{\mathcal{F}}$, ignoring dimensions that only contain zeros.

We apply standard techniques similar to Rahimi & Recht (2007), Kontorovich & Nadler (2009), Shervashidze et al. (2009), and Wu et al. (2019). For convenience we will perform the analysis using the homomorphism densities ψ_F. Let $D \in \mathbb{R}^{\mathcal{G}_n \times \mathcal{G}_n}$ be a diagonal matrix with $D_{FF} = \Pr_{X \sim \mathcal{D}}(X = F)$ and let $J_F \in \{0, 1\}^{\mathcal{G}_n \times \mathcal{G}_n}$ be a matrix that is 1 at the FFth position and 0 everywhere else. For the expectation of the random kernel $\langle \psi_F(G), \psi_F(G') \rangle$ it

[1] For a more recent proof see Theorem 5.29 and the comments below in Lovász (2012).

holds that

$$\begin{aligned}\mathbb{E}_{F\sim\mathcal{D}}[\langle\psi_F(G),\psi_F(G')\rangle] &= \mathbb{E}_{F\sim\mathcal{D}}[\psi_{\mathcal{G}_n}^{\mathsf{T}}(G)J_F\psi_{\mathcal{G}_n}(G')]\\ &= \langle\sqrt{D}\psi_{\mathcal{G}_n}(G),\sqrt{D}\psi_{\mathcal{G}_n}(G')\rangle\\ &=: k_{\mathcal{D}}(G,G')\,.\end{aligned}$$

Note that $k_{\mathcal{D}}(G,G')$ is still a complete kernel as the complete graph embedding $\psi_{\mathcal{G}_n}$ is just scaled by $\sqrt{D}$, which is invertible as $\mathcal{D}$ has full support. For a sample $\mathcal{F}$ of ℓ patterns we get the joint (averaged) embedding $\psi_{\mathcal{F}}(G) = {}^1\!/\!\sqrt{\ell}(t(F_1,G),\ldots,t(F_\ell,G))$ and get the corresponding (averaged) kernel

$$\tilde{k}_{\mathcal{F}}(G,G') = \langle\psi_{\mathcal{F}}(G),\psi_{\mathcal{F}}(G')\rangle = \frac{1}{\ell}\sum_{i=1}^{\ell}\psi_{F_i}(G)\psi_{F_i}(G')\,.$$

Applying a Hoeffding bound we get

$$\Pr\left(\left|\tilde{k}_{\mathcal{F}}(G,G') - k_{\mathcal{D}}(G,G')\right| > \varepsilon\right) \leq 2e^{-2\varepsilon^2\ell}\,.$$

Note that the previous bound holds for a fixed pair G and G'. We can apply it to each pair in the training sample to get the following result.

Theorem 8. *Let $\varepsilon,\delta\in(0,1)$, $\mathcal{D}$ be a distribution on $\mathcal{G}_n$ with full support, and let $S\subseteq\mathcal{G}_n$ be a finite set of graphs. If we sample $\mathcal{F}=(F_1,\ldots,F_\ell)\sim\mathcal{D}^\ell$ i.i.d. with*

$$\ell = \mathcal{O}\left(\frac{\log({}^{|S|}\!/\delta)}{\varepsilon^2}\right)$$

we can guarantee that

$$\max_{G,G'\in S}\left|\tilde{k}_{\mathcal{F}}(G,G') - k_{\mathcal{D}}(G,G')\right| < \varepsilon$$

with probability at least $1-\delta$.

Proof. We have to show that

$$\Pr\left(\max_{G,G'\in S}\left|\tilde{k}_{\mathcal{F}}(G,G') - k_{\mathcal{D}}(G,G')\right| > \varepsilon\right) < \delta\,.$$

By a union bound it is sufficient if

$$\sum_{G,G'\in S}\Pr\left(\left|\tilde{k}_{\mathcal{F}}(G,G') - k_{\mathcal{D}}(G,G')\right| > \varepsilon\right) < \delta$$

and by applying Hoeffding bound to each term in the sum get $|S|^2 2e^{-2\varepsilon^2\ell} < \delta$. Solving for ℓ yields that $\ell = \mathcal{O}\left(\frac{\log({}^{|S|}\!/\delta)}{\varepsilon^2}\right)$ is sufficient. □

Corollary 9. *Let $\varepsilon,\delta\in(0,1)$, $\mathcal{D}$ be a distribution on $\mathcal{G}_n$ with full support. If we sample $\mathcal{F}=(F_1,\ldots,F_\ell)\sim\mathcal{D}^\ell$ i.i.d. with*

$$\ell = \mathcal{O}\left(\frac{n^2+\log({}^1\!/\delta)}{\varepsilon^2}\right)$$

we can guarantee that

$$\max_{G,G'\in\mathcal{G}_n}\left|\tilde{k}_{\mathcal{F}}(G,G') - k_{\mathcal{D}}(G,G')\right| < \varepsilon$$

with probability at least $1-\delta$.

Proof. Apply Theorem 8 with $S=\mathcal{G}_n$. We upper bound the number of graphs with up to n vertices as $|\mathcal{G}_n|\leq 2^{(n^2)}$. □

Hence, we achieve a bound for all graphs in $\mathcal{G}_n$ while sampling only $\mathcal{O}(n^2)$ patterns.

While we stated the previously achieved bounds for kernels, we can easily transform them to bounds on the induced distances of the graph embeddings using Equation (1).

Corollary 10. *Let $\varepsilon,\delta\in(0,1)$, $\mathcal{D}$ be a distribution on $\mathcal{G}_n$ with full support. If we sample $\mathcal{F}=(F_1,\ldots,F_\ell)\sim\mathcal{D}^\ell$ i.i.d. with*

$$\ell = \mathcal{O}\left(\frac{n^2+\log({}^1\!/\delta)}{\varepsilon^2}\right)$$

we can guarantee that for all $G,G'\in\mathcal{G}_n$ simultaneously

$$\left|\|\psi_{\mathcal{F}}(G)-\psi_{\mathcal{F}}(G')\|^2 - \|\sqrt{D}(\psi_n(G)-\psi_n(G'))\|^2\right| < \varepsilon$$

with probability at least $1-\delta$.

Thus, our results apply not only to kernel methods, but also to learning methods that use the graph embedding directly, such as multilayer perceptrons.

3.2. Graphs with Unbounded Size

In this section, we generalise the previous results to the set of all finite graphs $\mathcal{G}_\infty$. Theorem 5 holds for $G,G'\in\mathcal{G}_\infty$ and the mapping φ_∞ that maps each $G\in\mathcal{G}_\infty$ to an infinite-dimensional vector. The resulting vector space, however, is not a Hilbert space with the usual inner product. To see this, consider any graph G that has at least one edge. Then $\hom(P_n,G)\geq 2$ for every path P_n of length $n\in\mathbb{N}$. Thus, the inner product $\langle\varphi_\infty(G),\varphi_\infty(G)\rangle$ is not finite.

To define a kernel on $\mathcal{G}_\infty$ without fixing a maximum size of graphs, i.e., restricting to $\mathcal{G}_n$ for some $n\in\mathbb{N}$, we define the countable-dimensional vector $\varphi_\infty^{\downarrow}(G) = \Big(\hom_{v(G)}(F,G)\Big)_{F\in\mathcal{G}_\infty}$ where

$$\hom_{v(G)}(F,G) = \begin{cases}\hom(F,G) & \text{if } v(F)\leq v(G)\,,\\ 0 & \text{if } v(F)>v(G)\,.\end{cases}$$

That is, $\varphi_\infty^{\downarrow}(G)$ is the projection of $\varphi_\infty(G)$ to the subspace that gives us the homomorphism counts for all graphs of *size at most of* G. Note that this is a well-defined map of graphs to a subspace of the ℓ^2 space, i.e., sequences $(x_i)_i$ over $\mathbb{R}$

with $\sum_i |x_i|^2 < \infty$. Hence, the kernel given by the canonical inner product $k^{\downarrow}_{\infty}(G, G') = \langle \varphi^{\downarrow}_{\infty}(G), \varphi^{\downarrow}_{\infty}(G') \rangle_{\ell^2}$ is finite and positive semi-definite. Note that we can rewrite $k^{\downarrow}_{\infty}(G, G') = k_{\min}(G, G') = \langle \varphi_{n'}(G), \varphi_{n'}(G') \rangle$ where $n' = \min\{v(G), v(G')\}$. While the first hunch might be to count patterns up to $\max\{v(G), v(G')\}$, this is not necessary to guarantee completeness.

Lemma 11. *$k_{\min}$ is a complete kernel on $\mathcal{G}_\infty$.*

The proof can be found in Appendix A.

Given a sample of graphs S, we note that for $n = \max_{G \in S} v(G)$ we only need to consider patterns up to size n.[2] As the number of graphs of a given size n is superexponential, it is impractical to compute all such counts. Hence, we propose to resort to sampling.

Theorem 12. *Let $\mathcal{D}$ be a distribution on $\mathcal{G}_\infty$ with full support and $G \in \mathcal{G}_\infty$. Then $\varphi^{\downarrow}_F(G) = \hom_{v(G)}(F, G) e_F$ with $F \sim \mathcal{D}$ and the corresponding kernel are expectation-complete.*

The proof can be found in Appendix A.

Note that $k_{\min}$ has the following interesting practical property. If we train a kernel-based classifier on a sample $S \subseteq \mathcal{G}_n$ and want to classify a graph with size larger than n we do not have to recompute the embeddings $\varphi^{\downarrow}_{\infty}(G)$ for $G \in S$ as the terms corresponding to patterns with size $> n$ in the kernel are zero anyway.

4. Computing Embeddings in Expected Polynomial Time

An expectation-complete graph embedding should be efficiently computable to be practical, otherwise we could simply use deterministic complete embeddings. In this section, we describe our main result achieving polynomial runtime in expectation. The best known algorithm (Díaz et al., 2002) to exactly compute $\hom(F, G)$ takes time

$$\mathcal{O}(v(F) v(G)^{\mathrm{tw}(F)+1}) \tag{2}$$

where $\mathrm{tw}(F)$ is the *treewidth* of the pattern graph F. Thus, a straightforward sampling strategy to achieve polynomial runtime in expectation is to give decreasing probability mass to patterns with higher treewidth. Unfortunately, in the case of $\mathcal{G}_\infty$, this is not possible.

Proposition 13. *There exists no distribution $\mathcal{D}$ with full support on $\mathcal{G}_\infty$ such that the expected runtime of Eq. (2) becomes polynomial in $v(G)$ for all $G \in \mathcal{G}_\infty$.*

The proof can be found in Appendix A.

To resolve this issue we have to take the size of the largest graph in our sample into account. For a given sample $S \subseteq \mathcal{G}_n$ of graphs, where n is the maximum number of vertices in S, we can construct simple distributions achieving polynomial time in expectation.

Theorem 14. *There exists a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ such that computing the expectation-complete graph embedding $\varphi^{\downarrow}_F(G)$ with $F \sim \mathcal{D}$ takes polynomial time in $v(G)$ in expectation for all $G \in \mathcal{G}_n$.*

Proof sketch. We first draw a treewidth upper bound k from an appropriate distribution. For example, to satisfy a runtime of $\mathcal{O}(v(G)^{d+1})$ in expectation for some constant $d \in \mathbb{N}$, a Poisson distribution with $\lambda \leq \frac{1 + d \log n}{n}$ is sufficient for any $G \in \mathcal{G}_n$. We have to ensure that each possible graph with treewidth up to k gets a nonzero probability of being drawn. For that we first draw a k-tree—a maximal graph of treewidth k—and then take a random subgraph of it. See Appendix A for the full proof. □

We do not require that the patterns are sampled uniformly at random. It merely suffices that each pattern has a nonzero probability of being drawn. We get a similar result for our random Lovász embedding.

Theorem 15. *There exists a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ such that computing the expectation-complete graph embedding $\varphi_F(G)$ with $F \sim \mathcal{D}$ takes polynomial time in $v(G)$ in expectation for all $G \in \mathcal{G}_n$.*

The proof can be found in Appendix A.

Combining these results with Theorem 8, we see that for any fixed δ and ε we need in total an expected polynomial runtime to construct the embedding $\varphi_{\mathcal{F}}$ with $\mathcal{F} = (F_1, \ldots, F_\ell)$ with $F_i \sim \mathcal{D}$ for all $i \in \{1, \ldots, \ell\}$ and ℓ as in Theorem 8.

5. Practical Application

So far, we have restricted our discussion to graphs without node attributes. However, many real world datasets have attributes on their vertices and edges.We now discuss how to apply our embedding and kernel in such contexts.

It is conceptually possible to devise sampling schemes and corresponding distributions $\mathcal{D}$ over graphs with discrete vertex and edge labels. However, in practice this tends to result in unusable probabilities. For any pattern F, a single edge with labeled endpoints which are not connected in G results in $\hom(F, G) = 0$. Hence, the resulting graph embeddings $\varphi_{\mathcal{F}}$ become very sparse and practically uninformative.

We instead propose to consider labeled graphs as unlabeled for the purpose of homomorphism counting and suggest to include attribute information by applying a message passing graph neural network (GNN). Combining any GNN graph level representation with our embedding for a fixed set of sampled patterns $\mathcal{F}$ as shown in Figure 2 is straightforward

[2] It is sufficient to go up to the size of the second largest graph.

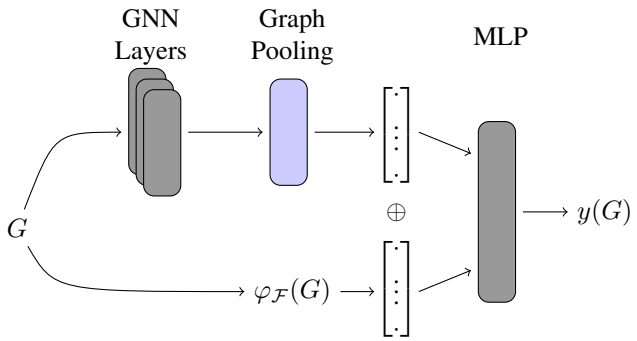

Figure 2. Architecture of combining expectation-complete embeddings with MPNN representations for graph learning.

and allows to make any GNN architecture more expressive. In particular the direct sum of $\varphi_{\mathcal{F}}$ and the GNN representation is expectation-complete on attribute-free graphs; a property that the GNN representation alone does not posess. Theorem 4 then implies that we can approximate any function on $\mathcal{G}_n$ using a suitable MLP with high probability.

6. Discussion and Related Work

k-WL test The k-dimensional Weisfeiler-Leman (WL) test[3] (Cai et al., 1992) and the Lovász vector restricted to the set $\mathcal{T}_k$ of graph patterns with treewidth up to k are equally expressive (Dvořák, 2010; Dell et al., 2018), that is, they distinguish the same non-isomorphic graphs. Puny et al. (2023) discuss this relationship in the context of invariant polynomials. We now propose a random graph embedding with expected polynomial runtime that matches the expressiveness of k-WL in expectation. The same holds for MPNNs and k-GNNs, as their expressiveness is bounded by k-WL (Xu et al., 2019; Morris et al., 2019). Let $\mathcal{D}$ be a distribution with full support on $\mathcal{T}_k$ and $\varphi_F^{k\text{-WL}}(\cdot)$ be the resulting random graph embedding where $F \sim \mathcal{D}$.

Theorem 16. *The graph embedding $\varphi_F^{k\text{-WL}}(\cdot)$ has the same expressiveness as the k-WL test in expectation for any $\mathcal{D}$ that has full support on $\mathcal{T}_k$. Furthermore, there is a specific such distribution $\mathcal{D}$ such that we can compute $\varphi_F^{k\text{-WL}}(G)$ in expected polynomial time $\mathcal{O}(v(G)^{k+1})$ for all $G \in \mathcal{G}_\infty$.*

Proposition 13 does not apply to the embedding $\varphi_F^{k\text{-WL}}(\cdot)$. In particular, the used distribution, which guarantees expected polynomial runtime, is independent of n and can be used for all $\mathcal{G}_\infty$.

As before, we can state Hoeffding-based bounds to approximate how close we are to the full embedding $\varphi_{\mathcal{T}_k}$. Morris et al. (2017) achieved similar bounds by sampling the k-tuple neighbourshoods of the k-WL test instead of the homomorphism counts.

Homomorphism-based graph embeddings. Dell et al. (2018) proposed a complete graph kernel based on homomorphism counts related to our $k_{\min}$ kernel. Instead of implicitly restricting the embedding to only a finite number of patterns, as we do, they weigh the homomorphism counts such that the inner product defined on the whole Lovász vectors converges. However, Dell et al. (2018) do not discuss how to compute their kernel and so, our approach can be seen as an efficient sampling-based alternative to their theoretical weighted kernel.

Using graph homomorphism counts as a feature embedding for graph learning tasks was proposed by Hoang & Maehara (2020) and Kühner (2021). Hoang & Maehara (2020) discuss various aspects of homomorphism counts important for learning tasks, in particular, universality aspects and their power to capture certain properties of graphs, such as bipartiteness. Instead of relying on sampling patterns, which we use to guarantee expectation in completeness, they propose to use a small number of fixed pattern graphs. This limits the practical usage of their approach due to computational complexity reasons. In their experiments the authors only use tree (GHC-tree(6)) and cycle patterns (GHC-cycle(8)) up to size 6 and 8, respectively, whereas we allow patterns of arbitrary size and treewidth, guaranteeing polynomial runtime in expectation. Similarly to Hoang & Maehara (2020), we use the computed embeddings as features for a kernel SVM (with RBF kernel) and an MLP. For first results using an SVM, see our preliminary work at Welke et al. (2022) and Thiessen et al. (2022).

Instead of embedding the whole graph into a vector of homomorphism counts, Barceló et al. (2021) proposed to use rooted homomorphism counts as node features in conjunction with a graph neural network (GNN). They discuss the required patterns to be as or more expressive than the k-WL test. We achieve this in expectation when selecting an appropriate sampling distribution, as discussed above.

Cut distance The distance induced by the Lovász vector of all homomorphism counts is strongly related to the *cut distance* (Borgs et al., 2006; Lovász, 2012). The cut distance is a well-studied and important distance on graphs that captures global structural but also sampling-based local information. It is well known that the distance given by homomorphism counts is close to the cut distance and hence has similar favourable properties. The cut distance, and hence homomorphism counts, capture the behaviour of all permutation-invariant functions on graphs. Using Corollary 10 we see that this also holds for random embeddings, as they converge to the distance induced by the Lovász vector with high probability. For a discussion on the importance of the cut distance and homomorphism counts in the context

[3]This refers to the *folklore* k-WL test, also called k-FWL.

of graph learning see Dell et al. (2018), Klopp & Verzelen (2019), Grohe (2020), and Hoang & Maehara (2020).

Random graph and node embeddings Wu et al. (2019) adapted random Fourier features (Rahimi & Recht, 2007) to graphs and proposed a sampling-based variant of the global alignment graph kernel. Similar sampling-based ideas were discussed before for the graphlet kernel (Shervashidze et al., 2009; Ghanem et al., 2021) and frequent-subtree kernels (Welke et al., 2015). The standard analysis of Rahimi & Recht (2007) does not apply in our situation, as they require a shift-invariant kernel. Also the analysis by Wu et al. (2019) does not apply here, as they use finite-dimensional node embeddings as a starting point. None of the previously mentioned papers discusses random graph features in the context of expressiveness or completeness. Fang et al. (2021) and Choromanski (2023) considered random features for node embeddings and node classification tasks.

Random node initialisation Instead of randomly embedding the whole graph, Abboud et al. (2021) and Sato et al. (2021) considered to initialise the vertices of the graphs with random labels. Through this they achieve universality in expectation. However, while for each realization of the random graph pattern F our graph embedding φ_F is universal in expectation and permutation-invariant, random node initialisation is only permutation-invariant in expectation.

Subgraph counts While subgraph counts are also a reasonable choice for expectation-complete graph embeddings, they have multiple drawbacks compared to homomorphism counts. Most importantly, from a computational perspective, computing subgraph counts even for graphs such as trees or paths is NP-hard (Alon et al., 1995; Marx & Pilipczuk, 2014), while we can compute homomorphism counts efficiently for pattern graphs with small treewidth (Díaz et al., 2002). In particular, all known exact algorithms for (induced) subgraph isomorphism counting have a worst-case runtime of $\mathcal{O}(v(G)^{v(F)})$, even for patterns with small treewidth. This one of the main reasons why the graphlet kernel (Shervashidze et al., 2009) and similar fixed pattern based approaches (Bouritsas et al., 2022) only count subgraphs up to size around 5 or are only sufficient. Alternative approaches exist, such as the cyclic pattern kernel (Horváth et al., 2004) and the neighbourhood-based kernel of Costa & De Grave (2010), that are efficiently computable in special cases, for example on most molecular graphs.

7. Empirical Evaluation

We analyze the performance of our expectation-complete embedding that can be computed in expected polynomial time. The details of the pattern sampling process are described in Appendix C. We evaluate our proposed embeddings in two contexts. We investigate how graph embeddings from message passing graph neural network (GNN) perform when augmented with our embeddings. To complement these results, we investigate the empirical expressive power of our embeddings on synthetic benchmark datasets. The code to sample patterns and to compute representations[4], as well as for the GNN experiments[5] is available.

Table 1. Performance of different GNNs on 9 OGB benchmarks and `ZINC`. Baseline of a GNN with homorphism counts is the same GNN without homomorphism counts. Results for GNNs with homorphism counts are averaged over 9 different random samples of pattern graphs.

	Top 1 / 2 / 3	Beats baseline
GIN	0% / 0% / 0%	-
GIN+hom	0% / 10% / 10%	100%
GCN	0% / 0% / 0%	-
GCN+hom	10% / 10% / 20%	90%
GIN+F	0% / 10% / 50%	-
GIN+hom +F	20% / 40% / 70%	90%
GCN+F	0% / 50% / 60%	-
GCN+hom+F	70% / 80% / 90%	90%

7.1. Improving GNNs with Graph-Level Homomorphism Counts

For graph-level prediction tasks, GNNs compute a graph embedding which is used by an MLP to make the final prediction. We propose to extend the learned graph embedding by concatenating it with a vector of homomorphism counts for a set of up to 50 sampled patterns $\mathcal{F}$ (cf. Section 5). As this approach is independent of the GNN it can boost the expressiveness of any GNN. Furthermore, it is possible to extend already trained GNNs by these patterns by simply changing the width of the MLP and fine tuning. We denote GNNs boosted by homomorphism counts by "GNN+hom". We compare two settings: with ("GNN+F") and without ("GNN") node and edge features. We determine whether our approach reliably boosts the prediction accuracy of GNNs.

Models. We use GIN (Xu et al., 2019) and GCN (Kipf & Welling, 2017) as baseline GNNs. We compare the baselines against GIN+hom and GCN+hom. When using homomorphism counts, we first train the model without these counts and then finetune the entire model with the full homomorphism vector. We normalize the vector of homomorphism counts such that each entry has 0 mean and a standard deviation of 1 over the training set. We base our hyperparameters on Hu et al. (2020) and tune only the dropout rate (for all hyperparameters see Table 4 in Appendix D). For models

[4]Representations: github.com/pwelke/homcount

[5]GNN evaluation: github.com/ocatias/HomCountGNNs

without homomorphism counts, we train and evaluate a model 10 times for the best hyperparameters. For models with homomorphism counts, we first find the best hyperparameters for one sample of homomorphism counts. Then, we train and evaluate the model with these hyperparameters for 8 different samples of pattern graphs and thus different homomorphism counts. For each model, we report the test result in the corresponding epoch with the best validation metric (see Appendix D). We report the average and standard deviation of all test results for a given type of model.

Setup. We evaluate on the commonly used molecule datasets `ZINC`, `ogbg-molhiv` and `ogbg-moltox21` (Hu et al., 2020). Furthermore, we also train on 7 additional small molecule datasets from Hu et al. (2020) (see Appendix D). For `ZINC` we use the same setup as Bodnar et al. (2021): we use a batch size of 128 and an initial learning rate of 10^{-3} which we reduce by half every 20 epochs without an improvement of the validation performance. We stop training after either 500 epochs or after the learning rate is smaller than 10^{-5}. To finetune on `ZINC`, we restart the training procedure with an initial learning rate of $5 \cdot 10^{-4}$. For datasets based on OGB, we train for 100 epochs with a batch size of 32 and a fixed learning rate of 10^{-3} which corresponds to the initial learning rate on `ZINC`. To finetune, we train for 100 additional epochs with a learning rate of $5 \cdot 10^{-4}$. We perform an ablation study in Appendix D.

Results. We summarize the results of the experiments in Table 1. The center column shows how often the best parameter setting for a variant (e.g. GIN+hom+F) was among the top 1, top 2, or top 3 scoring models among the ten datasets. Recall, that this references the predictive performance on the test in the epoch with the best performance on the validation set. We can immediately see that including homomorphism information is helpful for predictive performance as the best performing model for *every* dataset uses homomorphism counts. For each model, the rightmost column reports if a GNN variant with homomorphism counts beats its respective baseline GNN without added homomorphism counts. We can see that models with homomorphism counts outperform the baseline in at least 90% of the datasets. This demonstrates that besides theoretical guarantees, homomorphism counts also reliably improve the practical prediction performance of GNNs. Detailed results for all datasets and an ablation study can be found in Appendix D.

7.2. Expressiveness on Synthetic Datasets

We complement these results on real world graph datasets with an empirical analysis of our approach on synthetic benchmark datasets used to evaluate the expressiveness of graph learning approaches. On these benchmarks the labels encode isomorphism classes. Both datasets are balanced and have 10 (CSL) and 14 (PAULUS25) classes, respectively.

Table 2. Accuracy on synthetic data

Method	CSL	PAULUS25
GIN	10.00 ± 0.00	7.14 ± 0.00
GNTK	10.00 ± 0.00	7.14 ± 0.00
GHC-Tree	10.00 ± 0.00	7.14 ± 0.00
GHC-Cycle	100.0 ± 0.00	7.14 ± 0.00
WL	10.00 ± 0.00	7.14 ± 0.00
Ours	37.67 ± 9.11	100.0 ± 0.00

We sample a fixed number $\ell = 50$ of patterns and compute the sampled min kernel (resp. the corresponding embedding) as described in Section 3.2. Table 2 shows averaged accuracies of an SVM classifier trained on our feature sets on the datasets CSL (Murphy et al., 2019) and PAULUS25 (Hoang & Maehara, 2020)[6]. We follow the experimental design of Hoang & Maehara (2020) and compare to their published results. We also included GNTK (Du et al., 2019), GIN (Xu et al., 2019), and the WL-kernel (Shervashidze et al., 2011). Even with as little as 50 features, it is interesting to note that a SVM with RBF kernel and our features performs perfectly on the PAULUS25 dataset, i.e., it is able to decide isomorphism for the strongly regular graphs in this dataset. On the CSL dataset the min kernel performs better than all competitors except GHC-cycle, which was specifically designed for this dataset. The performance of the min kernel on this dataset increases monotonically for larger number of patterns, for instance to 48.8% for 200 patterns, see Appendix D.

8. Conclusion

In this work, we introduced the notion of expectation-complete graph embeddings—random embeddings, which in expectation can distinguish any pair of non-isomorphic graphs. We studied their general properties and have shown that repeated sampling will eventually allow us to distinguish any fixed pair of non-isomorphic graphs, which results in a universal representation for graphs of bounded size. We proposed to sample the Lovász vector of homomorphism counts as one possibility to achieve expectation-completeness and have shown favourable properties, such as bounds on the convergence of the random embedding to the full Lovász vector. Using a specific distribution which gives exponentially decreasing probability to patterns with large treewidth, we showed that computing our embedding takes polynomial time in expectation. We discussed that homomorphism counts of patterns with treewidth up to k can be seen as a sampling-based variant of the k-WL test with the same expressiveness in expectation and that homomorphism counts are strongly related to the cut-distance. Our

[6] Originally from www.distanceregular.org/graphs/paulus25.html

empirical results have shown that homomorphism counts of sampled patterns (a) tend to increase the performance of MPNNs on a set of benchmark datasets and (b) allow to learn classifiers that distinguish non-isomorphic graphs where MPNNs and other baselines fail.

As future work, we will investigate approximate counts to make our implementation more efficient (Beaujean et al., 2021). It is unclear how this affects expressiveness, as we loose permutation-invariance. Similar to Abboud et al. (2021) we would still retain permutation-invariance in expectation. Going beyond expressiveness results, our goal is to further study graph similarities suitable for graph learning, such as the cut distance as proposed by Grohe (2020). Finally, instead of sampling patterns from a fixed distribution, a more promising variant is to adapt the sampling process in a sample-dependent manner. One could, for example, draw new patterns until each graph in the sample has a unique embedding (up to isomorphism) or at least until we are at least as expressive as 1-WL on the given sample. Alternatively, we could pre-compute frequent or interesting patterns as proposed by Schulz et al. (2018) and use them to adapt the distribution. Such approaches would use the power of randomisation to select an appropriate graph embedding in a data-driven manner, instead of relying on a finite set of fixed and pre-determined patterns like previous work (Barceló et al., 2021; Bouritsas et al., 2022).

Acknowledgements

Part of this work has been funded by the Vienna Science and Technology Fund (WWTF) project ICT22-059 as well as by the Federal Ministry of Education and Research of Germany and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence, LAMARR22B.

References

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Alon, N., Yuster, R., and Zwick, U. Color-coding. *J. ACM*, 42(4):844–856, 1995.

Babai, L. Graph isomorphism in quasipolynomial time. In *STOC*, 2016.

Barceló, P., Geerts, F., Reutter, J., and Ryschkov, M. Graph Neural Networks with Local Graph Parameters. In *NeurIPS*, 2021.

Beaujean, P., Sikora, F., and Yger, F. Graph homomorphism features: Why not sample? In *Graph Embedding and Mining (GEM) Workshop at ECMLPKDD*, 2021.

Bodnar, C., Frasca, F., Otter, N., Wang, Y. G., Liò, P., Montúfar, G., and Bronstein, M. Weisfeiler and Lehman go cellular: CW networks. In *NeurIPS*, 2021.

Borgs, C., Chayes, J., Lovász, L., Sós, V. T., Szegedy, B., and Vesztergombi, K. Graph limits and parameter testing. In *STOC*, 2006.

Bouritsas, G., Frasca, F., Zafeiriou, S. P., and Bronstein, M. Improving graph neural network expressivity via subgraph isomorphism counting. *Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Cai, J.-Y., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

Caminiti, S., Fusco, E. G., and Petreschi, R. Bijective linear time coding and decoding for k-trees. *Theory of Computing Systems*, 46(2):284–300, 2010.

Choromanski, K. Taming graph kernels with random features. In *ICML*, 2023.

Costa, F. and De Grave, K. Fast neighborhood subgraph pairwise distance kernel. In *ICML*, 2010.

Curticapean, R., Dell, H., and Marx, D. Homomorphisms are a good basis for counting small subgraphs. In *STOC*, 2017.

Dell, H., Grohe, M., and Rattan, G. Lovász meets Weisfeiler and Leman. In *ICALP*, 2018.

Du, S. S., Hou, K., Salakhutdinov, R. R., Poczos, B., Wang, R., and Xu, K. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *NeurIPS*, 2019.

Dvořák, Z. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory*, 64(4):330–342, 2010.

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

Fang, J., Zhang, Q., Meng, Z., and Liang, S. Structure-aware random fourier kernel for graphs. *Advances in Neural Information Processing Systems*, 34:17681–17694, 2021.

Gärtner, T., Flach, P. A., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *COLT*, 2003.

Ghanem, H., Keriven, N., and Tremblay, N. Fast graph kernel with optical random features. In *International Conference on Acoustics, Speech and Signal Processing*, 2021.

Grohe, M. Word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In *PODS*, 2020.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Hoang, N. and Maehara, T. Graph homomorphism convolution. In *ICML*, 2020.

Horváth, T., Gärtner, T., and Wrobel, S. Cyclic pattern kernels for predictive graph mining. In *KDD*, 2004.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open Graph Benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Klopp, O. and Verzelen, N. Optimal graphon estimation in cut distance. *Probability Theory and Related Fields*, 174: 1033–1090, 2019.

Kontorovich, L. A. and Nadler, B. Universal kernel-based learning with applications to regular languages. *Journal of Machine Learning Research*, 10(39):1095–1129, 2009.

Kriege, N. M., Johansson, F. D., and Morris, C. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

Kühner, P. Graph embeddings based on homomorphism counts. Master's thesis, RWTH Aachen, 2021.

Lovász, L. Operations with structures. *Acta Mathematica Hungaria*, 18:321–328, 1967.

Lovász, L. *Large Networks and Graph Limits*, volume 60 of *Colloquium Publications*. American Mathematical Society, 2012.

Marx, D. and Pilipczuk, M. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). In *International Symposium on Theoretical Aspects of Computer Science*, 2014.

Morris, C., Kersting, K., and Mutzel, P. Glocalized Weisfeiler-Lehman graph kernels: Global-local feature maps of graphs. In *ICDM*, 2017.

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 *AAAI*, 2019.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In *ICML*, 2019.

Nie, S., Campos, C. P. d., and Ji, Q. Learning bounded tree-width Bayesian networks via sampling. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 2015.

Puny, O., Lim, D., Kiani, B. T., Maron, H., and Lipman, Y. Equivariant polynomials for graph neural networks. In *ICML*, 2023.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *NIPS*, 2007.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *ICDM*, 2021.

Schulz, T. H., Horváth, T., Welke, P., and Wrobel, S. Mining tree patterns with partially injective homomorphisms. In *ECMLPKDD*, 2018.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 2009.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Thiessen, M., Welke, P., and Gärtner, T. Expectation complete graph representations using graph homomorphisms. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

Welke, P., Horváth, T., and Wrobel, S. Probabilistic frequent subtree kernels. In *Workshop on New Frontiers in Mining Complex Patterns at ECMLPKDD*, 2015.

Welke, P., Thiessen, M., and Gärtner, T. Expectation complete graph representations using graph homomorphisms. In *Learning on Graphs Conference*, 2022.

Wu, L., Yen, I. E.-H., Zhang, Z., Xu, K., Zhao, L., Peng, X., Xia, Y., and Aggarwal, C. Scalable global alignment graph kernel using random features: From node embedding to graph embedding. In *KDD*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Yoo, J., Kang, U., Scanagatta, M., Corani, G., and Zaffalon, M. Sampling subgraphs with guaranteed treewidth for accurate and efficient graphical inference. In *WSDM*, 2020.

A. Proofs

Lemma 11. *$k_{\min}$ is a complete kernel on $\mathcal{G}_\infty$.*

Proof. Let $G, G' \in \mathcal{G}_\infty$. We have to show that

$$\varphi_F^{\downarrow}(G) = \varphi_F^{\downarrow}(G') \Leftrightarrow G \simeq G' \ ,$$

We start by the $\Rightarrow$ direction. There are two cases:

1. $v(G) = v(G')$:

   Then, by Theorem 5 we have $\varphi_n(G) = \varphi_n(G')$ iff $G \simeq G'$ for $n = \min\{v(G), v(G')\} = v(G) = v(G')$.

2. $v(G) \neq v(G')$:

   Let w.l.o.g. $0 < v(G) < v(G')$. Let P be the graph with exactly one vertex. Then $\hom(P, G) < \hom(P, G')$, i.e., we can distinguish graphs on different numbers of vertices using homomorphism counts. As $\min\{v(G), v(G')\} \geq 1$, we have $P \in \mathcal{G}_{v(G)}$ and hence $\varphi_{v(G)}(G) \neq \varphi_{v(G)}(G')$.

The $\Leftarrow$ direction follows directly from the fact that homomorphism counts are invariant under isomorphism. □

Theorem 12. *Let $\mathcal{D}$ be a distribution on $\mathcal{G}_\infty$ with full support and $G \in \mathcal{G}_\infty$. Then $\varphi_F^{\downarrow}(G) = \hom_{v(G)}(F, G) e_F$ with $F \sim \mathcal{D}$ and the corresponding kernel are expectation-complete.*

Proof. We can apply the same arguments as before from Theorem 7 to show that the expected embeddings of two graphs G, G' with size $n' = \min\{v(G), v(G')\}$ are equal iff their Lovász vector restricted to size n' are equal. By Lemma 11 we know that the latter only can happen if the two graphs are isomorphic. □

Proposition 13. *There exists no distribution $\mathcal{D}$ with full support on $\mathcal{G}_\infty$ such that the expected runtime of Eq. (2) becomes polynomial in $v(G)$ for all $G \in \mathcal{G}_\infty$.*

Proof. Let $\mathcal{D}$ be such a distribution and let $\mathcal{D}'$ be the marginal distribution on the treewidths of the graphs given by $p_k = \Pr_{F \sim \mathcal{D}}(\mathrm{tw}(F) = k) > 0$. Let G be a given input graph in the sample. Díaz et al. (2002) have shown that computing $\hom(F, G)$ takes time $\mathcal{O}\left(v(F) v(G)^{\mathrm{tw}(F)+1}\right)$. Assume for the purpose of contradiction that we can guarantee an expected polynomial runtime (ignoring the $v(F)$ term and constant factors for simplicity):

$$\mathbb{E}_{F \sim \mathcal{D}}[v(G)^{\mathrm{tw}(F)+1}] = \sum_{k=1}^{\infty} p_k v(G)^{k+1} \leq C v(G)^c$$

for some constants $C, c \in \mathbb{N}$. Then for all $k \geq c$, it must hold that $p_k v(G)^{k+1} \leq C v(G)^c$, as all summands are positive. However, for large enough $v(G)$ the left hand side is larger than the right hand side. Contradiction. □

Theorem 14. *There exists a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ such that computing the expectation-complete graph embedding $\varphi_F^{\downarrow}(G)$ with $F \sim \mathcal{D}$ takes polynomial time in $v(G)$ in expectation for all $G \in \mathcal{G}_n$.*

Proof. We draw a treewidth upper bound k from a Poisson distribution with parameter λ to be determined later. Select a distribution $\mathcal{D}_{n,k}$ which has full support on all graphs with treewidth up to k and size up to n, for example, the one described in Appendix C. Let $G \in \mathcal{G}_n$. Note that $\mathcal{D}_{n,k}$ is a fixed distribution for $\mathcal{G}_n$ and independent of G. Also, $\varphi_F^{\downarrow}(G) = 0$ for all patterns F with $v(F) > v(G)$ by definition of $\varphi^{\downarrow}$. Hence, in this case we do not have to run the homomorphism counting computation. Thus, we can restrict the support of $\mathcal{D}_{n,k}$ to $\mathcal{D}_{v(G),k}$. Overall, the runtime is determined by the homomorphism counting. Using the algorithm of (Díaz et al., 2002) this gives, for some constant $C \in \mathbb{N}$, an expected runtime of

$$\begin{aligned}
\mathbb{E}_{k \sim \mathrm{Poi}(\lambda), F \sim \mathcal{D}_{v(G),k}} \left[C v(F) v(G)^{\mathrm{tw}(F)+1} \right] &\leq \mathbb{E}_{k \sim \mathrm{Poi}(\lambda)} \left[C v(G)^{k+2} \right] \\
&= \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} C v(G)^{k+2} = \frac{C v(G)^2}{e^{\lambda}} e^{\lambda v(G)}.
\end{aligned}$$

We need to bound the right hand side by some polynomial $Dv(G)^d$ for some constants $D, d \in \mathbb{N}$. By rearranging terms we see that

$$\lambda \leq \frac{\ln \frac{D}{C} + (d-2)\ln v(G)}{v(G)-1}$$

is sufficient. To satisfy this inequality for all graphs in $\mathcal{G}_n$ simultaneuosly (meaning for all possible graph sizes up to n) we choose

$$\lambda \leq \frac{\ln \frac{D}{C} + (d-2)\ln n}{n-1},$$

which is valid as the right hand side is monotonically decreasing in $v(G)$.

□

Theorem 15. *There exists a distribution $\mathcal{D}$ with full support on $\mathcal{G}_n$ such that computing the expectation-complete graph embedding $\varphi_F(G)$ with $F \sim \mathcal{D}$ takes polynomial time in $v(G)$ in expectation for all $G \in \mathcal{G}_n$.*

Proof. The proof proceeds almost exactly as the one for Theorem 14. However, to guarantee a runtime polynomial in $\mathcal{O}(v(G))$ instead of merely $\mathcal{O}(n)$ we have to proceed slightly more careful.

As before in Theorem 14 we draw k, the treewidth upper bound, from a Poisson distribution with parameter λ and F from a distribution $\mathcal{D}_{n,k}$. Here, we additionally, require that $\mathbb{E}_F[v(F)]$ is bounded by a constant (while still having full support on the graphs of treewidth up to k and size up to n), which is easily satisfied by for example a Geometric distribution (with its parameter set to some constant) restricted to $\{1, \ldots, n\}$. We see that

$$\begin{aligned}
\mathbb{E}_{k\sim\mathrm{Poi}(\lambda),F\sim\mathcal{D}_{n,k}}[v(F)v(G)^{\mathrm{tw}(F)+1}] &= \sum_{i=1}^{\infty}\sum_{F'\in\mathcal{G}_n} \Pr(k=i, F=F')v(F)v(G)^{\mathrm{tw}(F)+1} \\
&\leq \sum_{i=1}^{\infty}\sum_{F'\in\mathcal{G}_n} \Pr(k=i)\Pr(F=F'|k=i)v(F)v(G)^{k+1} \\
&= \sum_{i=1}^{\infty}\Pr(k=i)v(G)^{k+1}\sum_{F'\in\mathcal{G}_n}\Pr(F=F'|k=i)v(F) \\
&= \sum_{i=1}^{\infty}\Pr(k=i)v(G)^{k+1}\mathbb{E}_{F\sim\mathcal{D}_{n,i}}[v(F)]\,.
\end{aligned}$$

As the expectation of $v(F)$ is by assumption bounded by a constant, it remains to upper bound $\sum_{i=1}^{\infty}\Pr(k=i)v(G)^{k+1}$ by a polynomial in $v(G)$. This can be achieved by choosing λ as before in Theorem 14. □

B. Matching the Expressiveness of k-WL in Expectation

We devise a graph embedding matching the expressiveness of the k-WL test in expecation.

Theorem 16. *The graph embedding $\varphi_F^{k\text{-}WL}(\cdot)$ has the same expressiveness as the k-WL test in expectation for any $\mathcal{D}$ that has full support on $\mathcal{T}_k$. Furthermore, there is a specific such distribution $\mathcal{D}$ such that we can compute $\varphi_F^{k\text{-}WL}(G)$ in expected polynomial time $\mathcal{O}(v(G)^{k+1})$ for all $G \in \mathcal{G}_\infty$.*

Proof. Let $\mathcal{T}_k$ be the set of graphs with treewidth up to k and $\mathcal{D}$ be a distribution with full support on $\mathcal{T}_k$. Then by the same arguments as before in Theorem 7, the expected embeddings of two graphs G and G' are equal iff their Lovász vectors restricted to patterns in $\mathcal{T}_k$ are equal. By Dvořák (2010) and Dell et al. (2018) the latter happens iff k-WL returns the same color histogram for both graphs. This proves the first claim.

For the second claim note that the worst-case runtime for any pattern $F \in \mathcal{T}_k$ is $\mathcal{O}\left(v(F)v(G)^{k+1}\right)$ by Díaz et al. (2002). However, the equivalence between homomorphism counts on $\mathcal{T}_k$ and k-WL requires to inspect also patterns F of all sizes, in particular, also larger than the size n of the input graph. To remedy this, we can draw the pattern size $m = v(F)$ from

Algorithm 1 Sampling algorithm for a pattern set

input: the maximum graph size n, a number ℓ of requested patterns
output: a list P of ℓ patterns

Initialize $P = \{\text{singleton, edge, wedge, triangle}\}$
for $i = 5$ to ℓ **do**
  Draw pattern size $N \sim \mathrm{Geom}(1 - \sqrt[n-3]{0.01}) + 3$
  Draw pattern treewidth $k \sim \mathrm{Poi}(\frac{1+\log(n)}{n}) + \mathcal{U}(3)$
  $k = \min(k, N-1)$ // maximum treewidth of graph on N vertices is $N-1$
  Sample a maximal graph F with treewidth k and N vertices
  for $e \in E(F)$ **do**
    Remove e from F with probability $p = 0.1$
  end for
  Add F to P
end for
return P

some distribution with bounded expectation and full support on $\mathbb{N}$. For example, a geometrically distributed $m \sim \mathrm{Geom}(p)$ with any constant parameter $p \in (0,1)$ and expectation $\mathbb{E}[m] = \frac{1}{1-p}$ is sufficient. By linearity of expectation then

$$\begin{aligned}\mathbb{E}_{F\sim\mathcal{D}}\left[v(F)v(G)^{\mathrm{tw}(F)+1}\right] &\leq \mathbb{E}_{F\sim\mathcal{D}}\left[v(F)v(G)^{k+1}\right] \\ &= \mathbb{E}_{F\sim\mathcal{D}}\left[v(F)\right]v(G)^{k+1} \\ &= \mathcal{O}\left(v(G)^{k+1}\right).\end{aligned}$$

□

C. Sampling Details

To obtain a practical sampling algorithm for unlabeled graphs that draws from a distribution with full support on $\mathcal{G}_n$, we proceed as follows: We first draw a pattern size $N \leq n$ from some distribution $\mathcal{D}_1$ and then draw a treewidth upper bound $k < N$ from some distribution $\mathcal{D}_2$. Then we want to sample any graph with treewidth at most k and N vertices with a nonzero probability. While guaranteed uniform sampling would be preferable, we resort to a simple sampling scheme that is easy to implement. A natural strategy is to first sample a k-*tree*, which is a maximal graph with treewidth k, and then take a random subgraph of it. Uniform sampling of k-trees is described by Nie et al. (2015) and Caminiti et al. (2010). Alternatively, the strategy of Yoo et al. (2020) is also possible. Note that we only have to guarantee that each pattern has a nonzero probability of being sampled; it does not have to be uniform. We achieve a nonzero probability for each pattern of at most a given treewidth k by first constructing a random k-tree P through its tree decomposition, by uniformly drawing a tree T on $N-k$ vertices and choosing a root. We then create P as the (unique up to isomorphism) k-tree that has T as tree decomposition. We then randomly remove edges from P i.i.d. with fixed probability $p_{\text{removal}} > 0$. This ensures that each subgraph of P will be created with nonzero probability.

We choose $\mathcal{D}_1$ as a geometric distribution with parameter $p = 1 - \sqrt[n]{0.01}$ to ensure that $N \leq n$ with probability 0.99. While we require in Theorem 15 that the expectation of N is bounded by a constant to satisfy an expected runtime that is polynomial in each graph size $v(G)$, this sampling only guarantees a runtime polynomial in n (the upper bound on all graphs in the training sample), as $v(F)$ goes linearly in the runtime and could exceed $v(G)$. For the whole training sample this still has an expected polynomial runtime. The min kernel, however, can be computed in polynomial runtime in $v(G)$ in expectation for this sampling scheme. To achieve polynomial runtime in expectation, we choose $\mathcal{D}_2$ as a Poisson distribution with parameter $\lambda = \frac{1+\log(n)}{n}$, where we add a number from the set $\{1,2,3\}$ (drawn uniformly at random) to the outcome. This ensures polynomial runtime in expectation, while it increases the probability of drawing nontrivial treewidths. Finally, we set $p_{\text{removal}} = 0.1$. This sampling scheme assigns high probability to small graphs with low treewidth. When drawing multiple patterns (e.g. $l = 50$ as in our experiments), we observe that small patterns of size up to three are typically drawn multiple times. We don't filter out isomorphic patterns to maintain expected polynomial time. Instead, to practically improve

Table 3. Average accuracy of an SVM using our min kernel on CSL for varying number of patterns l over ten independent samples of patterns.

ℓ	Accuracy
20	28.3% ± 6.7%
50	35.0% ± 8.6%
100	38.7% ± 8.4%
150	44.6% ± 6.2%
200	48.8% ± 3.4%

Table 4. Hyperparameter grid used for all models.

PARAMETER ↓	VALUES ↓
EMBEDDING DIMENSION	300
NUMBER OF GNN LAYERS	5
NUMBER OF MLP LAYERS	2
DROPOUT RATE	0, 0.5
POOLING OPERATION	MEAN

the pattern distribution, we include the four nonisomorphic patterns up to size three (singleton, edge, wedge, and triangle) deterministically and draw the remaining pattern sizes from $\tilde{\mathcal{D}}_1 = \mathrm{Geom}(1 - \sqrt[n-3]{0.01}) + 3$. Algorithm 1 summarizes the sampling process for a fixed number of ℓ patterns.

D. Experimental Details

Our source code for the pattern sampling and homomorphism counting is available on github[7]. The repository additionally contains the evaluation code for the synthetic datasets, with an SVM using an RBF kernel and started as a fork of the code of Hoang & Maehara (2020)[8]. We rely on the C++ code of Curticapean et al. (2017)[9] to efficiently compute homomorphism counts. While the code computes a tree decomposition itself we decided to simply provide it with our tree decomposition of the k-tree which we compute as part of the sampling process, to make the computation more efficient[10]. The datasets in the correct format as well as the sampled graph representations used for the evaluation in this paper can be downloaded[11] to reproduce the experiments in this paper.

The source code for the GNN evaluation is available in a separate github repository[12]. We implement our models in PyTorch and PyTorch Geometric and train on a single NVIDIA GeForce RTX 3080 GPU. We evaluate on the molecular graph level tasks from the Open graph Benchmark (Hu et al., 2020) and on the `ZINC` subset dataset provided by Gómez-Bombarelli et al. (2018). We use the provided train/validate/test splits. Table 4 shows the values of the hyperparameters used for each of the ten datasets. The hyperparameters are based on Hu et al. (2020) with the difference that we are using two layers in the final MLP instead of one as we have found this to yield significantly better predictions in preliminary experiments. Table 5 shows the relevant metrics and results for all datasets.

Ablation Study. We perform an ablation study to investigate whether improvements from homomorphism counts are due to the homomorphism counts or because we finetune the model for more epochs. For this, we train GIN with misaligned homomorphism counts i.e., counts that were computed for a different graph. We denote this as GIN+MIS. We compare the performance of baselines against models with (misaligned) features on all datasets with features except `ZINC` and `ogbg-molhiv` due to their large size. Table 6 shows the results of the ablations. We see (1) that GIN with homomorphism counts always outperforms GIN with misaligned features. Interestingly, (2) GIN with misaligned features outperforms the baseline GNN in 7 out of 8 datasets. From (2) it follows that some of the improvements of using homomorphism counts is dude to the additional finetuning. However, from (1) it follows that much of improvement of using homomorphism counts is due to the homomorphism counts and not because of the finetuning.

Stability and Increasing Sample Size on CSL We investigate the stability of predictive performance over independent pattern sets as well as the performance of the min kernel for larger samples. We repeat the experiment on CSL shown in Table 2 ten times each for varying number of sampled patterns $\ell \in \{20, 50, 100, 150, 200\}$. Table 3 shows the results. The

[7]Pattern sampling and representations: github.com/pwelke/homcount
[8]Code of Hoang & Maehara: github.com/gear/graph-homomorphism-network
[9]Homomorphism counting: github.com/ChristianLebeda/HomSub
[10]Adapted version of homomorphism counting: github.com/pwelke/HomSub
[11]Links to download datasets and graph representations: github.com/pwelke/homcount
[12]GNN evaluation: github.com/ocatias/HomCountGNNs

Table 5. Results of different GNNs on molecular datasets. **Bold** results are GNNs with homomorphism counts that are better than the same GNN without homomorphism counts. Results with homomorphism counts are averaged over 9 different samples of pattern graphs.

DATA SET → ↓ MODEL	MOLBACE roc-auc ↑	MOLCLINTOX roc-auc ↑	MOLBBBP roc-auc↑	MOLSIDER roc-auc ↑	MOLTOXCAST roc-auc ↑
GIN	82.2 ± 2.0	61.2 ± 4.5	60.9 ± 2.4	57.5 ± 1.4	57.1 ± 0.8
GIN+hom	$\mathbf{82.7 \pm 1.8}$	$\mathbf{61.5 \pm 4.1}$	$\mathbf{63.0 \pm 1.1}$	$\mathbf{58.4 \pm 1.2}$	$\mathbf{58.1 \pm 0.5}$
GCN	81.4 ± 2.4	68.4 ± 3.6	59.2 ± 1.0	58.2 ± 1.3	58.6 ± 0.6
GCN+hom	$\mathbf{84.6 \pm 1.3}$	63.4 ± 4.7	$\mathbf{61.2 \pm 0.7}$	$\mathbf{59.2 \pm 1.2}$	$\mathbf{59.4 \pm 0.4}$
GIN+F	75.5 ± 3.0	84.8 ± 3.7	67.2 ± 1.5	57.7 ± 1.8	61.8 ± 0.8
GIN+hom+F	$\mathbf{76.4 \pm 2.6}$	$\mathbf{86.9 \pm 3.5}$	$\mathbf{68.8 \pm 1.3}$	$\mathbf{58.4 \pm 1.5}$	$\mathbf{63.2 \pm 0.8}$
GCN+F	82.2 ± 1.4	88.2 ± 3.0	66.4 ± 2.6	59.3 ± 1.6	65.7 ± 0.4
GCN+hom+F	81.3 ± 1.6	$\mathbf{90.4 \pm 2.0}$	$\mathbf{70.8 \pm 1.2}$	$\mathbf{60.0 \pm 1.9}$	$\mathbf{65.8 \pm 0.8}$
	MOLLIPO rmse ↓	MOLTOX21 roc-auc ↑	MOLESOL rsmse ↓	MOLHIV roc-auc ↑	ZINC mae ↓
GIN	1.062 ± 0.025	65.4 ± 1.9	1.852 ± 0.044	69.1 ± 2.2	1.262 ± 0.017
GIN+hom	$\mathbf{1.006 \pm 0.017}$	$\mathbf{67.5 \pm 1.1}$	$\mathbf{1.746 \pm 0.096}$	$\mathbf{71.0 \pm 1.9}$	$\mathbf{1.231 \pm 0.014}$
GCN	1.056 ± 0.035	66.7 ± 0.7	1.855 ± 0.073	69.1 ± 2.2	1.281 ± 0.013
GCN+hom	$\mathbf{0.986 \pm 0.015}$	$\mathbf{66.8 \pm 1.1}$	$\mathbf{1.735 \pm 0.066}$	$\mathbf{72.2 \pm 1.4}$	$\mathbf{1.26 \pm 0.014}$
GIN+F	0.739 ± 0.019	75.4 ± 0.9	1.197 ± 0.061	76.5 ± 2.0	0.208 ± 0.005
GIN+hom+F	$\mathbf{0.71 \pm 0.021}$	75.2 ± 0.8	$\mathbf{1.014 \pm 0.044}$	$\mathbf{77.7 \pm 1.5}$	$\mathbf{0.174 \pm 0.005}$
GCN+F	1.188 ± 1.387	77.2 ± 0.6	1.197 ± 0.069	78.3 ± 1.0	0.234 ± 0.007
GCN+hom+F	$\mathbf{0.816 \pm 0.282}$	$\mathbf{78.0 \pm 0.6}$	$\mathbf{0.991 \pm 0.045}$	$\mathbf{78.8 \pm 1.3}$	$\mathbf{0.207 \pm 0.008}$

target of the prediction task on CSL is the isomorphism class of the input graph. Our theory implies that the expressiveness of the representations increases with number of samples. We indeed see that the performance on CSL increases with number of sampled patterns. Furthermore, the variance over different sets drops with increasing number of sampled patterns. Thus, at least on CSL, the performance of an SVM using the min kernel is not very sensitive to the pattern set and increases with the number of samples. However, we note that we could design a distribution over cycle graph patterns alone which would allow to perfectly distinguish the isomorphism classes on CSL.

Table 6. Ablations of GIN with misaligned homomorphism counts (GIN+MIS+F) against GIN without homomorphism counts (GIN+F) and GIN with homomorphism counts(homGNNGIN+F). Results with (misaligned) homomorphism counts are averaged over 9 different samples of pattern graphs. **Bold** results are GIN with (misaligned) homomorphism counts that are better than GIN without homomorphism counts. Red results are GIN+hom+F that outperforms GIN+MIS+F.

DATA SET → ↓ MODEL	MOLBACE roc-auc ↑	MOLCLINTOX roc-auc ↑	MOLBBBP roc-auc↑	MOLSIDER roc-auc ↑	MOLTOXCAST roc-auc ↑
	ogbg-molbace	ogbg-molclintox	ogbg-molbbbp	ogbg-molsider	ogbg-moltoxcast
GIN+F	75.5 ± 3.0	84.8 ± 3.7	67.2 ± 1.5	57.7 ± 1.8	61.8 ± 0.8
GIN+MIS+F	$\mathbf{76.3 \pm 3.3}$	$\mathbf{86.0 \pm 3.6}$	$\mathbf{67.9 \pm 2.3}$	$\mathbf{58.3 \pm 1.9}$	$\mathbf{62.7 \pm 0.9}$
GIN+hom+F	$\mathbf{76.4 \pm 2.6}$	$\mathbf{86.9 \pm 3.5}$	$\mathbf{68.8 \pm 1.3}$	$\mathbf{58.4 \pm 1.5}$	$\mathbf{63.2 \pm 0.8}$
	MOLLIPO rmse ↓	MOLTOX21 roc-auc ↑	MOLESOL rsmse ↓		
	ogbg-mollipo	ogbg-moltox21	ogbg-molesol		
GIN+F	0.739 ± 0.019	75.4 ± 0.9	1.197 ± 0.061		
GIN+MIS+F	$\mathbf{0.732 \pm 0.017}$	74.9 ± 0.9	$\mathbf{1.175 \pm 0.067}$		
GIN+hom+F	$\mathbf{0.71 \pm 0.021}$	75.2 ± 0.8	$\mathbf{1.014 \pm 0.044}$		