Rethinking Local Structural Awareness in Graph Representation Learning

Emily Jin

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2025

Acknowledgements

First and foremost, I would like to thank my supervisors, Michael Bronstein and İsmail Ceylan, for their constant support and guidance throughout my DPhil. Without their patience, insight, and encouragement, this work would not have been possible. I would also like to thank the University of Oxford, SABS CDT, and AstraZeneca for helping to fund my research, and more specifically Stephen Bonner, Edward Morrissey, and Cheng Ye for giving me the freedom to pursue my own interests and always providing a welcoming space for discussion. I am especially grateful to Matthias Lanzinger and Chenghao Liu, who have not only set the gold standard as collaborators, but have also become lifelong friends. To all my lab mates, thank you for creating a fun space for all those late nights and deadlines.

On a more personal note, I am grateful to all my friends here at Oxford who have helped create the amazing memories we've shared, and to all my friends from home, who have cheered me on from across the pond. To Arian, thank you for being someone I can always rely on, no matter what challenges arise. Finally, I am especially thankful to my family for their unconditional love and support throughout my entire life. To my amazing parents, Hua and Xin, you continue to serve as an inspiration to me every day. To my brother Olivier and his wife Connie, I can't wait to see you both become the wonderful parents I know you will be. And to my new niece Danica, I love you so much.

Abstract

Graph-structured data lies at the heart of many modern applications, from molecular modeling and materials discovery to social networks and relational databases. For these types of problems, graph representation learning has emerged as a powerful framework for reasoning. Models such as graph neural networks (GNNs) are particularly appealing due to their ability to capture complex, implicit relationships. However, a key open question remains in how to best encode and generalize local structural information within these models. This affects both the ability of current methods to intelligently reason over structure in data as well as their applicability to more intricate, real-world scenarios. In this thesis, we examine existing methods for representing local structure in GNNs and propose two novel approaches for enhancing their representational capacity. The first uses graph homomorphism counts as a unifying framework for improving model expressivity, and the second introduces a new paradigm for leveraging local information to capture global symmetries in periodic data to enable large-scale, efficient learning. Together, these contributions advance our understanding of local structural representations in GNNs and expand their practical utility.

Publications

This thesis is based on the following publications [1–3]:

1. **Emily Jin**, Michael Bronstein, İsmail İlkan Ceylan, Matthias Lanzinger. Homomorphism Counts for Graph Neural Networks: All About That Basis. In *ICML*. 2024.

2. Linus Bao*, **Emily Jin***, Michael Bronstein, İsmail İlkan Ceylan, Matthias Lanzinger. Homomorphism Counts as Structural Encodings for Graph Learning. In *ICLR*. 2025. *Equal Contribution.

3. **Emily Jin***, Andrei Nica*, Mikhail Galkin, Jarrid Rector-Brooks, Kelvin Lee, Santiago Miret, Frances Arnold, Michael Bronstein, Joey Bose, Alex Tong, Chenghao Liu. OXtal: An All-Atom Diffusion Model for Organic Crystal Structure Prediction. In *ICLR*. 2026. *Equal Contribution.

During my DPhil, I also published the following journal paper, which is not included in this thesis [4]:

4. **Emily Jin**, J Alison Noble, Mireille Gomes. A Review of Computer-Aided Diagnostic Algorithms for Cervical Neoplasia and an Assessment of Their Applicability to Female Genital Schistosomiasis. In *Mayo Clinic Proceedings: Digital Health* 1.3 (2023).

Contents

1 Introduction

Modern advances in deep learning have transformed the field of machine learning, where neural network models have achieved remarkable success in long-standing challenges such as machine translation [5, 6], image classification [7–9], and protein folding [10–12]. The ability of deep learning methods to extract patterns from high-dimensional data and generalize beyond raw training samples has established them as a powerful framework for approximating complex, non-linear functions. Motivated by these successes, *graph representation learning* has emerged as a prominent research direction aimed at extending this paradigm to more complex, non-Euclidean, *graph-structured* data [13, 14].

Graph-structured data is ubiquitous across an array of scientific and technological domains [14]. For example, in social networks, nodes may represent individuals within a community, while edges indicate social relationships or interactions. In computational chemistry, molecules can naturally be modelled as graphs, where atoms correspond to nodes and chemical bonds correspond to edges. Even in computer vision, images can be interpreted as special graphs, where adjacent pixels are connected to form a regular grid.

At its core, the goal of graph representation learning is to develop methods that effectively encode the structural and relational information inherent within graph-structured data to enable efficient learning on downstream tasks [13]. In particular, it seeks to map nodes, edges, or entire graphs into low-dimensional vector spaces—referred to as *embeddings*—that preserve the relevant topological or semantic properties of the original graph. These learned embeddings can then be leveraged to address relevant challenges across a wide range of applications, including recommender systems [15], drug discovery [16, 17], and materials science [18].

One prominent family of models that has shown significant promise in this field are graph neural networks (GNNs) [19–21]. Most GNNs operate by iteratively propagating and transforming information along the edges of a graph, enabling each node to update its representation based on both its own features and the features of its neighbors [22]. This iterative process—referred to as *message passing*—allows nodes to accumulate information (or "messages") from increasingly larger neighborhoods, thereby capturing both local and global structural dependencies within the graph.

Another key characteristic of GNNs is their ability to respect the inherent symmetries of graph data. Unlike images or sequences, where the order of elements is fixed, graphs lack any canonical ordering of nodes. To ensure that learned representations are consistent regardless of how the graph is indexed, GNN architectures enforce permutation invariance through symmetric aggregation functions such as summation, averaging, or maximization [19, 20, 23]. This permutation invariance allows GNNs to generalize effectively across graphs of varying sizes and structures. In other words, GNNs are capable of *inductive reasoning*, meaning they are able to generalize to unseen nodes in a graph.

Building on this idea of symmetry, *equivariant* graph neural networks extend these principles of invariance beyond node permutations to encompass broader geometric transformations, such as rotations, translations, and reflections in Euclidean space (commonly referred to as the E(3) space group) [24]. These models are designed so that their outputs transform predictably when the input graph is transformed — a property known as *equivariance*. By embedding physical or spatial symmetries directly into the network architecture, equivariant GNNs are especially useful for learning on geometric or physical systems, such as molecular structures or point clouds. This explicit handling of symmetry ensures that learned representations remain consistent with the underlying laws of geometry and physics.

Despite their success and prevalence in a variety of fields, GNNs still face several theoretical and practical limitations. One such theoretical limitation pertains to their *expressivity*. The expressivity of a GNN is defined by its ability to distinguish (i.e. generate different representations for) two different non-isomorphic graphs [23]. Critically, standard message-passing GNNs are unable to count the occurrences of basic graph patterns, such as paths or cycles. This limitation poses a significant bottleneck for several applications, especially within the molecular domain, where the presence of rings or other functional groups govern the physical properties of a molecule.

Efforts to overcome these constraints have inspired many different research directions. One particularly rich body of work focuses on improving the expressive power of GNNs through techniques such as higher-order message passing [25], or enriching node features to produce subgraph-based [26] or homomorphism-based [27] architectures. These methods aim to capture more complex structural dependencies that conventional message-passing schemes often miss. However, many of these approaches often suffer from substantial scalability limitations or depend on ad hoc, trial-and-error feature engineering to achieve good performance. As a result, they frequently fall short when applied to domain-specific problems that demand both interpretability and computational efficiency.

Another approach is to expand beyond the local message-passing paradigm and use architectures that can capture global dependencies and relational hierarchies without being restricted to local neighborhood aggregation. Within this category, *graph transformers* have emerged as a compelling alternative [28–30]. Inspired by the success of the Transformer architecture in natural language processing [6], graph transformers replace localized message passing with global attention mechanisms that allow each node to "attend" to every other node in the graph, weighted by a learned relevance score [28]. This attention mechanism enables flexible long-range communication, offering a potential route to overcome the locality limitations of traditional GNNs.

However, applying transformers directly to graphs introduces new challenges. Graph transformers can be seen as a special case of GNNs that operate over complete graphs [31], which consequently sacrifices the underlying node adjacency information of the original graph. To recover the original graph structure, graph transformers rely on enhancing node features with effective positional or structural encoding vectors. Early approaches employed random walk structural encodings (RWSE) [32] or Laplacian eigenvectors (LapPE) [33], but these encodings often capture only coarse relational patterns and fail to account for the rich combinatorial variety of local structures. The problem of how to define expressive yet efficient structural encodings thus remains one of the central open questions in graph representation learning.

Graph transformers can also present a series of scalability issues, especially when combined with equivariance [34]. In these cases, the explicit symmetries imposed by equivariant transformer architectures computationally hinders the generalization of graph transformer-based models to large, complex systems. This is particularly relevant for real-world domains such as molecular crystal modelling, which often contain many more atoms than typical molecular graphs.

All of these questions touch on a deeper issue: the *inductive biases* that underlie learning on graphs. In deep learning, inductive bias refers to the set of assumptions a model makes about the structure of the data, which guides generalization beyond the training distribution [14]. Effective graph inductive biases determine not only how well a model fits to observed data, but also how robustly it extrapolates to unseen structures [13]. Therefore, designing appropriate inductive biases for different applications is both a theoretical and practical challenge: too little structure yields models that fail to capture essential relational dependencies, while too much structure can impose rigidity and limit adaptability.

In this thesis, we will primarily focus on molecular applications of graph representation learning. Molecular data is particularly rich in its structural complexity, from the molecular graphs that underpin computational chemistry to the crystal lattices that define the behavior of organic materials [35, 36]. Atoms and molecules interact via intricate bonds and geometric constraints, while also exhibiting vast combinatorial diversity. Being able to accurately model a given compound's *intramolecular* and *intermolecular* interactions allows us to better understand its function and material properties [37–39]. As such, the ability to effectively capture patterns and *local strurcture* in graph-based molecular data has become a central pursuit in both deep learning and computational chemistry [40, 41].

We consider two complementary notions of local structure in this thesis. The first is the local structure within a single molecule (intramolecular), where nodes denote atoms and edges denote bonds. In this setting, local structure is characterised by the presence of specific subgraph patterns, such as rings, functional groups, or other chemically meaningful motifs. These motifs play a fundamental role in determining molecular properties, including solubility and toxicity [41, 42]. Consequently, graph neural networks (GNNs) have been widely applied to learn graph-level representations of molecular structures for downstream property prediction tasks. Unfortunately, existing GNN models that incorporate these local motifs often remain limited in both expressivity and scalability [26, 27].

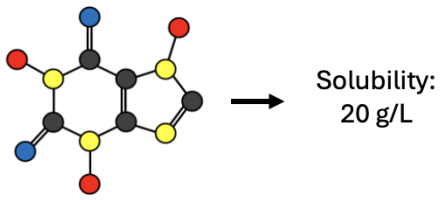


Figure 1.1: Example molecular graph and property prediction task for a Caffeine molecule, which contains local chemical motifs such as a 6-cycle, 5-cycle, etc.

The second notion of local structure that we consider refers to the structure between multiple molecules (intermolecular) in a periodic graph, where nodes denote molecules and edges denote interactions between molecules. Here, local structure refers to the spatial arrangement of neighboring molecules in space, from which the global symmetries of a molecular crystal lattice may emerge. This crystal packing directly governs the physical properties of all materials [43], and significant effort has been devoted to developing models that predict crystal structures directly from molecular information, a problem known as *crystal structure prediction* (CSP) [37–39]. Within this context, GNNs are often integrated into *generative* frameworks, where they serve as powerful encoders and decoders that learn to capture the complex underlying distribution of diverse graph-structured data. Popular generative models that rely on GNN backbones include variational autoencoders (VAEs) [44, 45], diffusion based models [46, 47], and flow matching models [48, 49].

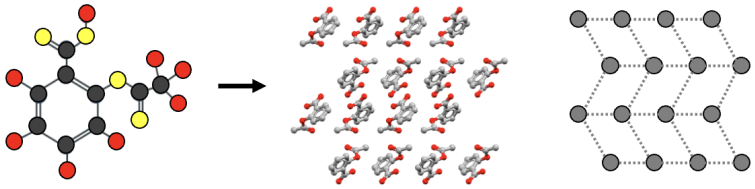


Figure 1.2: Example crystal packing for an Asprin molecule and a rough schematic of its corresponding intermolecular graph, which forms a periodic lattice.

For molecular systems, equivariant architectures based on the SE(3) symmetry group are commonly employed, as they explicitly account for both *translational* and *rotational* symmetries [50]. This is particularly important because certain physical properties may change depending on a molecule's chirality [51]. Equivariant models have proven especially popular for modelling crystalline materials, where periodicity introduces additional geometric symmetries [52]. However, enforcing explicit equivariance often incurs substantial computational cost, making it difficult for such models to scale to larger molecular crystals that are prevalent in industrial settings. Consequently, there is a need for scalable solutions that can capture local interactions between groups of molecules while maintaining computational tractability within a generative framework [39].

Balancing expressivity, inductive bias, and scalability in molecular and materials modelling forms the central theme of this thesis. We bridge the gap between theoretical understanding and practical application by investigating how graph neural networks and graph transformers encode local structural information, and

we introduce new approaches that are (i) provably more expressive and (ii) computationally efficient. Practically, these contributions are explored at both the intramolecular scale, where we enhance the representation of local chemical motifs, and the intermolecular scale, where we propose a novel framework for modelling periodic symmetries in large molecular crystals through learned local interactions. By re-examining current paradigms for representing local structure in complex graph-structured systems, this thesis advances both the theoretical understanding of graph representation learning and its utility for real-world molecular problems.

The remainder of the thesis is organized as follows:

- In Chapter 2, we provide the necessary background and preliminary information that will be used throughout the thesis. More specifically, we formally describe graphs, graph neural networks, graph transformers, and various molecular applications of graph representation learning, which are the primary focus of our empirical evaluations.

- In Chapter 3, we introduce a novel method of studying the expressivity of GNNs via the use of graph motif parameters. In particular, we show that using the *homomorphism basis* of a graph motif parameter provides a more principled way of capturing local structure and increasing the expressivity of GNNs. We provide theoretical results as well as empirical results that highlight the practicality of our approach for several different tasks, including molecular property prediction.

- In Chapter 4, we present Motif Structural Encoding (MoSE), a novel structural encoding for introducing inductive biases into graph transformers and other models with weak inductive biases. We prove that MoSE fully generalizes existing encoding methods, such as random walk structural encodings (RWSE), and comes with a set of expressivity guarantees. Empirically, we show that MoSE consistently improves performance across both molecular benchmarks and other domains.

- In Chapter 5, we propose OXtal, a novel lattice-free diffusion model for molecular crystal structure prediction. Our new Stoichiometric Stochastic Shell Sampling (S^4) approach learns local interactions that capture periodic symmetries without explicitly enforcing model equivariance or defining a unit cell. This enables more scalable training and generalizability for molecular crystal modelling.

- In Chapter 6, we summarize the main contributions of this thesis and propose potential future directions for the work presented.

2
Background

2.1 Graphs, Homomorphisms, and Invariants

An undirected graph G is a tuple (V, E) where V is a set of *vertices* and $E \subseteq V \times V$ is a symmetric *edge* relation. In some contexts, we write $V(G)$ and $E(G)$ for the vertices and edges, respectively, of graph G. Let ρ be a partition of $V(G)$. The quotient graph G/ρ is the graph where the vertices in each block are contracted into a single vertex, i.e., the vertices of G/ρ are the blocks $B \in \rho$, and there is an edge (B_1, B_2) iff there is an edge (v, u) in G for any $v \in B_1, u \in B_2$.

We say that G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The *induced subgraph* of G by $U \subseteq V(G)$ is the graph $G[U]$ with vertices U and exactly those edges from G where both ends of the edge are in U.

A *homomorphism* from graph G to graph H is a function $h : V(G) \to V(H)$ such that for every $(v, w) \in E(G)$, $(h(v), h(w)) \in E(H)$. An *isomorphism* between graphs G and H, denoted $G \simeq H$, is a bijection $f : V(G) \to V(H)$ such that $(v, w) \in E(G)$ if and only if $(f(v), f(w)) \in E(H)$, for each $v, w \in V(G)$. An *automorphism* is an isomorphism of a graph onto itself.

We write $\mathsf{Hom}(G, H)$ for the number of homomorphisms from G to H, $\mathsf{Sub}(G, H)$ for the number of subgraphs H' of H such that $H' \simeq G$, and $\mathsf{IndSub}(G, H)$ for the number of sets $U \subseteq V(H)$ such that $H[U] \simeq G$, i.e., the number of times G is an (induced) subgraph of H. By $\mathsf{Hom}(G, \cdot)$, we denote the function that maps graph H to $\mathsf{Hom}(G, H)$, and analogously for Sub. For a set of graphs $\mathbb{G}$, we write $\mathsf{Hom}(\mathbb{G})$ to represent the set of functions $\{\mathsf{Hom}(G, \cdot) \mid G \in \mathbb{G}\}$.

A *graph invariant* ξ is a function on graphs such that for all graphs G, H, and all isomorphisms f between G and H, we have $\xi(G) = \xi(H)$. A graph invariant

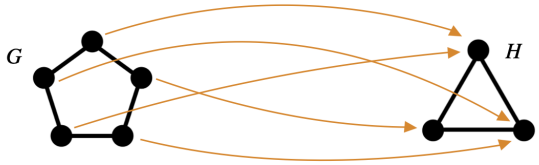


Figure 2.1: Example homomorphism from graph G to graph H, where all edges in G are preserved in H. Note that this mapping is not bijective, as not all nodes in G are preserved in H.

distinguishes two graphs G and H, denoted $G \not\equiv_\xi H$, if $\xi(G) \neq \xi(H)$, and conversely, we write $G \equiv_\xi H$ if $\xi(G) = \xi(H)$.

2.1.1 Graph Motif Parameters

A *graph parameter* is a function that maps graphs into $\mathbb{Q}$. A *graph motif parameter* [53] is a graph parameter that can be expressed as a finite linear combination of homomorphism counts into G. Formally, Γ is a graph motif parameter, if there are graphs $F_1, \ldots, F_\ell$ and a function α mapping them to $\mathbb{Q} \setminus \{0\}$, such that for every graph G:

$$\Gamma(G) = \sum_{i=1}^{\ell} \alpha(F_i) \cdot \mathsf{Hom}(F_i, G), \tag{2.1}$$

where $\alpha(F_i)$ are rational constants depending only on Γ. We refer to the set $\{F_1, \ldots, F_\ell\}$ as the *support* $\mathsf{Supp}(\Gamma)$ of Γ. With Ω representing the class of all graphs, each graph F induces an (infinite) vector $\boldsymbol{l}_F = (\mathsf{Hom}(F, G))_{G \in \Omega}$. The vectors $\boldsymbol{l}_{F_1}, \ldots, \boldsymbol{l}_{F_\ell}$ form a basis of the space of graph motif parameters $(\Gamma(G))_{G \in \Omega}$ [54]. We say that a motif parameter Γ is *connected* if every graph in $\mathsf{Supp}(\Gamma)$ is connected. Informally we also refer to the graphs in the support as the *homomorphism basis* of Γ.

For any graph F, $\mathsf{Sub}(F, \cdot)$ is a graph motif parameter. Its support, commonly called $\mathsf{Spasm}(F)$, is the set of all quotient graphs of F (up to isomorphism) [55]. When the host graphs are assumed to be loop-free, the Spasm is implicitly understood to only contain the loop-free quotients. If F is connected, $\mathsf{Sub}(F, \cdot)$ is a connected graph motif parameter. The $\alpha(F)$ coefficients roughly take the form:

$$\alpha(F) \approx \frac{f(\text{lattice})}{\mathsf{Aut}(F)}, \tag{2.2}$$

where $f(\text{lattice})$ is a function of the lattice of the given basis pattern F, and $\mathsf{Aut}(F)$ is the number of automorphisms of F onto itself. Note that the coefficients

have alternating signs, and this is reflected in the $f(\text{lattice})$ values. More precisely, Equation (2.1) can be expanded as follows to obtain the full equation for computing $\mathsf{Sub}(F, \cdot)$:

$$\mathsf{Sub}(F, \cdot) = \sum_{\rho} \frac{(-1)^{|V(F)|-|V(F/\rho)|} \cdot \prod_{B \in \rho}(|B|-1)!}{\mathsf{Aut}(F)} \cdot \mathsf{Hom}(F/\rho, \cdot). \tag{2.3}$$

As an example, consider the 5-cycle  that has two non-trivial loop-free quotients:  and  (cf. Example 2.1.1).

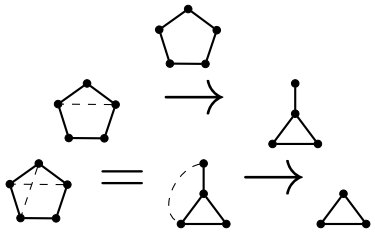
Example 2.1.1. We can express $\mathsf{Sub}($$, G)$ as the finite linear sum:

$$\mathsf{Sub}(\text{  }, G) = \frac{1}{10}\mathsf{Hom}(\text{}, G) - \frac{1}{2}\mathsf{Hom}(\text{}, G) + \frac{1}{2}\mathsf{Hom}(\text{}, G),$$

where the terms for  ,  , and define the *homomorphism basis* of the target function $\mathsf{Sub}($$, \cdot)$. △

We see that the graphs in the basis of the target pattern can easily be obtained by contracting non-adjacent vertices:

Example 2.1.2. Consider the example from Example 2.1.1. The graphs in the basis of the target pattern are those that are obtained by contracting non-adjacent vertices:



which results in a richer set of patterns in the basis. △

Nguyen and Maehara [56] present a generalization of homomorphism counts that allow for vertex weighting. For graph G, let $\omega : V(G) \to \mathbb{R}_{\geq 0}$ describe *node weights*. The *weighted homomorphism count* from a graph F into G is given as:

$$\omega\text{-}\mathsf{Hom}(F, G) = \sum_{f \in \mathcal{H}} \left(\prod_{v \in V(F)} \omega(f(v)) \right),$$

where $\mathcal{H}$ is the set of all homomorphisms from F to G. We define the node-rooted version $\omega\text{-}\mathsf{Hom}_{u \to v}(F, G)$ by restricting $\mathcal{H}$ to only those homomorphisms which map $u \in V(F)$ to $v \in V(G)$. The mapping $\omega\text{-}\mathsf{Hom}_{\to(\cdot)}(F, \cdot)$ is defined analogously to the unweighted case: the graph-node pair (G, v) gets mapped to $\omega\text{-}\mathsf{Hom}_{u \to v}(F, G)$ where $u \in V(F)$ is fixed arbitrarily. Setting $\omega(v) = 1$ for all $v \in V(G)$ recovers

the unweighted count $\omega\text{-}\mathsf{Hom}(F, G) = \mathsf{Hom}(F, G)$, and similarly for the node-rooted version. Weighted homomorphism counts are well-studied in the context of their connection to graph isomorphism [57, 58], and in the context of the universal approximation capabilities and empirical performance of graph classifiers which use $\omega\text{-}\mathsf{Hom}(F, \cdot)$ counts as a graph embedding [56].

2.2 Graph Neural Networks

Graph neural networks (GNNs) have emerged as one of the most powerful tools within graph machine learning due to their ability to both incorporate feature information and explicitly consider the structure of the graph simultaneously. Embeddings for a given node are updated iteratively by aggregating information from neighboring nodes. Due to their construction, GNNs can capture higher-order information and non-linear patterns, such as graph-level properties and structural details. GNNs are also inherently inductive, meaning that they are able to generalize to new nodes or entirely new graphs after training.

Despite these advantages, GNNs also have several limitations. Many GNN architectures are limited in their expressivity, meaning that they are unable to distinguish between certain non-isomorphic graphs. Additionally, GNNs may suffer from over-smoothing, where node representations become indistinguishable after multiple layers of message passing. This can lead to a loss of important structural information and reduced performance on tasks that require fine-grained distinctions between nodes or graphs. GNNs may also struggle with capturing long-range dependencies in graphs, as information is typically propagated through local neighborhoods. Related to this is the problem of over-squashing, where information from distant nodes is compressed into a fixed-size representation, leading to loss of important details.

In the following section, we take a deeper look at message passing GNN architectures and explore some existing literature related to expressivity.

2.2.1 Message Passing Neural Networks

Let us consider featured graphs $G = (V, E, \zeta)$ where $\zeta : V(G) \to \mathbb{R}$, and so $\zeta(u)$ denotes the feature of a node $u \in V$. All notions introduced earlier extend to featured graphs in the usual way. We focus on *Message-Passing Neural Networks (MPNNs)* [22] that encapsulate the vast majority of GNNs. An MPNN updates

the initial node representations $\boldsymbol{h}_v^{(0)} = \zeta(v)$ of each node v for $0 \leq \ell < L$ iterations based on its own state and the state of its neighbors $\mathcal{N}_v$ as:

$$\boldsymbol{h}_v^{\ell+1} = \textsc{Upd}^{(\ell)}\left(\boldsymbol{h}_v^{\ell}, \textsc{Agg}^{(\ell)}\left(\boldsymbol{h}_v^{\ell}, \{\!\{\boldsymbol{h}_u^{\ell} \mid u \in \mathcal{N}_v\}\!\}\right)\right),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset and $\textsc{Upd}^{(\ell)}$ and $\textsc{Agg}^{(\ell)}$ are differentiable *update* and *aggregation* functions, respectively. We denote by $d^{(\ell)}$ the dimension of the node embeddings at iteration (layer) ℓ. The final representations $\boldsymbol{h}_v^{(L)}$ of each node v can be used for predicting node-level properties, or they can be pooled to form a graph embedding vector $\boldsymbol{z}_G$, which can be used for predicting graph-level properties.

Prominent Models. Within the framework of MPNNs, there are a few fundamental networks that have been widely adopted due to their effectiveness and simplicity. Below, we briefly describe three of the most prominent models: Graph Convolutional Network (GCN) [19], Graph Attention Network (GAT) [21], and Graph Isomorphism Network (GIN) [23].

Graph Convolutional Network (GCN). GCN [19] extends the notion of convolution to the graph domain. More specifically, Kipf and Welling [19] propose a first-order approximation of convolutions on graphs that yields a simple message-passing rule:

$$H^{(\ell+1)} = \sigma\Big(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(\ell)}W^{(\ell)}\Big), \tag{2.4}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $W^{(\ell)}$ are learnable weights, and $\sigma(\cdot)$ is a nonlinear activation, such as rectified linear unit (ReLU), sigmoid, hyperloic tangent (tanh), etc. This essentially performs a weighted average of neighboring node features at each iteration. In the MPNN formulation that we defined above, GCN uses mean aggregation and the update function is effectively the combination of applying a weight matrix and the specified nonlinearity:

$$\boldsymbol{h}_v^{(\ell+1)} = \sigma\left(W^{(\ell+1)} \sum_{u \in \mathcal{N}_v \cup \{v\}} \frac{\boldsymbol{h}_u^{(\ell)}}{\sqrt{\tilde{d}_v + \tilde{d}_u}}\right). \tag{2.5}$$

Note that the self loop means that GCN jointly considers the node's own features along with its neighbors' features during the aggregation step. Furthermore, although GCN is able to capture local graph structure via message passing by explicitly considering adjacency and neighborhood sizes, the use of mean aggregation means that two structurally distinct neighborhoods with the same graph may map to the same representation. This is an important design choice that we address more later on in Section 2.2.2.

Graph Attention Network (GAT). GAT [21] integrates attention mechanisms into the message-passing framework, allowing nodes to learn the relative importance of their neighbors. Given node embeddings $\{\boldsymbol{h}_v^{(\ell)}\}_{v\in V}$, the unnormalized attention coefficients between nodes v and u are computed as:

$$e_{vu} = \mathrm{LeakyReLU}\Big(\boldsymbol{a}^\top \Big[W\boldsymbol{h}_v^{(\ell)} \,\|\, W\boldsymbol{h}_u^{(\ell)}\Big]\Big), \tag{2.6}$$

where $W \in \mathbb{R}^{d'\times d^{(\ell)}}$ is a learnable linear transformation, $\boldsymbol{a} \in \mathbb{R}^{2d'}$ is a learnable attention vector, and $\|$ denotes concatenation. These scores are normalized across the neighborhood of v using the softmax function:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k\in\mathcal{N}_v}\exp(e_{vk})}. \tag{2.7}$$

The node representation is then updated via a weighted aggregation:

$$\boldsymbol{h}_v^{(\ell+1)} = \sigma\left(\sum_{u\in\mathcal{N}_v} \alpha_{vu} W\boldsymbol{h}_u^{(\ell)}\right). \tag{2.8}$$

To stabilize training and increase model performance, multi-head attention is employed:

$$\boldsymbol{h}_v^{(\ell+1)} = \|_{k=1}^{K}\sigma\left(\sum_{u\in\mathcal{N}_v} \alpha_{vu}^{(k)} W^{(k)}\boldsymbol{h}_u^{(\ell)}\right), \tag{2.9}$$

where $\|$ denotes concatenation across K attention heads. We will address this attention mechanism further in Section 2.3.

Graph Isomorphism Network (GIN). GIN [23] is a powerful GNN architecture that was designed to use an injective aggregation function. Its message passing function is defined as:

$$\boldsymbol{h}_v^{(\ell+1)} = \mathrm{MLP}^{(\ell)}\left((1+\epsilon^{(\ell)})\,\boldsymbol{h}_v^{(\ell)} + \sum_{u\in\mathcal{N}_v}\boldsymbol{h}_u^{(\ell)}\right), \tag{2.10}$$

where $\epsilon^{(\ell)}$ is either a learnable scalar or a fixed constant. Unlike mean or max aggregations, this *sum* aggregation is injective under mild assumptions, allowing the network to distinguish between different neighborhood structures. The subsequent multilayer perceptron (MLP) provides a learnable transformation that enhances representational capacity. Xu et al. [23] proposed GIN to address the limited expressivity of earlier GNNs, which were shown to be no more powerful than the 1-Weisfeiler–Lehman (1-WL) graph isomorphism test under certain conditions. We will now discuss the Weisfeiler-Leman hierarchy and its connection to GNN expressivity in more detail.

2.2.2 Weisfeiler-Leman Hierarchy

Expressive Power. A *graph invariant* is a function $\xi(\cdot)$ which acts on graphs, satisfying $\xi(G) = \xi(H)$ whenever $G \cong H$ for all graphs G and H. For indexed families of graph invariants $\{A_i\}_{i\in I}$ and $\{B_j\}_{j\in J}$, we define the *expressivity* relation $A \preceq B$ to denote: for any choice of $i \in I$, there exists a choice of $j \in J$ such that $B_j(G) = B_j(H)$ implies that $A_i(G) = A_i(H)$ for all G and H. If $A \preceq B$ and $B \preceq A$, then we say that $A \simeq B$. We write $A \prec B$ when B has strictly greater expressive power, $A \preceq B$ but $A \not\simeq B$. When we reference $\preceq$ for some fixed graph invariant $\xi(\cdot)$, we interpret ξ as a family which contains only one graph invariant (hence the indexing is trivial). Noting that a GNN architecture can be treated as a family of graph invariants indexed by the model weights, the relation $\preceq$ generalizes common notions of the graph-distinguishability expressive power for GNNs [59]. As we will see below, the $\preceq$ relation naturally extends to expressivity comparisons between structural or positional encoding schemes as well.

MPNNs and Weisfeiler-Lehman Tests. Recall that MPNNs are defined by Gilmer et al. [22] through a node representation update scheme in which a node's neighbors are iteratively aggregated for each update. Xu et al. [23] and Morris et al. [25] note that an MPNN is most expressive when its update and aggregation schemes are injective; such an MPNN can be abstractly represented as a color refinement scheme known as the *1-dimensional Weisfeiler-Lehman test* (1-WL). Injective functions include the sum aggregation used by GIN [23], but exclude mean or max aggregations used by GCN [19] and GraphSAGE [20], respectively.

Given graph G, the 1-WL test induces the node coloring $c^{(t)} : V(G) \to \Sigma$ for all t up until some termination step T, at which point the graph-level 1-WL label is defined as the multiset of final node colors $\text{1-WL}(G) = \{\!\{c^{(T)}(v) : v \in V(G)\}\!\}$. Graphs G and H are considered *1-WL indistinguishable* if they have identical graph labels $\text{1-WL}(G) = \text{1-WL}(H)$. Crucially, it holds that MPNN $\preceq$ 1-WL where we treat "MPNN" as a family of graph invariants indexed by both the choice of message-passing architecture and the model parameters, and we treat 1-WL as a singleton family containing only the graph invariant $G \mapsto \text{1-WL}(G)$ [23].

Weisfeiler-Leman Hierarchy. We can extend the Weisfeiler-Lehman test to "higher dimensions" by coloring k-tuples of nodes. The k-WL test induces node tuple coloring $c^{(t)} : V(G)^k \to \Sigma$ for $t = 1, ..., T$ iterations, and then it aggregates a final graph-level k-WL label analogously to 1-WL [60]. We will refer to the 1-WL algorithm [61], as well as to the usual k-WL hierarchy, which are iterative algorithms

that yield graph invariants. In this thesis, we will always refer to the *folklore* version of these algorithms (k-WL), rather than the oblivious version (oblivious k-WL), where the only difference is in the dimension counts: k-WL is equivalent to oblivious $(k+1)$-WL, for every $k > 1$ [62, 63]. It has been shown that the k-WL tests form a well-ordered hierarchy of expressive power, where $k\text{-WL} \prec (k+1)\text{-WL}$ [64]. Here, we treat k-WL as a singleton family (trivial indexing) for each particular choice of k.

2.2.3 Higher-Order Graph Neural Networks

Recent research has devoted substantial attention to developing *higher-order graph neural networks* to overcome the limited expressivity of standard MPNNs. As discussed above, classical MPNNs are at most as powerful as the 1-dimensional Weisfeiler–Lehman test (1-WL) in distinguishing non-isomorphic graphs. This limitation arises from the fact that MPNNs operate solely on node-level representations and exchange information through pairwise edges, which restricts their ability to capture richer substructural dependencies such as motifs, cycles, and higher-order interactions. Higher-order GNNs thereby seek to increase the expressive power of GNNs by incorporating more complex relational structures in different ways. In this section, we will explore some popular higher-order GNN models that are particularly relevant to this thesis.

Prominent Models. Several higher-order GNN architectures have been proposed to balance expressive power and computational feasibility, which we will cover in more breadth in Section 3.2. Here, we provide a detailed description of methods that are particularly relevant for this thesis, which include *k-GNNs* [25], and pattern-based methods such as *Graph Substructure Network (GSN)* [26] and Local Graph Parameter Graph Neural Networks (LGP-GNNs) [27]. Although they share a common motivation—to exceed the 1-WL limit, they differ in how they represent higher-order interactions and in the trade-offs they make between expressivity and efficiency.

k-GNN. k-GNN [25] explicitly extends message passing to k-tuples of nodes, directly emulating the behavior of the *oblivious* k-WL algorithm. Formally, consider a graph $G = (V, E)$ and fix an integer $k \geq 2$. The model assigns a feature vector $h_S^{(\ell)}$ to each subset of nodes $S \subseteq V$ with $|S| = k$, representing the embedding of that node set at layer ℓ. Each tuple S interacts with neighboring tuples T that differ from S in exactly one node:

$$N(S) = \{\, T \subseteq V : |T| = k,\ |S \cap T| = k - 1 \,\}.$$

These neighboring tuples define the higher-order adjacency structure on which message passing occurs. The layer-wise update rule for k-GNN is given by

$$\boldsymbol{h}_S^{(\ell+1)} = \sigma\left(W_1^{(\ell)}\boldsymbol{h}_S^{(\ell)} + \sum_{T\in N(S)} W_2^{(\ell)}\boldsymbol{h}_T^{(\ell)}\right), \tag{2.11}$$

where $W_1^{(\ell)}$ and $W_2^{(\ell)}$ are learnable weight matrices and $\sigma(\cdot)$ is a nonlinear activation function.

In theory, a k-GNN possesses the expressive power of $(k-1)$-WL, with strictly increasing expressive power for increasing k, paralleling the k-WL hierarchy. However, this expressivity comes at a steep computational cost: maintaining and updating embeddings for all k-tuples requires $\mathcal{O}(|V|^k)$ complexity. Consequently, these models are memory-intensive and often intractable for large graphs or high values of k. As a result, their practical utility is limited.

Graph Substructure Network (GSN). GSN [26] extends the standard message-passing framework by augmenting node features with explicit information about local graph substructures, such as cycles and cliques. This design allows the model to capture complex structural patterns that are beyond the reach of standard MPNNs without explicitly simulating higher-order message passing

In GSN, each node $v \in V$ is assigned an additional substructure feature vector s_v that encodes the presence or count of specific motifs or topological patterns involving that node (also referred to as orbitals). These substructure embeddings are computed once at preprocessing time and concatenated to the initial node features. This effectively conditions the node's representation on its local structural context.

From a theoretical perspective, Bouritsas et al. [26] demonstrate that augmenting nodes with appropriate substructure identifiers can enhance the expressive power of a GNN beyond the 1-WL limit in some cases, which are dependent on the choice of substructures used. This presents a practical path toward higher-order reasoning within the message-passing paradigm. However, the process of selecting specific substructures to encode is very adhoc, and its expressive benefits are not fully characterized.

Local Graph Parameter Graph Neural Network (LGP-GNN). Similarly to GSN, LGP-GNN [27] enhances node representations by incorporating local structural information. However, instead of using subgraph isomorphism counts, LGP-GNN uses homomorphism counts of the respective predefined patterns, which are referred to as local graph parameters.

Barceló et al. [27] show that LGP-GNNs can surpass the expressive power of 1-WL under certain conditions, depending on the choice of local graph parameters used. However, similar to GSN, the selection of appropriate local parameters is somewhat arbitrary, and the full extent of its expressive potential remains an open question, which we will explore further in Chapter 3.

Now that we have explored various higher-order GNN architectures and their expressivity, we will turn our attention to Transformers, which are another powerful class of models that have revolutionized machine learning in recent years. We see that graph transformers are very closely related to their GNN counterparts, and we will discuss how these models leverage different encoding mechanisms to capture graph structure.

2.3 Graph Transformers

2.3.1 Transformer Architecture

The Transformer architecture [6] has become a cornerstone of modern machine learning, underpinning state-of-the-art models in natural language processing, computer vision, and beyond. Transformers introduced a paradigm shift by replacing recurrence and convolution with *self-attention* mechanisms, enabling the model to learn which relationships to "attend" to within a sequence.

Self-Attention. Given a sequence of input representations $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{n \times d}$, each token is projected into three distinct subspaces through learnable matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$, producing queries, keys, and values:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V. \tag{2.12}$$

Attention weights are computed as a scaled dot-product between queries and keys, followed by a softmax normalization:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.13}$$

This operation enables each token to attend selectively to all others, allowing information to propagate globally across the sequence. To enhance representational capacity, Transformers employ multi-head attention, in which h parallel attention heads compute independent representations:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O, \tag{2.14}$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_Q^{(i)}, \mathbf{K}\mathbf{W}_K^{(i)}, \mathbf{V}\mathbf{W}_V^{(i)}). \tag{2.15}$$

Positional Encodings. Unlike recurrent networks, the Transformer lacks an inherent notion of order. To incorporate positional information, a deterministic or learned *positional encoding* $\mathbf{P} \in \mathbb{R}^{n \times d}$ is added to the input embeddings:

$$\mathbf{Z}^{(0)} = \mathbf{X} + \mathbf{P}. \tag{2.16}$$

In the original Transformer model [6], the positional encoding is *deterministic* and based on sine and cosine functions of different frequencies. For each position *pos* and specific dimension i, the encoding is defined as:

$$\text{P}_{(pos,\,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \tag{2.17}$$

$$\text{P}_{(pos,\,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right). \tag{2.18}$$

Here, *pos* is the token position (starting from 0), i indexes the embedding dimension, and d is the model dimensionality (i.e. of the output sample). Each dimension of the positional encoding corresponds to a sinusoid with a different wavelength, allowing the model to represent both absolute and relative positions through simple linear operations on these encodings.

Encoder. The encoder is composed of L stacked layers, each consisting of multi-head self-attention and a feed-forward network, combined via residual connections and layer normalization:

$$\mathbf{Z}'^{(l)} = \text{LayerNorm}\Big(\mathbf{Z}^{(l-1)} + \text{MultiHead}\Big(\mathbf{Z}^{(l-1)}\Big)\Big), \tag{2.19}$$

$$\mathbf{Z}^{(l)} = \text{LayerNorm}\Big(\mathbf{Z}'^{(l)} + \text{FFN}\Big(\mathbf{Z}'^{(l)}\Big)\Big), \tag{2.20}$$

where the position-wise feed-forward network (FFN) is defined as:

$$\text{FFN}(\mathbf{z}) = \max(0, \mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{2.21}$$

The decoder mirrors this structure, incorporating both self-attention and encoder–decoder attention to condition on encoded representations.

2.3.2 Graph-Aware Self Attention

Graph transformers extend the self-attention framework to operate on graph-structured data [28]. As mentioned previously, traditional GNNs struggle with capturing long-range dependencies due to their local message passing aggregation schemes. Incorporating messages or signals from distant nodes requires many layers, which can lead to over-smoothing and information loss.

To overcome this limitation, graph transformers adopt the global attention mechanism of the Transformer [6], enabling every node to "attend" to every other node. In other words, graph transformers can be viewed as GNNs that operate over fully-connected graphs, where attention weights determine the strength of interactions between nodes [31]. This allows for direct modeling of long-range dependencies without the need for deep architectures.

When applied to graphs, a single "head" of self-attention learns a ℓ^{th}-layer hidden representation $\boldsymbol{h}_v^{(\ell)} \in \mathbb{R}^d$ of every node $v \in V(G)$ by taking the weighted sum:

$$\boldsymbol{h}_v^{(\ell)} = \phi^{(\ell)} \left(\sum_{u \in V(G)} \alpha_{v,u}^{(\ell)} \boldsymbol{h}_u^{(\ell-1)} \right), \tag{2.22}$$

where the *attention coefficients* are given by:

$$\alpha_{v,u}^{(\ell)} = \psi^{(\ell)} \left(\boldsymbol{h}_v^{(\ell-1)}, \boldsymbol{h}_u^{(\ell-1)} \right). \tag{2.23}$$

Here, $\phi^{(\ell)}$ and $\psi^{(\ell)}$ are learnably-parameterized functions. The most common implementation of $\phi^{(\ell)}$ and $\psi^{(\ell)}$ is scaled dot product attention, as defined by Vaswani et al. [6]. Most graph transformers also utilize multiple attention heads in parallel (whose outputs are concatenated at each layer), and interweave self-attention layers with fully-connected feed-forward layers.

2.3.3 Graph Positional and Structural Encodings

Since basic self-attention does not receive the graph adjacency matrix as an input, many graph transformer models choose to inject a graph's structural information into the model inputs by way of *node positional or structural encodings*. More broadly, we can define a node positional or structural encoding to mean any isomorphism-invariant mapping *pe* which takes in a node-graph pair (v, G) and outputs a vector label $pe(v, G) \in \mathbb{R}^d$. Then, the graph-level PE label is defined as the multiset of node-level labels $\text{PE}(G) = \{\!\{pe(v, G) : v \in V(G)\}\!\}$. Various different encoding methods have been developed for graph transformers, which we highlight in more detail in Section 4.2.

In practice, each node's embedding vector is then concatenated or added to the node's initial feature vector, and the resulting node embedding is passed into the model as an input. Note that sometimes, the initial node feature and positional or structural encoding vector are passed through MLPs before being combined and fed into the rest of the model.

Laplacian Positional Encoding (LapPE). Dwivedi et al. [33] introduced LapPE, a class of positional encodings based on graph Laplacian eigenvectors [65]. Formally, these vectors are defined by factorizing the graph Laplacian matrix:

$$\Delta = I - D^{-1/2} A D^{-1/2} = U^{\top} \Lambda U, \tag{2.24}$$

where A is the adjacency matrix, D is the degree matrix, Λ is a diagonal matrix of eigenvalues, and U is a matrix whose columns are the corresponding eigenvectors. LapPE is constructed by selecting the k smallest non-trivial eigenvectors:

$$p_i = [u_1(i), u_2(i) \cdots, u_k(i)] \in \mathbb{R}^k. \tag{2.25}$$

These eigenvectors form a smooth coordinate system over the graph, preserving both local neighborhood structure and global connectivity. As low-frequency Laplacian eigenvectors vary slowly over edges, they provide stable and smooth positional signals for nearby nodes. LapPE can be interpreted as a natural generalization of sinusoidal positional encodings used in Transformers [6]. However, it is important to note that the eigenvectors are only defined up to a sign, which can lead to ambiguities in the positional encodings. To address this, Dwivedi et al. [33] use random sign flipping to construct their LapPE vectors.

Random Walk Structural Encoding (RWSE). Dwivedi et al. [32] introduced RWSE, a class of structural encodings based on random walks, which Rampásek et al. [29] later demonstrated to be particularly effective for molecular graphs. RWSE_k is defined by k-steps of random walks on the graph:

$$p_i = \left[\text{RW}_{ii}, \text{RW}_{ii}^2, \cdots, \text{RW}_{ii}^k\right] \in \mathbb{R}^k, \tag{2.26}$$

where $\text{RW} = AD^{-1}$ is the random walk operator, A is the adjacency matrix, and D is the degree matrix. Note that RWSE uses a low-complexity version of the random walk matrix by considering only the diagonal entries, which correspond to the probability of returning to the starting node after i steps (i.e. RW_{ii}). Also, RWSE does not suffer from the same sign ambiguity issues as LapPE as mentioned above.

Using our general definition of node positional and structural encodings above, we can treat RWSE_k as a family of graph invariants (mapping into multisets with elements in $\mathbb{R}^d$) indexed by k. Hence, we can compare the graph-level expressivity of RWSE_k and any other general positional or structural encoding scheme under $\preceq$.

Relative Random-Walk Positional Encoding (RRWP). Ma et al. [30] introduce RRWP as part of their GRIT graph transformer architecture (which we will discuss in more detail below in Section 2.3.4). For a fixed horizon k, the RRWP positional encoding between nodes i and j is defined as:

$$p_{i,j} = \left[I,\ M,\ M^2,\ \ldots,\ M^{k-1}\right]_{i,j} \in \mathbb{R}^k, \tag{2.27}$$

where I is the identity matrix and M is the random-walk transition matrix given by $M = D^{-1}A$, with D being the degree matrix and A the adjacency matrix. Ma et al. [30] show that these encodings capture multi-hop connectivity information and subsume shortest-path distances as a special case, while providing strictly greater expressive power for distinguishing graph structures.

2.3.4 Prominent Models

Although Dwivedi and Bresson [28] were the first to formally generalize transformer neural networks to graphs, numerous different architectures have since been proposed under the umbrella of graph transformers. Some of these models which are particularly relevant in this thesis include GPS [29], GRIT [30], and the explicitly equivariant Equiformer [34, 66] model. A more complete survey of graph transformer architectures is provided in Section 4.2.

General, Powerful, Scalable Graph Transformer (GPS). GPS [29] combines the strengths of local message passing and global attention within a unified framework. Each layer integrates a standard MPNN block with a Transformer-style global attention block:

$$\boldsymbol{h}_v^{(\ell+1)} = \text{MPNN}^{(\ell)}(\boldsymbol{h}_v^{(\ell)}, N_v) + \text{Transformer}^{(\ell)}\Big(\boldsymbol{h}_v^{(\ell)}, \{\boldsymbol{h}_u^{(\ell)} : u \in V\}\Big),$$

where the MPNN component captures fine-grained local structure while the Transformer component aggregates long-range dependencies across the entire graph. Rampásek et al. [29] also evaluate several different positional and structural encodings for graph transformers, including LapPE and RWSE, which are detailed above. Furthermore, they experiment with various different MPNN components,

including GIN and GatedGCN [67], which is a variation of the GCN model that incorporates learnable, edge-wise gates to control information flow.

Rampásek et al. [29] show that GPS is a universal function approximator on graphs, meaning that in principle, it can represent any function on graph-structured data given sufficient capacity. Interestingly, their empirical results also show that this hybrid MPNN+Transformer architecture often outperforms pure Transformer-based models, highlighting the complementary strengths of local and global information aggregation.

Graph Inductive Bias Transformer (GRIT). GRIT [30] is a recent graph transformer architecture that removes traditional message-passing, instead relying on strong graph inductive biases embedded directly into the model. These include learned RRWP encodings, an attention mechanism that updates both node and node-pair representations, and per-layer injection of node-degree information with batch normalization.

Unlike standard graph transformers, GRIT maintains both node representations $\boldsymbol{h}_i$ and node-pair representations $\boldsymbol{e}_{i,j}$, the latter initialized using RRWP. A single GRIT attention layer computes

$$\widehat{\boldsymbol{e}}_{i,j} = \sigma(\rho((W_Q \boldsymbol{h}_i + W_K \boldsymbol{h}_j) \odot W_{E_w} \boldsymbol{e}_{i,j}) + W_{E_b} \boldsymbol{e}_{i,j}), \tag{2.28}$$

$$\alpha_{ij} = \mathrm{softmax}_{j \in V}\Big(W_A \widehat{\boldsymbol{e}}_{i,j}\Big), \tag{2.29}$$

$$\widehat{\boldsymbol{h}}_i = \sum_{j \in V} \alpha_{ij}\Big(W_V \boldsymbol{h}_j + W_{E_v} \widehat{\boldsymbol{e}}_{i,j}\Big), \tag{2.30}$$

where $\odot$ denotes elementwise multiplication, $\rho(\cdot)$ is a signed square-root nonlinearity, $\sigma(\cdot)$ is a pointwise activation function, and all W are learnable parameter matrices. Importantly, the node-pair representations are updated at each layer, allowing the model to refine its structural encodings during learning.

Since self-attention is invariant to node degree, it can result in the loss of important topological information. GRIT addresses this by explicitly injecting degree information into the node representations. Let d_i denote the degree of node i, and let $\boldsymbol{x}_i^{\mathrm{out}}$ be the node output after attention and feed-forward layers. Degree information is incorporated via

$$\boldsymbol{h}_i^{\mathrm{out}\,\prime} = \boldsymbol{h}_i^{\mathrm{out}} \odot \boldsymbol{\theta}_1 + \log(1 + d_i)\, \boldsymbol{h}_i^{\mathrm{out}} \odot \boldsymbol{\theta}_2, \tag{2.31}$$

where $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$ are learnable scaling parameters. To preserve degree information, GRIT employs Batch Normalization rather than Layer Normalization, which would otherwise remove degree-dependent scaling effects.

All together, the relative random-walk positional encodings, learnable node-pair updates, and degree-aware scaling enable GRIT to recover classical graph propagation operators and to distinguish graph structures beyond shortest-path-based methods. This work demonstrates that carefully designed inductive biases allow graph transformer architectures to perform competitively without explicit message passing.

Equiformer. Equiformer [34, 66] extends the Transformer framework to continuous 3D geometric graphs, enforcing exact equivariance under the E(3) symmetry group of rotations, translations, and reflections (as well as the SE(3) group of rotations and translations). The model leverages equivariant features built from irreducible representations (irreps), replacing standard Transformer operations with their equivariant counterparts, and explicitly includes tensor-product interactions between irreps channels.

Note that a mapping f is equivariant to a group G (e.g. SE(3)) when, for any group element $g \in G$, the action on inputs commutes with the action on outputs:

$$f\Big(D_X(g)\, x\Big) = D_Y(g)\, f(x), \tag{2.32}$$

where $D_X(g)$ and $D_Y(g)$ are the representation matrices acting on input and output vector spaces respectively. In other words, applying a transformation to the input corresponds to applying the same transformation to the output.

A single Equiformer layer computes equivariant pair features, attention weights, and messages as follows:

$$\widehat{\boldsymbol{e}}_{ij}^{(\ell)} = \mathrm{MLP}\Big(\, \mathcal{T}\Big(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{r}_{ij}\Big)\, \Big), \tag{2.33}$$

$$\alpha_{ij}^{(\ell)} = \mathrm{softmax}_{j\in\mathcal{N}(i)}\Big(\mathrm{MLP}\Big(\widehat{\boldsymbol{e}}_{ij}^{(\ell)}\Big)\Big), \tag{2.34}$$

$$\boldsymbol{m}_{ij}^{(\ell)} = \Phi\Big(\boldsymbol{h}_j^{(\ell)},\ \widehat{\boldsymbol{e}}_{ij}^{(\ell)}\Big), \tag{2.35}$$

$$\boldsymbol{h}_i^{(\ell+1)} = \sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(\ell)}\, \boldsymbol{m}_{ij}^{(\ell)} \ +\ \mathrm{FFN}^{(\ell)}\Big(\boldsymbol{h}_i^{(\ell)}\Big), \tag{2.36}$$

where $\boldsymbol{r}_{ij}$ denote geometric information (e.g. relative displacement encoded as irreps), $\mathcal{T}(\cdot)$ denotes equivariant tensor-product interactions, and $\Phi(\cdot)$ is an equivariant message function (typically also involving tensor products and nonlinearities). Note that all intermediate operations are implemented to preserve equivariance to 3D rotations and translations.

The tensor products combine irreps channels to produce higher-order geometric features in the form:

$$\mathcal{T}\Big(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)}, \boldsymbol{r}_{ij}\Big) \;=\; \sum_{l_1,l_2} C^{(l_1,l_2)}\Big(\boldsymbol{h}_i^{(\ell)(l_1)} \otimes \boldsymbol{r}_{ij}^{(l_2)}\Big), \tag{2.37}$$

where $\boldsymbol{h}^{(l)}$ denotes channels of degree l (irreps type) and $C^{(l_1,l_2)}$ are learnable coupling coefficients. These tensor products are the primary computational cost driver, especially as the maximum degree $L_{\max}$ increases.

In an effort to make Equiformer more accessible, Liao et al. [34] introduce EquiformerV2, which incorporates several architectural and algorithmic improvements to enhance efficiency and scalability. However, the computational overhead of maintaining equivariance is still on the order of $O(L_{\max}^3)$, which remains a significant bottleneck that we will address in Chapter 5.

2.4 Continuous-Time Diffusion Models

Many GNNs and graph transformers have been successfully integrated into generative modeling frameworks for graphs, with remarkable success in applications such as protein structure prediction and inorganic materials discovery [11, 18]. Here, we will provided a brief overview of the diffusion paradigm, which underpins many state-of-the-art generative models.

A diffusion model solves the generative modelling problem by being the solution to a (Itô) stochastic differential equation (SDE) [68],

$$\mathrm{d}\mathbf{X}_t = f_t(\mathbf{X}_t)\,\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \quad \mathbf{X}_0 \sim p_0. \tag{2.38}$$

In Equation (2.38), we use boldface to denote random variables and normal script to denote functions or samples. The function $f_t : \mathbb{R}^d \to \mathbb{R}^d$ corresponds to the average direction of evolution, while $\sigma_t : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient for the Wiener process $\mathbf{W}_t$. By convention, this is termed the forward noising process that starts from time $t = 0$ and progressively corrupts data until the terminal time $t = 1$ where we reach a structureless prior $p_1(x_1) := \mathcal{N}(0, I)$.

Under mild regularity assumptions, forward SDEs of the form of Equation (2.38) admit a time reversal, which itself is another SDE in the reverse direction $t = 1$ to $t = 0$ and transmutes a prior sample $x_1 \sim p_1$ to a sample from the target distribution $x_0 \sim p_0$ [69],

$$\mathrm{d}\mathbf{X}_t = \Big(f_t(\mathbf{X}_t)\,\mathrm{d}t - \sigma_t^2 \nabla_x \log p_t(\mathbf{X}_t)\Big)\,\mathrm{d}t + \sigma_t \mathrm{d}\overline{\mathbf{W}}_t, \quad \mathbf{X}_1 \sim p_1. \tag{2.39}$$

Here $\overline{\mathbf{W}}_t$ is another standard Wiener process, and the *reverse-time* SDE shares the same time marginal density p_t as the forward SDE. Critically, the forward and reverse SDEs are linked via the Stein score $\nabla_x \log p_t(x_t)$, which is the key quantity of interest in the design of diffusion models. More precisely, diffusion models estimate the score function by forming an ℓ_2-regression objective using a denoising network $D_\theta(x_t, t)$. For example, for the variance exploding (VE) SDE family of forward processes: $p_t = p_0 * \mathcal{N}(0, \sigma_t^2)$, this corresponds to learning the set of optimal denoisers, i.e., the set of conditional expectations, across time $D(x_t, t) = \mathbb{E}\left[\mathbf{X}_0 | \mathbf{X}_t = x_t\right], \forall t \in [0, 1]$.

Owing to the nature of Gaussian convolution, we can convert this to a simple *simulation-free* training *conditional* objective for any Bregman divergence with convex F, $\mathbb{B}_\mathrm{F}$ [Holderrieth et al. [70], Prop. 2]:

$$\mathcal{L}_c(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1), x_0\sim p(x_0), x_t\sim p_t(x_t|x_0)}\left[\lambda(t)\mathbb{B}_\mathrm{F}(x_0, D_\theta(x_t, t))\right], \tag{2.40}$$

where $\lambda(t) : [0, 1] \to \mathbb{R}_+$ is any weighting function. For a sufficiently expressive family of denoisers $\mathcal{H} = \{D_\theta : \theta \in \Theta\}$, the minimizer to Equation (2.40) is $D^*_\theta(x_t, t) = D(x_t, t) = \mathbb{E}\left[\mathbf{X}_0 | \mathbf{X}_t = x_t\right]$. Given a trained denoising model, diffusion models generate samples at inference by simulating the reverse-time dynamics of (2.39) with the learned score s_θ computed as $s_\theta \leq (D_\theta(x_t, t) - x_t)/\sigma_t^2$ in place of the true score at x_t, $\nabla_x \log p_t(x_t)$.

2.4.1 Diffusion Transformers

Recently, Diffusion Transformers (DiTs) [71] have emerged as a powerful class of generative models that combine the denoising diffusion framework with transformer-based architectures. In practice, DiTs use a transformer network to parameterize the denoising network $D_\theta(x_t, t)$ using a sequence of latent tokens.

At a high level, a DiT model consists of repeated Transformer layers of the form

$$\boldsymbol{h}^{(\ell+1)} = \mathrm{FFN}^{(\ell)}\Big(\mathrm{SelfAttn}^{(\ell)}(\boldsymbol{h}^{(\ell)})\Big), \tag{2.41}$$

where self-attention enables global interactions between all latent tokens. This architectural bias is particularly well-suited to diffusion models, where global coherence must be gradually recovered during the reverse-time denoising process.

Although originally proposed for image generation, the core idea of DiTs extends naturally to graph-structured domains. In this setting, the noisy state x_t corresponds to a collection of node-level features defined over a graph. Each node is treated as a token, and the transformer processes the resulting set of node embeddings using self-attention to model global interactions across the structure.

Scalability and Empirical Performance. A central result of Peebles and Xie [71] is that diffusion models exhibit strong *scaling behavior* when paired with Transformer backbones. As model depth, width, and token count increase, DiTs demonstrate predictable improvements in sample quality. This characteristic is especially relevant in the architectural design choices that we make in Chapter 5, and speaks to the intricate interplay of representational capacity, scale, and computational efficeincy that we study in this thesis.

2.5 Molecular Applications of Graph Learning

As mentioned in Chapter 1, the primary application of graph learning that we will address in this thesis is molecular modeling. The entire drug discovery process can take over 10 years and cost over $1 billion from initial target identification to market release [42]. Due to the vast expense and high attrition levels of drug candidates, there has been an increased interest in using computational methods to expedite various parts of the pipeline. Graph representation learning can be applied to nearly every stage of the drug development process, including target identification, molecular property prediction, small molecule de novo design, protein engineering, drug repurposing, molecular crystal structure prediction, and more [16, 17]. All of these downstream applications can more or less directly be mapped to one of the three categories of graph-learning tasks outlined below:

1. **Graph-level Tasks:** Predicting properties or labels for entire graphs. This includes tasks such as molecular property prediction, where each molecule is represented as a graph and the goal is to predict properties like solubility or toxicity. Each model is evaluated using the mean average error (MAE), which is defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{2.42}$$

   where y_i is the predicted value, x_i is the true value, and n is the total number of data points.

2. **Node-level Tasks:** Predicting properties or labels for individual nodes within a graph. Molecular structure generation (sometimes referred to as molecular conformer generation) can be framed as a node-level task, where the goal is to predict the 3D coordinates of each atom (node) in a molecule. For molecular crystal packing, this extends to predicting the atom coordinates of multiple

molecules in a periodic lattice. Generated structures are typically evaluated using metrics such as root-mean-square deviation (RMSD) between predicted and true atomic positions.

Given two molecular structures with N corresponding atoms, let their predicted and true 3D coordinates be:

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}, \quad \mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\},$$

where each $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^3$.

RMSD can then be computed as:

$$\text{RMSD}(\mathbf{X}, \mathbf{Y}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{y}_i\|^2}.$$

Note that RMSD is typically calculated after optimal alignment of the two structures to account for rotational and translational differences.

For *de novo generation*, models are typically evaluated using additional metrics such as validity, uniqueness, and novelty of the generated structures. However, this is beyond the scope of this thesis, since we will focus on molecular crystal structure prediciton (Chapter 5).

3. **Edge-level Tasks:** Predicting relationships or interactions between pairs of nodes. These include link prediction tasks in homogeneous (or heterogeneous) graphs, where the goal is to predict missing edges (relationships) between entities (nodes). In molecular applications, this can involve predicting interactions between different genes or between a drug and its target protein.

   Models for link prediction are often evaluated using the Hits@k metric, which is a rank-based metric. Hits@k measures the fraction of test samples (in this case, samples are potential edges) for which the ground-truth label appears within the model's top-k ranked predictions against a set of negative samples. Formally, let N denote the number of test samples, y_i the ground-truth label for the i-th sample, and $\hat{Y}_i^{(k)}$ the set of top-k predictions produced by the model. Then

$$\text{Hits@}k = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\Big[\, y_i \in \hat{Y}_i^{(k)} \,\Big],$$

   where $\mathbf{1}[\cdot]$ is the indicator function that equals 1 if the condition inside holds and 0 otherwise.

Now that we have outlined the necessary preliminaries, we will proceed to present our contributions in advancing the understanding of local structural awareness in graph learning. In Chapter 3, we explore how to properly encode local structure within a given graph in GNNs using graph motif parameters. In Chapter 4, we propose a general graph structural encoding technique based on homomorphism counts that is particularly useful for models that lack inherent graph inductive biases. Finally, in Chapter 5, we introduce OXTAL, a scalable all-atom diffusion transformer that leverages new ways of capturing local interactions between molecular graphs to model periodic symmetries for molecular crystal structure prediction.

3

Graph Motif Parameters for Graph Neural Networks

In this chapter, we study the expressivity of GNNs with respect to injecting graph motif parameters. We prove that in terms of expressiveness, it is highly beneficial to inject the homomorphism basis of a parameter instead of the parameter itself, even for higher-order GNNs. The main contributions of this chapter include both theoretical and empirical results, which are particularly relevant for molecular datasets, where local structural motifs, such as rings or other functional groups, strongly influence the properties of a molecule. We also provide additional experimental results for other domains to support the generalizability of our approach.

The content of this chapter is largely based off of the work published in Jin et al. [1], with the theoretical results presented in Section 3.3 completed in collaboration with Matthias Lanzinger.

3.1 Introduction

Graph neural networks (GNNs) [72, 73] are a class of architectures for learning invariant functions on graphs. Their success in various domains [74–76] has led to the development of a number of architectures [19, 21, 23, 77]. Most of the architectures used in practice fall under the framework of message passing neural networks (MPNNs) [22], where the core idea is to iteratively update each node's representation based on the messages received from the node's neighbors.

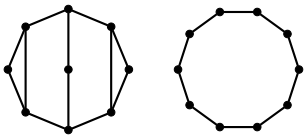
Despite their popularity, several limitations have been identified for GNNs, particularly pertaining to their expressive power. The expressive power of MPNNs

is upper bounded by the *1-dimensional Weisfeiler Leman graph isomorphism test (1-WL)* [23, 25]. Hence, all known inexpressiveness results for 1-WL apply to MPNNs. In particular, this implies that MPNNs cannot express a wide variety of *graph motif parameters*, which include functions that count the occurrences of basic graph patterns, such as paths or cycles. Since these patterns are abundant in real-world domains (e.g., the number of cycles in a molecule may dictate certain properties), a large body of work proposes enhancing the input graphs with such pattern counts directly as node features.

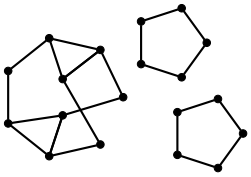
One line of work enriches the graph features with *subgraph* counts [26], while another enriches the graph features with *homomorphism* counts [27]. Although both approaches increase the expressive power of standard MPNNs, it remains unclear how these approaches compare to one another in terms of expressiveness gained. Furthermore, both of these approaches are sub-optimal in a certain sense.

We first show an example that presents two 1-WL indistinguishable graphs, where injecting (i.e. enriching features with) homomorphism counts is a better strategy than injecting subgraph counts.

Example 3.1.1. Consider the graphs G_1 and G_2 below, which are non isomorphic and indistinguishable by 1-WL. These graphs both contain *two* 5-cycles as *subgraphs*. Recall that $\mathsf{Sub}(F, G)$ is the number of times F occurs as a subgraph in G, and $\mathsf{Hom}(F, G)$ is the number of homomorphisms from F to G. In this case, we formally write $\mathsf{Sub}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_1) = \mathsf{Sub}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_2) = 2$.

G_1 

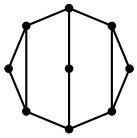
(a) $\mathsf{Sub}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_1) = 2$ and $\mathsf{Hom}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_1) = 120$.

G_2 

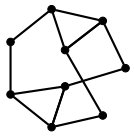
(b) $\mathsf{Sub}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_2) = 2$ and $\mathsf{Hom}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_2) = 140$.

Here, we see that adding subgraph counts for 5-cycles to 1-WL does not help distinguish these graphs, but adding homomorphism counts for 5-cycles does help in distinguish them since $\mathsf{Hom}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_1) \neq \mathsf{Hom}(\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc\!\!\!\!\bigcirc, G_2)$. △

How about the other direction? Are there graph pairs where injecting subgraph counts is the superior approach? We see that this is the case and provide an example below.

H_1 

(a) $\mathsf{Sub}(\hexagon, H_1) = 2$ and $\mathsf{Hom}(\hexagon, H_1) = 120$.

H_2 

(b) $\mathsf{Sub}(\hexagon, H_2) = 0$ and $\mathsf{Hom}(\hexagon, H_2) = 120$.

Example 3.1.2. Let H_1 be the first component of G_1 and H_2 be the first component of G_2 (from Example 3.1.1).

It is easy to verify that H_1 and H_2 cannot be distinguished by 1-WL. Furthermore we observe that:

$$\mathsf{Hom}(\hexagon, H_1) = \mathsf{Hom}(\hexagon, H_2) = 120.$$

Thus, adding homomorphism counts for $\hexagon$ does not help in distinguishing these graphs. Conversely, we have $\mathsf{Sub}(\hexagon, H_1) = 2$ and $\mathsf{Sub}(\hexagon, H_2) = 0$, which is sufficient to distinguish these graphs. △

Example 3.1.1 and Example 3.1.2 show that these two approaches are incomparable in terms of the expressiveness gain that they yield. This begs the question: what information can we inject into GNNs that strictly subsumes both approaches?

In this chapter, we answer this question by building on recent advances in graph theory that show that *graph motif parameters*, which encapsulate many functions of interest, can be expressed as a *linear combination of homomorphism counts* (cf. Section 2.1.1). In a nutshell, we show that enriching GNNs with the homomorphism counts of *all* structures in the "basis" of the target pattern yields strictly more expressive architectures without incurring any additional overhead in terms of computational complexity compared to existing approaches. In a sense, this targeted expressiveness gain is very precise and *fine-grained*. We prove a series of theoretical results on node-level and graph-level *motif parameters* and empirically validate them on both molecular and other standard benchmark datasets.

Main Results. The key results from this chapter can be summarized as follows:

- We study the expressivity of GNNs with respect to injecting graph motif parameters, and show that, in terms of expressiveness, it is highly beneficial to inject the homomorphism basis of a parameter instead of the parameter itself, even for higher-order GNNs (Section 3.3).

- This gain in expressiveness comes with additional benefits. In particular, we show how our method can naturally avoid redundant information (Section 3.3.3), and that it applies both on the graph-level and the node-level (Section 3.3.4).

- Our approach provides a flexible framework for injecting information far beyond the fixed pattern counts considered in previous work, e.g., graphlet counts or logical properties (Section 3.3.5).
- We empirically validate the theoretical findings and demonstrate the efficacy of our approach on a wide variety of benchmarks, where we observe state of the art results in expressivity and significant improvements in molecular property prediction and link-prediction tasks (Section 3.4).

From a broader perspective, this work strengthens the connections between recent graph-theoretical results and graph machine learning, informing future work for the targeted design of GNN architectures suitable for a domain of interest. We show that this is especially beneficial for molecular applications and highlight the importance of informed local representations.

3.2 Related Work

It is well-known that the expressive power of MPNNs is upper bounded by 1-WL in terms of graph distinguishability [23, 25]. Models such as graph isomorphism network (GIN) [23] can match this bound, whereas other popular architectures such as graph convolutional networks (GCNs) [19], or graph attention networks (GATs) [21] cannot.

This expressiveness limitation has motivated a large body of work in the graph machine learning literature, including the study of *higher-order* GNN architectures [25, 78–80] which generally match the expressive power of the k-WL algorithm for some $k > 1$. Some works consider graphs with *unique node features* [81], and others augment the node features with *random features* [82, 83] to achieve higher expressive power. Expressive power of GNNs has also been evaluated in terms of their ability to detect certain graph components, such as biconnected components [84]. Building on the classical literature of isomorphism testing and exploiting graph components (and associated graph decompositions) has led to efficient, isomorphism-complete learning algorithms for planar graphs [85].

There are different means for studying expressive power, and our work's primary focus is on measuring expressive power based on the ability to capture graph motif parameters. MPNNs are very limited in terms of their expressive power when it comes to counting patterns and so any function of this form goes beyond the ability of these architectures [86]. This has led to a rich line of work of GNN architectures that capture such pattern counts explicitly [26, 27, 87], and others that operate over subgraphs [88–90]. Zhang et al. [91] also recently provided an expressiveness hierarchy for subgraph GNNs via the so-called subgraph Weisfeiler-Leman tests.

Papp and Wattenhofer [92] investigated the expressiveness of specific properties, such as counts for all k-vertex substructures. They presented some specific results on the expressiveness gain such as showing that GNNs with all counts for k-vertex substructures can count k-cycles, but not $k-1$ cycles. Our framework provides unified answers for these specific expressivity questions. Barceló et al. [27] presented one of the early works highlighting the importance of using homomorphism counts with GNNs. Very recently, Welke et al. [93] propose a randomized approach that uses homomorphism counts to distinguish graphs *in expectation*. The work of Zhang et al. [94] is closely related to ours, where it has been shown that the expressiveness of GNN architectures under consideration can be characterized in terms of homomorphism counts. This further motivates our approach as will be discussed in Section 3.3.3.

Other works have studied homomorphism counts for more traditional machine learning models, such as support vector machines (SVMs) and random forest classifiers [56, 95]. Grohe [62] also investigated using homomorphism vectors to represent graphs.

3.3 All About The Homomorphism Basis

In this section, we examine the expressivity of GNNs with respect to graph motif parameters and introduce an efficient new method for incorporating these parameters by using their *homomorphism basis*. We show that our approach provides provably stronger expressive power, alongside a range of additional theoretical and practical advantages.

Expressing a Graph Parameter. We say that a GNN architecture Ψ can *express* a graph parameter f if for every pair of graphs G, H, there exists a model parametrization Ψ_θ such that $G \equiv_{\Psi_\theta} H$ implies $f(G) = f(H)$. In other words, there can be no situation where two graphs are equivalent to the model Ψ_θ but f is different on the two graphs. We are interested in the question of how the expressiveness gained by different sets of additional features compare to one another. Thus, we extend our notion of expressiveness to the case where Ψ additionally has access to a set of additional feature maps $\mathbb{A}$, i.e., invariant functions that map the input graph to a rational number. We say that a GNN architecture Ψ with $\mathbb{A}$ can *express* a graph parameter f if for every pair of graphs G, H, there exists a model parametrization Ψ_θ such that if $G \equiv_{\Psi_\theta} H$ and $a(G) = a(H)$ for all $a \in \mathbb{A}$, then $f(G) = f(H)$. For sets of features $\mathbb{A}, \mathbb{A}'$ and GNN Ψ, we say that Ψ with $\mathbb{A}$ is *at least as expressive* as Ψ with $\mathbb{A}'$ if the former expresses all functions that are expressed by the latter. Similarly, Ψ with $\mathbb{A}'$ is *strictly more expressive* than Ψ with $\mathbb{A}$ if the former is at least as expressive and can express strictly more functions than the latter.

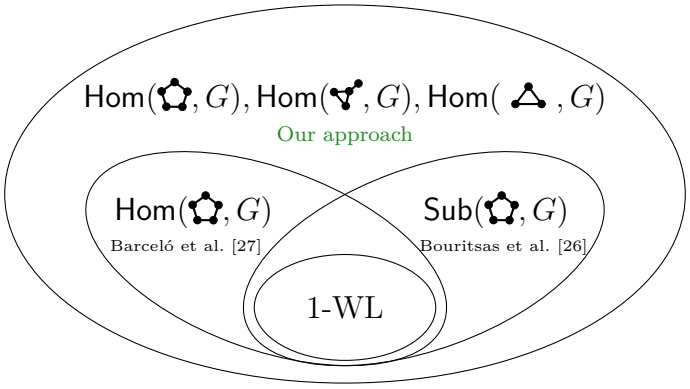


Figure 3.3: Expressiveness gain from injecting parameters. All inclusions are proper. All three features require (asymptotically) equivalent effort to calculate; they all can be computed in quadratic time (and no better).

3.3.1 It Is More Expressive

Recall that $\mathbb{A} = \{\mathsf{Sub}(⬠, \cdot)\}$ and $\mathbb{A}' = \{\mathsf{Hom}(⬠, \cdot)\}$ are incomparable if added to 1-WL expressiveness (Example 3.1.1 and Example 3.1.2). This also holds in general for arbitrary patterns that have more than one acyclic graph in their basis (follows from Theorem 3.3.1). At the same time, 1-WL with either kind of additional feature can be more expressive than plain 1-WL, as previously observed by Bouritsas et al. [26] and Barceló et al. [96].

We show that the homomorphism basis of $\mathsf{Sub}(⬠, \cdot)$ is more expressive than either approach (Figure 3.3). In fact, we prove a much more general and stronger statement. For (almost) every connected graph motif parameter Γ, providing the homomorphism basis $\mathbb{B}_\Gamma$ as additional features is *strictly* more expressive than injecting Γ.

Theorem 3.3.1. *Let Γ be a connected graph motif parameter and $k \geq 1$. Then k-WL with $\mathbb{B}_\Gamma$ is at least as expressive as k-WL with $\{\Gamma\}$. Moreover, if at least two functions in $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$ cannot be expressed by k-WL with $\{\Gamma\}$, then Ψ with $\mathbb{B}_\Gamma$ is strictly more expressive.*

We will prove a more general version of this theorem that holds not just for the steps of the k-WL hierarchy, but for all GNN models that follow a certain natural behavior with respect to homomorphisms. It is well known that $G \equiv_{k-\mathrm{WL}} H$ if and only if $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$ for all graphs F of treewidth at most k [97]. At a high level, the treewidth of a graph roughly describes how cyclic it is (i.e. acyclic graphs have a treewidth of 1, cyclic graphs have a tree width of 2, and graphs such as the 4-clique have a treewidth of 3). Furthermore, for every graph F with treewidth greater than k, there are G, H with $G \equiv_{k-\mathrm{WL}} H$ where

$\mathsf{Hom}(F, G) \neq \mathsf{Hom}(F, H)$, i.e., the class of graphs with treewidth at most k is the maximal class for which homomorphism vector equivalence is equivalent to k-WL equivalence [98]. This means that there is a concrete set of graphs whose homomorphism counts precisely determine the expressivity of the model. Zhang et al. [94] recently showed that this property also holds true for other commonly studied models that do not precisely match the expressiveness of the k-WL hierarchy, e.g., Subgraph GNNs and Local k-GNNs.

Definition 3.3.2. Let G, H be graphs and let $\mathcal{F}$ be a set of graphs. We say that G and H are *homomorphism indistinguishable* under $\mathcal{F}$, and we write $G \equiv_{\mathcal{F}} H$, if $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$ for all $F \in \mathcal{F}$.

Definition 3.3.3. We say that a class of graphs $\mathcal{F}$ is *closed under restriction to connected components* if for every graph $F \in \mathcal{F}$, every *maximal* connected component of F is in also in $\mathcal{F}$.

Definition 3.3.4. Let Ψ be a GNN model. We say that Ψ is *strongly homomorphism expressive* if there is a set of graphs $\mathcal{F}$ such that:

(i) for every pair of graphs G, H, it holds that $G \equiv_{\Psi} H$ if and only if $G \equiv_{\mathcal{F}} H$,

(ii) $\mathcal{F}$ is closed under restriction to connected components,

(iii) and $\mathcal{F}$ is the maximal set satisfying the first property.

We refer to $\mathcal{F}$ as the homomorphism expressiveness of Ψ.

As mentioned above, any GNN model with expressiveness equal to k-WL is strongly homomorphism expressive with $\mathcal{F}$ being the set of treewidth k graphs. Zhang et al. [94] showed that Local k-GNNs, local k-FGNNs, and Subgraph GNNs are also strongly homomorphism expressive.

In this light, it becomes clear that the expressiveness of a strongly homomorphism expressive Ψ with $\mathsf{Hom}(\mathsf{Spasm}(G))$ is strictly more expressive than Ψ with $\mathsf{Hom}(G, \cdot)$ exactly when there is some $F \in \mathsf{Spasm}(G)$ that is not in the homomorphism expressiveness of Ψ.

The theorem we will prove in the rest of this section is the following stronger version of Theorem 3.3.1.

Theorem 3.3.5. *Let Γ be a connected graph motif parameter and let Ψ be a strongly homomorphism expressive GNN model. Then Ψ with $\mathbb{B}_\Gamma$ is at least as expressive as Ψ with $\{\Gamma\}$. Moreover, if at least two functions in $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$ are not expressed by Ψ with $\{\Gamma\}$, then Ψ with $\mathbb{B}_\Gamma$ is strictly more expressive.*

Observe that the condition for the strictly more expressive case cannot be improved. If only one term $\mathsf{Hom}(F^*, \cdot)$ is not expressed by Ψ, then for every G, H with $\Gamma(G) = \Gamma(H)$ and $G \equiv_\Psi H$ we have from the former that

$$\sum_{F \in \mathsf{Supp}(\Gamma)} \alpha(F)\mathsf{Hom}(F, G) = \sum_{F \in \mathsf{Supp}(\Gamma)} \alpha(F)\mathsf{Hom}(F, H).$$

Since all terms in the sum, except for F^* are expressed by Ψ, those terms must be equal on both sides. Hence we are left with $\mathsf{Hom}(F^*, G) = \mathsf{Hom}(F^*, H)$ whenever $\Gamma(G) = \Gamma(H)$.

The technical challenge in proving Theorem 3.3.5 comes from the analysis of how equality under Ψ interacts with equality under Γ. The definition of strongly homomorphism expressive GNNs guarantees, that for every graph F for which $\mathsf{Hom}(F, \cdot)$ is not expressed by Ψ, there are graphs $G \equiv_\Psi H$ that differ on $\mathsf{Hom}(F, \cdot)$. On a technical level, the existence of such graphs needs to imply that there are graphs that are equivalent under both Ψ and Γ. Our proof shows how to construct such a graph whenever Γ is connected. Our argument requires connectedness of the graph motif parameter in order to control Γ in the construction. In particular, when Γ is connected, it is additive over disjoint union, which ultimately allows us to construct graphs through these operations in a way where Γ is controlled.

We will first observe a number of important properties of strongly homomorphism expressive GNNs based on well known properties of Hom. To this end we first define two operations on graphs. For graphs G, H we write $G + H$ for the disjoint union of G and H, i.e., the graph that is obtained by relabeling H to share no vertices with G and then combining their vertices and edges into a single graph. Moreover, we write $G \times H$ for their *categorical product*, which is the graph P with $V(P) = V(G) \times V(H)$ and $((a, u), (b, v)) \in E(P)$ if and only if $(a, b) \in E(G)$ and $(u, v) \in E(H)$. For positive integer c we write cG for the c-fold disjoint union of G with itself, and similarly G^c for the c-fold product.

Next, we will use the following results from Lovász [54]:

Proposition 3.3.6 (from Lovász [54])**.** *For graphs F, G, H, the following statements hold:*

- $\mathsf{Hom}(F + G, H) = \mathsf{Hom}(F, H)\mathsf{Hom}(G, H)$,
- $\mathsf{Hom}(F, G + H) = \mathsf{Hom}(F, G) + \mathsf{Hom}(F, H)$ *if F is connected,*
- $\mathsf{Hom}(F, G \times H) = \mathsf{Hom}(F, G)\mathsf{Hom}(F, H)$.

Using these statements, we can show the following:

Lemma 3.3.7. *Let Ψ be a strongly homomorphism expressive GNN. Let G, G', H, H' be graphs such that $G \equiv_\Psi G'$ and $H \equiv_\Psi H'$. The following two statements hold:*

1. $G + H \equiv_\Psi G' + H'$
2. $G \times H \equiv_\Psi G' \times H'$

Proof of Lemma 3.3.7. Let $\mathcal{F}$ be the homomorphism expressiveness of Ψ. Let F be a connected graph in $\mathcal{F}$. We have

$$\begin{aligned}\mathsf{Hom}(F, G + H) &= \mathsf{Hom}(F, G) + \mathsf{Hom}(F, H) \\ &= \mathsf{Hom}(F, G') + \mathsf{Hom}(F, H') \\ &= \mathsf{Hom}(F, G' + H')\end{aligned}$$

where the outer equalities are by Proposition 3.3.6 and the middle equality is by the assumption that $G \equiv_\Psi G'$ and $H \equiv_\Psi H'$. The argument for the second statement is analogous.

This completes the argument for connected F. Since $\mathcal{F}$ is closed under restriction to connected components, every graph $F' \in \mathcal{F}$ is a disjoint union of connected graphs in F. By the first point of Proposition 3.3.6, the equality on all the connected components of F' immediately extends to F'. □

Proof of Theorem 3.3.5. Let A, B be two graphs in $\mathsf{Supp}(\Gamma)$ that are not expressed by Ψ with Γ. Our plan will be to show that there are graphs G, H that are equivalent under Ψ and have $\Gamma(G) = \Gamma(H)$, but can be distinguished by either $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$.

Claim 1. Assuming Ψ with Γ does not express at least one of $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$, then there are graphs G_A, H_A, G_B, H_B that satisfy the following properties:

- $G_A \equiv_\Psi H_A$ and $G_B \equiv_\Psi H_B$
- $\mathsf{Hom}(A, G_A) \neq \mathsf{Hom}(A, H_A)$ and $\mathsf{Hom}(B, G_B) \neq \mathsf{Hom}(B, H_B)$
- $\max(\mathsf{Hom}(A, G_A), \mathsf{Hom}(A, H_A)) > \max(\mathsf{Hom}(B, G_A), \mathsf{Hom}(B, H_A))$

Proof of Claim 1: The first two items follow immediately by assumption that Ψ cannot express $\mathsf{Hom}(A, \cdot)$ and $\mathsf{Hom}(B, \cdot)$. For the last point, assume the two maxima are the same, say $\mathsf{Hom}(A, G_A) = \mathsf{Hom}(B, G_A) > \mathsf{Hom}(B, H_A)$. Recall that it is known that there always exists a graph X on which $\mathsf{Hom}(A, X) \neq \mathsf{Hom}(B, X)$ ([54]). Assume $\mathsf{Hom}(A, X) > \mathsf{Hom}(B, X)$ without loss of generality.

We have $G_A + X \equiv_\Psi H_A + X$ by Lemma 3.3.7. That is we have $\mathsf{Hom}(A, G_A + X) = \mathsf{Hom}(A, G_A) + \mathsf{Hom}(A, X) > \mathsf{Hom}(B, G_A) + \mathsf{Hom}(B, X) = \mathsf{Hom}(B, G_A + X)$. Similarly, $\mathsf{Hom}(A, H_A + X) = \mathsf{Hom}(A, H_A) + \mathsf{Hom}(A, X) < \mathsf{Hom}(A, G_A) + \mathsf{Hom}(A, X)$. So replacing G_A, H_A with $G_A + X, H_A + X$ satisfies the statement. $\triangleleft$

If $\Gamma(G_A) = \Gamma(H_A)$ or $\Gamma(G_B) = \Gamma(H_B)$, the theorem would be proven: if Γ would agree on one of the pairs, Ψ with Γ would not express the respective homomorphism count function. Assuming that Γ is indeed the same on these two pairs of graphs, we show that we can always construct graphs G, H that are still equivalent under Ψ and disagree on either $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$, but also have $\Gamma(G) = \Gamma(H)$. We make a key observation on Γ first. Recall that Γ is a *connected* graph motif parameter, thus from Proposition 3.3.6 together with Equation (2.1), we observe that $\Gamma(G + H) = \Gamma(G) + \Gamma(H)$.

For positive integer i, we define $\delta_i = \Gamma(G_A^i) - \Gamma(H_A^i)$ and similarly $\eta_i = \Gamma(H_B^i) - \Gamma(G_B^i)$ (recall, G^i is the i-fold categorical product). We will only care about η_0, which we therefore refer to as simply η. Assume, w.l.o.g., $\eta > 0$, otherwise switch G_B, H_B accordingly.

For i where $\delta_i > 0$ define $G_i := \eta G_A^i + \delta_i G_B$ an $H_i := \eta H_A^i + \delta_i H_B$.

Claim 2. For every positive integer i where $\delta_i > 0$ we have $\Gamma(G_i) = \Gamma(H_i)$ and $G_i \equiv_\Psi H_i$.

Proof of Claim 2: Recall from above that $\Gamma(cG) = c\Gamma(G)$. For the equality under Γ we then observe that

$$\begin{aligned}
\Gamma(G_i) - \Gamma(H_i) &= \eta\Gamma(G_A^i) + \delta_i\Gamma(G_B) - \eta\Gamma(H_A^i) - \delta_i\Gamma(H_B) \\
&= \delta_i\Big(\Gamma(G_B) - \Gamma(H_B)\Big) + \eta\Big(\Gamma(G_A^i) - \Gamma(H_A^i)\Big) \\
&= \delta_i(-\eta) + \delta_i\eta \\
&= 0
\end{aligned}$$

Since G_i and H_i are constructed from disjoint unions and products of graphs that are equivalent under Ψ, we also have $G_i \equiv_\Psi H_i$ by Lemma 3.3.7. $\triangleleft$

There are arbitrarily many i such that δ_i has positive sign (if the sign is negative for all δ_i, switch G_A, H_A). As above, we are done if $\mathsf{Hom}(A, G_i) \neq \mathsf{Hom}(A, H_i)$ for any i where $\delta_i > 0$ since we have $G_i \equiv_\Psi H_i$ and $\Gamma(G_i) = \Gamma(H_i)$ (and the same for B). Let us thus assume that this is not the case. For each of i we then get the following equations:

$$\begin{aligned}\eta\mathsf{Hom}(A, G_A^i) + \delta_i\mathsf{Hom}(A, G_B) &= \mathsf{Hom}(A, G_i) \\ &= \mathsf{Hom}(A, H_i) \\ &= \eta\mathsf{Hom}(A, H_A^i) + \delta_i\mathsf{Hom}(A, H_B) \qquad (3.1)\end{aligned}$$

$$\begin{aligned}\eta\mathsf{Hom}(B, G_A^i) + \delta_i\mathsf{Hom}(B, G_B) &= \mathsf{Hom}(B, G_i) \\ &= \mathsf{Hom}(B, H_i) \\ &= \eta\mathsf{Hom}(B, H_A^i) + \delta_i\mathsf{Hom}(B, H_B) \qquad (3.2)\end{aligned}$$

First, suppose that $\mathsf{Hom}(B, G_A^i) = \mathsf{Hom}(B, H_A^i)$. Simplifying Equation (3.2) gives $\mathsf{Hom}(B, G_B) = \mathsf{Hom}(B, H_B)$ which contradicts our choice of B, H_B, G_B. In the other case we have $\mathsf{Hom}(B, G_A^i) \neq \mathsf{Hom}(B, H_A^i)$. Solving both equations for $\frac{\eta}{\delta_i}$ (recall $\delta_i \neq 0$), identifying and rewriting gives:

$$\frac{\mathsf{Hom}(A, H_B) - \mathsf{Hom}(A, G_B)}{\mathsf{Hom}(B, H_B) - \mathsf{Hom}(B, G_B)} = \frac{\mathsf{Hom}(A, G_A^i) - \mathsf{Hom}(A, H_A^i)}{\mathsf{Hom}(B, G_A^i) - \mathsf{Hom}(B, H_A^i)} \qquad (3.3)$$

Observe that the left side of Equation (3.3) is independent of i, and therefore must be equal to some fixed constant ratio $r > 0$. Let a, b, c, d represent the values of $\mathsf{Hom}(A, G_A)$, $\mathsf{Hom}(A, H_A)$, $\mathsf{Hom}(B, G_A)$, and $\mathsf{Hom}(B, H_A)$, respectively. Then Equation (3.3) can be rewritten as:

$$\frac{a^i - b^i}{c^i - d^i} = r \qquad (3.4)$$

Recall from Claim 1 that $a \neq b$ and $c \neq d$. W.l.o.g. assume $a > b$ and $c > d$ (swapping values accordingly). From Claim 1 we also have that $\max(a, b) > \max(c, d)$, and thus $a > c$. Asymptotically, as $i \to \infty$, $r \to \infty$, since the ratio will be dominated by the numerator. This contradicts the assumption that r is a fixed finite constant.

That is, the ratio on the right-hand side of Equation (3.3) cannot stay constant. The equation was derived from the assumption that $\mathsf{Hom}(A, \cdot)$ or $\mathsf{Hom}(B, \cdot)$ both (individually) are the same for G_i and H_i. In consequence, there must be an i for which $\Gamma(G_i) = \Gamma(H_i)$ and $G_i \equiv_\Psi H_i$ but $\mathsf{Hom}(A, G_i) \neq \mathsf{Hom}(A, H_i)$ or $\mathsf{Hom}(B, G_i) \neq \mathsf{Hom}(B, H_i)$. Say Hom is different for A, then Ψ with Γ does not express $\mathsf{Hom}(A, \cdot)$.

□

We wish to emphasize that the gain in expressiveness observed in Theorem 3.3.1 is not only a matter of narrow theoretical increase. This is illustrated well by subgraph counting. Recall the basis of $\Gamma = \mathsf{Sub}($  $, \cdot)$ from Example 2.1.2. There, 1-WL with $\mathbb{B}_\Gamma$ does not only distinguish $\mathsf{Sub}($  $, \cdot)$ but also $\mathsf{Sub}($  $, \cdot)$ and $\mathsf{Sub}($  $, \cdot)$. In general, we observe the following.

Proposition 3.3.8. *Let Ψ be a GNN, let G be a graph, and let $F \in \mathsf{Spasm}(G)$. Then $\mathsf{Sub}(F, \cdot)$ can be expressed by Ψ with $\mathbb{B}_{\mathsf{Sub}(G,\cdot)}$.*

Proof of Proposition 3.3.8. Since $\mathsf{Sub}(G, \cdot)$ is a graph motif parameter with support equal to $\mathsf{Spasm}(G)$, we have that $\mathsf{Sub}(G, \cdot)$ is determined by $\mathsf{Hom}(\mathsf{Spasm}(G))$ since it functionally depends on the terms of $\mathsf{Hom}(\mathsf{Spasm}(G))$.

Observe that if $F \in \mathsf{Spasm}(G)$, then $\mathsf{Spasm}(F) \subseteq \mathsf{Spasm}(G)$. In particular, $H \in \mathsf{Spasm}(F)$ implies that $H \simeq F/\rho$ for some partition ρ of $V(F)$. Furthermore, $F \simeq G/\rho'$ where ρ' is some partition of $V(G)$ since $F \in \mathsf{Spasm}(G)$. Thus, ρ is a partition of blocks of ρ', and hence there is also a partition ρ'', obtained by applying the two partitions one after another, such that $H \simeq G/\rho''$. □

3.3.2 It Is Computationally Efficient

In practice, the additional features given by $\mathbb{A}$ to a GNN must be computed externally for each input graph. Curticapean, Dell and Marx [53] have shown that, in slightly simplified terms, the most efficient way to compute $\Gamma(G)$ is to compute the terms of the homomorphism basis and then obtain $\Gamma(G)$ via Equation (2.1). Hence, providing a GNN with $\mathbb{B}_\Gamma$ is not just more expressive but also computationally no more effort than providing Γ itself. The precise technical statements are intricate as they require tools from parameterized complexity theory. Formal details are provided below.

Popular subgraph counting implementations often exhibit significantly worse complexity than required to compute $\mathsf{Hom}(\mathsf{Supp}(\Gamma))$. Typically, to compute $\mathsf{Sub}(F, G)$ they have running times exponential in $|V(G)|$ [99]. Earlier works on injecting structural information in GNNs focus on particular settings where this worst-case behavior is not observed: sparse input graphs such as molecules and special patterns like cycles allow for good ad hoc algorithms for such graphs. For slightly more complex graphs or patterns, direct computation of Sub becomes intractable and computation via the homomorphism basis becomes much more efficient also in practice.

Now, we summarize key results from computational complexity theory that pertain to the arguments above. For the sake of a simpler and more streamlined

presentation, we will first focus on the special case of subgraph counting. Afterwards, we discuss how these results apply in general for graph motif parameters and how this relates to practical algorithms. To begin, we thus consider the following algorithmic problem.

#Sub
Input Graphs G, H
Output $\mathsf{Sub}(G, H)$

The problem is well known to be #P-hard (intuitively, the counting analogue of NP-hardness) and therefore considered to generally be intractable. It is then natural to consider for which pattern graphs – G in the definition of #Sub above – there might be specialized, more efficient algorithms. This particular question is primarily studied in the context of *parameterized complexity theory* and in particular the problem parameterized by the pattern graph G, defined as follows.

p-#Sub
Input Graphs G, H
Parameter G
Output $\mathsf{Sub}(G, H)$

In the parameterized setting, the classical requirement for tractability is relaxed. For graph G we write $|G|$ for $|V(G)| + |E(G)|$[1]. A *fixed-parameter tractable* (or fpt) algorithm for p-#Sub is an algorithm whose runtime is bounded by $f(|G|)|H|^{O(1)}$ where f is a computable function. That is, we allow some possible superpolynomial effort in G, as long as this yields an algorithm that is polynomial in the usually much bigger host graph H.

In general, it is easy to see that p-#Sub is hard for the complexity class #W[1] (counting the number of k-cliques in a graph is the canonical #W[1]-complete problem when parameterized by k). This class can be understood as the parameterized counting complexity equivalent of NP and it is a standard assumption in parameterized complexity that hardness for #W[1] implies that there is no fpt algorithm for the problem. This means that there is no general algorithm that can efficiently compute Sub. However, recent results of Curticapean, Dell and Marx [53] have greatly clarified the picture for which specific pattern graphs G an fpt algorithm is possible. Namely, the complexity of p-#Sub depends precisely on the treewidth of the graphs in $\mathsf{Spasm}(G)$.

[1]For the sake of complexity analysis, this is the size of a standard representation of a graph in a Turing Machine up to a log factor.

In particular, there is an algorithm that runs in time $f(|G|)O(|V(H)|^{k+1})$ where k is the maximal treewidth of graphs in the Spasm. The algorithm is conceptually simple once we know that $\mathsf{Sub}(G, \cdot)$ is a graph motif parameter.

1. First compute $\mathsf{Spasm}(G)$ and the corresponding coefficients. This clearly is independent of the graph H and takes $g(|G|)$ time. This can be seen as a preprocessing step from a practical point of view.

2. For each $F \in \mathsf{Spasm}(G)$, compute $\mathsf{Hom}(F, H)$ and then compute $\mathsf{Sub}(G, \cdot)$ according to Equation (2.3). Computing the number of homomorphisms $\mathsf{Hom}(F, H)$ using a tree decomposition is well understood and requires $O(|H|^k)$ time, where k is the treewidth of F [100].

Note that this algorithm is universal for any graph motif parameter. The only part that changes is how to compute the basis and the coefficients in the first step. This of course entirely depends on the kind of graph motif parameter, but by definition of a graph motif parameter, the basis does not depend on the input graph (H in this subgraph counting setting).

However, homomorphism counts are not merely an efficient way to compute subgraph counts. In a sense, there can be no more efficient way to compute subgraph counts. Formally speaking, both sides of Equation (2.1) are fpt-interreducible to each other. More precise analysis from Curticapean, Dell and Marx [53] reveals that these reductions can be performed very efficiently, and in fact so efficiently that we obtain very precise lower bounds for these problems.

Theorem 3.3.9 (Simplified from Curticapean, Dell and Marx [53])**.** *Let G be a graph and k be the maximum treewidth in $\mathsf{Spasm}(G)$. Then there is no $f(|G|)|H|^{o(k/\log k)}$ time algorithm that computes $\mathsf{Sub}(G, \cdot)$ for input H if #ETH holds.*[2]

Note that it is conjectured that the bound of $o(k/\log k)$ can be improved to $o(k)$ [101]. Under this conjecture, this shows that there is no way to really significantly improve on the computation of $\mathsf{Sub}(G, H)$ via the homomorphism basis as was described above.

Intuitively, this then means that graph motif parameters cannot be computed more efficiently than via their basis terms. Thus, it supports the claim from above that computing a graph motif parameter directly cannot be significantly less computational effort than computing the homomorphism counts for the basis graphs.

[2] #ETH is the counting version of the Exponential Time Hypothesis, a standard assumption in parameterized complexity.

Beyond #Sub First, we emphasize that the original result for Theorem 3.3.9 by Curticapean, Dell and Marx [53] is not restricted to subgraph counting but holds for arbitrary graph motif parameters (the parameter then becomes the encoding size of the basis). The same principle generalizes even further and is notably not limited to plain graphs. When we have directed graphs (and thus also directed patterns), the picture changes slightly but still no improvement over counting homomorphisms in the basis is possible [102]. Analogous statements hold with node and/or edge-labels, and general relational structures [103].

Practical Considerations. In the context of this thesis, it is of particular note that the fpt algorithm that is obtained from using the homomorphism basis is practical. To the best of our knowledge, there is no general implementation for #Sub that does not use the homomorphism basis and avoids exponential runtime in either $|V(G)|$ or $|E(G)|$. Popular subgraph counting algorithms such as VF2 [104] even report factorial time complexity in the worst case.

In contrast, the fpt algorithm consists of the two steps defined above, both of which are feasible for realistic pattern sizes. First, the Spasm and the coefficients need to be computed. Even naive algorithms that enumerate all possible partitions are efficient enough up to patterns of node size 12 without specialized hardware (from preliminary experiments). In a second step, the algorithm requires the computation of $\mathsf{Hom}(F, H)$ for each $F \in \mathsf{Spasm}(G)$. While this may superficially seem similar to #Sub, counting homomorphisms is much easier as they can be counted via dynamic programming on tree decompositions. In this setting, each entry in the decomposition can be computed independently, so they can be run in parallel, and combining the independent solutions requires only linear time. Theoretically, this process naively requires $O(|V(H)|^{k+1})$ time, where k is the treewidth of F (i.e. if $n = |V(H)|$, the complexity is $O(n^{k+1})$). However, there is always at least one edge in a bag of the tree decomposition of F by definition of its construction. Therefore, there are only $m = |E(H)|$ edges in H to map this edge to. This constrains the time complexity to $O(m \cdot n^{k-1})$, and since $m \leq n^2$, we get $m \cdot n^{k-1} \leq |H|^k$ and a simplified runtime of $O(|H|^k)$ for connected sparse graphs, where $|H| = n + m$.

In practice, this is magnitudes easier on most patterns, e.g., the worst case exponent for the 7-cycle C_7 using VF2 would be 7, whereas every graph in the basis of C_7 has treewidth 2 (see Figure A.3 for reference). In this thesis, the maximum treewidth in the basis of a motif parameter that we consider is 5 (for the 5-clique K_5). However, the vast majority of our experiments use motif parameters with a maximum treewidth 2 in their bases. Regarding size, the largest graph motif

parameter we use contains 8 nodes (for the 8-cycle C_8). Details on the size of the host graphs that we perform counting in are provided in Table 3.14.

Not only is counting homomorphisms easier than naive subgraph counting algorithms, but also much more flexible for large tasks. Recall that we need to compute $\mathsf{Hom}(F, H)$ for each $F \in \mathsf{Spasm}(G)$. However, these computations are entirely independent and can all be performed in parallel without any deeper algorithmic considerations. We demonstrate the efficacy of counting homomorphisms for GNN tasks by computing all the homomorphism counts for our experiments in significantly less time than is required for the training of the respective models (see Table 3.14 for details).

3.3.3 It Avoids Redundancy

While expressiveness is a natural criterion to consider with respect to injecting additional features, we also want to avoid redundant information. In this regard, the homomorphism basis is ideally suited as the expressiveness of GNNs with respect to homomorphism counts is well-understood. In practice, this means that instead of providing all of $\mathbb{B}_\Gamma$ as additional features to distinguish some target function Γ, we can often precisely determine which functions in $\mathbb{B}_\Gamma$ are not already distinguished by the GNN and only inject those.

Recent results provide a clear picture of when a GNN can distinguish a function $\mathsf{Hom}(F, \cdot)$. Neuen [98] recently showed that $(k-1)$-WL cannot distinguish any function $\mathsf{Hom}(F, \cdot)$ when k is the *treewidth* of F. A classic result by Dvorák [97] states that k-WL is enough to distinguish all such functions. Geerts and Reutter [105] showed an upper bound by k-WL for pattern counting based on the treewidth of patterns in the Spasm. This has significant practical implications: for architectures such as GIN [23] that reach 1-WL expressiveness, injecting only the cyclic graphs of $\mathbb{B}_\Gamma$ gives equivalent expressiveness to injecting all of $\mathbb{B}_\Gamma$. On the other hand, architectures like GAT [21] and GCN [19] are weaker than 1-WL and thus motivate providing all of $\mathbb{B}_\Gamma$ as additional features to the network. Recently, Zhang et al. [94] gave precise definitions for the sets of functions $\mathsf{Hom}(F, \cdot)$ that can (and cannot) be expressed by Local k-GNNs and Subgraph GNNs, thus showing precisely what part of $\mathbb{B}_\Gamma$ can already be expressed by the corresponding GNN architectures.

Furthermore, it is never necessary to consider homomorphism counts from disconnected graphs. Note that this is not trivial but particular to homomorphism counting, e.g., no similar simplification can be made for subgraph counting. See Section 3.3.5 below for an application of this observation.

Proposition 3.3.10. *Let Ψ be a GNN, let $\mathbb{G}$ be a set of graphs and let $\mathsf{CC}(\mathbb{G})$ be the set of the (maximal) connected components in $\mathbb{G}$. Then Ψ with $\mathsf{Hom}(\mathsf{CC}(\mathbb{G}))$ is at least as expressive as Ψ with $\mathsf{Hom}(\mathbb{G})$.*

Proof of Proposition 3.3.10. From the first point of Proposition 3.3.6 it follows that for any disconnected graph G, the homomorphisms from G into any other graph are determined fully by the homomorphism counts from the connected components of G. Proposition 3.3.10 then follows immediately. □

3.3.4 It Works for Node-Level Subgraph Information

Previous work has shown the efficacy of providing subgraph counts at node-level. We will use the term *anchored graph* to mean a graph G with a marked vertex $\circledast$. When G is anchored, we write $\mathsf{Sub}(G, H, v)$ for the number of subgraphs H' of H where $G \simeq H'$ such that the marked vertex maps to v in the isomorphism. In particular, Bouritsas et al. [26] propose labeling each node $v \in H$ with $\mathsf{Sub}(G', H, v)$ for every way G' of marking a graph G.

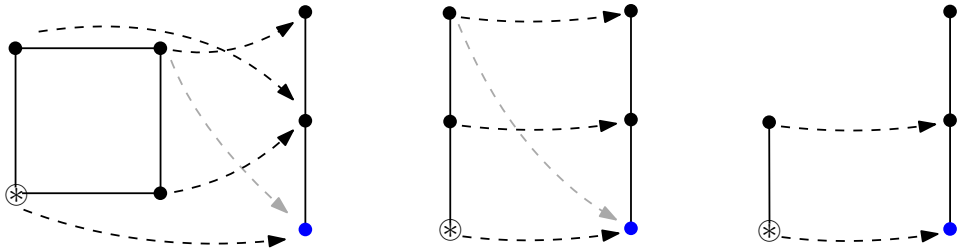
Note that local homomorphism counts of the graphs in the Spasm themselves are not enough to determine local subgraph counts via Equation (2.1). We demonstrate this claim by example. In Figure 3.4, we show why node-level homomorphism for the part of $\mathsf{Spasm}^{\circledast}$ without the second version of the 3 vertex path are insufficient. Figure 3.4a and Figure 3.4b illustrate the homomorphisms to two graphs H_1 and H_2 (gray arrows show mappings for a second homomorphism, and all cases have only 1 or 2 homomorphisms), both clearly without any  subgraphs. From left to right there are 2, 2, and 1 homomorphisms into H_1 and 2, 1, and 1 homomorphisms into H_2. Thus, expressing the subgraph count as a linear combination of homomorphism counts would require constants $\alpha_1, \alpha_2, \alpha_3$ such that

$$\alpha_1 \cdot 2 + \alpha_2 \cdot 2 + \alpha_3 \cdot 1 = \alpha_1 \cdot 2 + \alpha_2 \cdot 1 + \alpha_3 \cdot 1$$

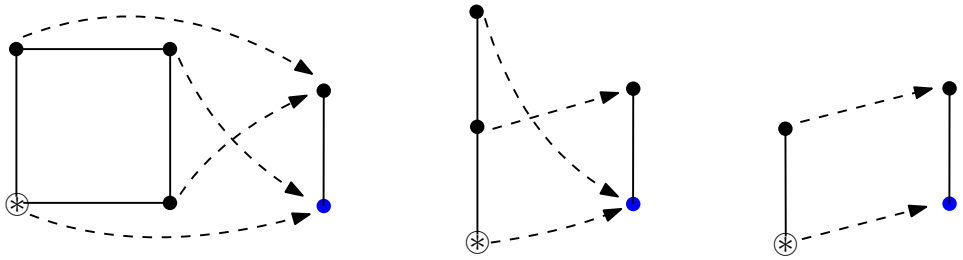
Clearly this implies $\alpha_2 = 0$, and we see that these graphs are insufficient to express the number of 4-cycles a particular vertex is part of.

Similarly, it is insufficient to only count the paths from the 3-vertex path anchored in the middle vertex, or even being provided only the sum of both 3-vertex path counts. The counterexample for both these cases is the n-vertex star graph: the path anchored at the middle vertex has $(n-1)^2$ homomorphisms to the center vertex, whereas the number of homomorphisms from all three other patterns is $n-1$.

Although Spasm itself is not enough to directly capture node-level subgraph counts, we can naturally extend the framework that we have been using so far to



(a) Homomorphisms from $\mathsf{Spasm}($  $)$ to H_1 with the anchor mapping to the blue vertex.

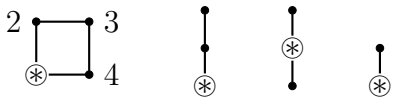


(b) Homomorphisms from $\mathsf{Spasm}($  $)$ to H_2 with the anchor mapping to the blue vertex.

Figure 3.4: Example showing that the Spasm itself is not enough to determine node-level subgraph counts for each orbit of the pattern graph.

this setting. For an anchored graph G, the anchor of quotient G/ρ is the vertex for the block $B \in \rho$ that contains the anchor of G. Analogously to the standard case, $\mathsf{Spasm}^{\circledast}(G)$ is the set of all (loop-free) quotient graphs of G.

Example 3.3.11. The $\mathsf{Spasm}($  $)$ contains ,  , and , whereas $\mathsf{Spasm}^{\circledast}($ $)$ consists of the four graphs:



In $\mathsf{Spasm}^{\circledast}$, there are two anchored versions of . The quotient that identifies vertices 2 and 4 has the anchor at the end of the path, while the quotient that identifies the $\circledast$ and 3 has its anchor on the middle vertex. △

With this definition of $\mathsf{Spasm}^{\circledast}$, we can now show that homomorphism counts are indeed also suited to node-level tasks and the previous arguments apply similarly in this setting.

Theorem 3.3.12. *Let G be an anchored graph and let H be a graph and $v \in V(H)$. Then*

$$\mathsf{Sub}(G, H, v) = \sum_{F \in \mathsf{Spasm}^{\circledast}(G)} \alpha_F \cdot \mathsf{Hom}(F, H)[\circledast \mapsto v]$$

where $\mathsf{Hom}(F, H)[\circledast \mapsto v]$ is the number of homomorphisms that map the anchor of F to $v \in V(H)$.

It follows that the arguments of the previous sections apply analogously also to this setting. With respect to Section 3.3.3 particularly, Lanzinger and Barceló [106] showed that $\mathsf{Hom}(F, H)[\circledast \mapsto v]$ can be computed from the node-level k-WL labeling for every graph F of treewidth k. Hence, in our example above, node-level homomorphism counts of  to 1-WL are sufficient additional information to determine node-level subgraph counts of the four cycle.

In addition to the counting functions introduced in the main body, we will require the counts for two more special kinds of homomorphisms. To that end, we write $\mathsf{Inj}(G, H)$ for the number of *injective* homomorphisms from G to H. Moreover, an *automorphism* of G is a homomorphism from G to G, and we write $\mathsf{Aut}(G)$ for the number of automorphisms of G. In particular, we will care about anchored graphs G and the number of automorphisms that map the anchor to itself, which we denote as $\mathsf{Aut}^{\circledast}(G)$. We write $\mathsf{Sub}(G, H)[\circledast = v]$ for the number subgraphs of G that are isomorphic to anchored graph G when v in H is set as the anchor of H. This is equivalent to $\mathsf{Sub}(G, H, v)$ as defined previously but aligns better with the proof.

Finally, we will need to reason about the lattice of vertex partitions of a graph (cf. Section 2.1.1). Let ρ, ρ' be partitions of $V(G)$. We say that ρ is a coarsening of ρ', or $\rho \geq \rho'$ if for every block $B' \in \rho'$ there is a block $B \in \rho$ such that $B' \subseteq B$.

Proof of Theorem 3.3.12. With the appropriate setup, we can follow the same steps as the analogous graph level proofs by Curticapean, Dell and Marx [53]. Recall that we want to show that

$$\mathsf{Sub}(G, H)[\circledast = v] = \sum_{F \in \mathsf{Spasm}^{\circledast}(G)} \alpha_F \cdot \mathsf{Hom}(F, H)[\circledast \mapsto v].$$

We start from two basic observations. First,

$$\mathsf{Hom}(G, H)[\circledast \mapsto v] = \sum_{\rho} \mathsf{Inj}(G/\rho, H)[\circledast \mapsto v] \tag{3.5}$$

where the sum ranges over all partitions of $V(G)$. The second is that

$$\mathsf{Inj}(G, H)[\circledast \mapsto v] = \mathsf{Aut}^{\circledast}(G)\mathsf{Sub}(G, H)[\circledast = v] \tag{3.6}$$

where $\mathsf{Aut}^{\circledast}$ is the number of automorphisms of G that map the anchor to itself. Observe that $\mathsf{Hom}(G, H)[\circledast \mapsto v]$ is the upward zeta transformation of $\mathsf{Inj}(G/\rho, H)[\circledast \mapsto v]$ on the partition lattice from the trivial partition $\perp$ where every block is a singleton.

Möbius transformation of Equation (3.5) then gives us the expression

$$\mathsf{Inj}(G/\rho, H)[\circledast \mapsto v] = \sum_{\rho' \geq \rho} (-1)^{|\rho| - |\rho'|} \left(\prod_{B \in \rho'} (\lambda(\rho, \rho', B) - 1)! \right) \mathsf{Hom}(G/\rho', H)[\circledast \mapsto v]$$

which, fixing $\rho = \bot$ is

$$\mathsf{Inj}(G,H)[\circledast \mapsto v] = \sum_{\rho' \geq \rho} (-1)^{|V(G)|-|\rho'|} \left(\prod_{B \in \rho'} (|B|-1)! \right) \mathsf{Hom}(G/\rho', H)[\circledast \mapsto v]$$

Note that by definition, G/ρ' is isomorphic to a term in $\mathsf{Spasm}^{\circledast}(G)$ for every partition ρ'. Collecting the terms for every graph in $\mathsf{Spasm}^{\circledast}(G)$, we then get

$$\mathsf{Inj}(G,H)[\circledast \mapsto v] = \sum_{F \in \mathsf{Spasm}^{\circledast}(G)} (-1)^{|V(G)|-|V(F)|} \mathsf{Hom}(F,H)[\circledast \mapsto v] \left(\prod_{\substack{B \in \rho' \\ G/\rho' \simeq F}} (|B|-1)! \right)$$

Combining this with Equation (3.6), we get desired expression for $\mathsf{Sub}(G,H)[\circledast = v]$. In particular, the coefficient α_F is

$$\alpha_F = \frac{(-1)^{|V(G)|-|V(F)|}}{\mathsf{Aut}^{\circledast}(G)} \left(\prod_{\substack{B \in \rho' \\ G/\rho' \simeq F}} (|B|-1)! \right)$$

□

To supplement the complexity discussion from Section 3.3.2, we note that it is straightforward to reduce graph-level subgraph counting to node-level subgraph counting. Suppose we want to count $\mathsf{Sub}(G,H)$. Create the anchored graph G' by adding a new vertex v' to G that is connected to every other vertex. Additionally, set v' as the anchor of G'. Similarly, create H' by adding a new vertex u' to H that is connected to all other vertices. It is then easy to see that $\mathsf{Sub}(G,H) = \mathsf{Sub}(G',H',u')$. Furthermore, we can observe that since the new vertex v' is connected to every other vertex, it can never be contracted with any other vertex to create a loop-free quotient. That is, every graph $F \in \mathsf{Spasm}(G')$ is a graph in $\mathsf{Spasm}(G)$ with an additional vertex that is connected to all other vertices. This means that the maximal treewidth in $\mathsf{Spasm}(G')$ is at most 1 higher than $\mathsf{Spasm}(G)$. Hence, we have a linear reduction from graph-level subgraph counting to node-level subgraph counting that increases the treewidth of the pattern by at most 1, i.e., node-level subgraph counting also critically depends on the treewidth of the Spasm.

3.3.5 It Goes Beyond Pattern Counting

We have mainly focused on the setting where a GNN is provided additional information motivated by some target structure that is considered important to the

task (e.g., cycles). However, the framework of graph motif parameters also motivates a natural alternative approach that is not dependent on such structure selection.

For some (decidable) graph property ϕ, let $\mathsf{IndSub}_k^\phi(G)$ denote the number of induced subgraphs of G on k vertices that satisfy ϕ. Through the choice of ϕ – e.g., connectedness, hamiltonicity, planarity – IndSub_k^ϕ can cover an immense variety of graph parameters. For example, if ψ is the property of being isomorphic to a fixed graph F, then $\mathsf{IndSub}(F, G) = \mathsf{IndSub}_{|V(F)|}^\psi(G)$. Similarly, when ϕ is connectedness, $\mathsf{IndSub}_k^\phi(G)$ counts the number of k-graphlets in G, which is a popular metric in graph analysis [99]. It is known that all such functions IndSub_k^ϕ are indeed graph motif parameters [107].

For some settings where there is no obvious kind of structure to inject information for, this presents an attractive alternative. In fact, it is possible to provide practical amounts of additional information to a GNN, such that it can distinguish *all* functions IndSub_k^ϕ for small k. In concrete terms, by inspection of the arguments by Roth and Schmitt [107] in combination with Proposition 3.3.10, we observe the following:

Proposition 3.3.13. *Let Ψ be a GNN, let ϕ be a decidable graph property, and let $k \geq 1$. Let $\Omega_{\leq k}^{\mathsf{con}}$ be the set of connected graphs with at most k vertices. Then IndSub_k^ϕ can be expressed by Ψ with $\mathsf{Hom}(\Omega_{\leq k}^{con})$.*

Proof of Proposition 3.3.13. Investigation of the proof of Theorem 12 from Roth and Schmitt [107] reveals that $\mathsf{Supp}(\mathsf{IndSub}_k^\phi) \subseteq \Omega_k$. Thus, in relation to our setting, this means that a GNN Ψ with $\mathsf{Hom}(\Omega_k)$ expresses IndSub_k^ϕ. We then apply Proposition 3.3.10 to observe the final statement. □

Graph Motif Parameters for Beyond Counting Patterns. The possible applications of our approach range even further. Although our exposition primarily focuses on subgraph and homomorphism counts for GNNs, our main objective is to consider the overarching class of graph motif parameters, which has broad implications beyond fixed pattern counting. Counting the satisfying assignments for certain fragments of first-order logic in a graph is also a graph motif parameter [108]. This presents the possibility of targetedly injecting information to express logic-specified properties.

In a molecular setting, this provides an attractive way of incorporating additional domain knowledge into a given task at hand.

Example 3.3.14. We can express whether an atom (i.e., node) in a molecular dataset is part of an alcohol group with the following first-order formula:

$$\psi(x, y, z) := (\mathsf{bond}(x, y) \wedge \mathsf{bond}(y, z) \wedge \mathsf{O}(y) \wedge \mathsf{H}(z))$$

Similarly, we can express whether an atom is part of an acyl halide functional group as:

$$\begin{aligned}\tau(x, y, z, h) := (\mathsf{bond}(x, y) \wedge \mathsf{dbond}(y, z) \wedge \mathsf{bond}(y, h) \wedge \\ \mathsf{C}(y) \wedge \mathsf{O}(z) \wedge (\mathsf{F}(h) \vee \mathsf{Cl}(h) \vee \mathsf{Br}(h) \vee \mathsf{I}(h)))\end{aligned}$$

where $\mathsf{O}, \mathsf{H}, \mathsf{C}, \mathsf{F}, \mathsf{Cl}, \mathsf{Br}$, and I represent corresponding elements in the periodic table, bond denotes a single bond between atoms, and dbond denotes a double bond. We can also disjunctively combine these properties to express that an atom is part of an alcohol or acyl halide functional group:

$$\phi(x, y, z, h) := \psi(x, y, z) \vee \tau(x, y, z, h)$$

The number of satisfying assignments of $\phi(x, y, z, h)$ is a graph motif parameter that can be expressed as a linear combination of homomorphism counts. This can be done at node-level and for specific variables, e.g., how often variable x is mapped to node v in a satisfying assignment. △

Example 3.3.14 captures a general property that goes beyond fixed pattern counting, yet our method can still precisely identify which homomorphism counts need to be injected in order to lift a model to the desired level of expressivity. By considering the broader framework of graph motif parameters, we provide an exact and principled way of achieving targeted expressivity gains for GNN architectures.

Overall, we summarize the theoretical results of this chapter using the following example.

Example 3.3.15. Consider a scenario where we want to increase the expressivity of GIN to be able to capture the graph motif parameter above in Example 3.3.14. Our results show that:

1. Injecting the homomorphism basis of this parameter will be more expressive than injecting the parameter itself (Theorem 3.3.1).

2. Computing the basis is the most efficient way to compute a graph motif parameter, so the extra expressivity achieved from point 1 comes at no additional computational cost (Section 3.3.2).

3. Since GIN already matches 1-WL in expressivity, we only need to inject the basis elements that are above 1-WL. Specifically, this means that we can exclude features like $\mathsf{Hom}(\bullet\!\!-\!\!\bullet, \cdot)$ from the set of features that must be injected into GIN (Section 3.3.3).

4. Following from Points 1 and 3, we have now precisely identified which homomorphism counts need to be injected into GIN in order to lift this model to the desired level of expressivity.

5. This approach extends far beyond subgraph pattern counting, as evidenced by Example 3.3.14 itself, which presents a graph motif parameter that is a disjoint union. △

3.4 Experimental Evaluation

Now, we will present a series of experiments on real-world tasks and architectures that leverage the theoretical results presented above. We begin by first exploring how to best encode these homomorphism basis counts as features into a GNN model (Section 3.4.1). Then, we will present our results for molecular property prediction tasks with the ZINC (Section 3.4.2) and QM9 (Section 3.4.3) molecular datasets. Finally, we show the generalizability of our approach by providing results for link prediction on the COLLAB dataset (Section 3.4.4) and a dedicated expressiveness experiment on the synthetic BREC expressivity benchmark (Section 3.4.5). Additional experimental details and model hyperparameters for each experiment are reported in Appendix A.

3.4.1 Generating Encodings for Homomorphism Counts

In multiple of the experiments that we show below in this section, we inject homomorphism counts of graphs in $\mathsf{Spasm}(G)$ at node-level. In the terminology of Section 3.3.4, for every graph $F \in \mathsf{Spasm}(G)$, we pick an arbitrary vertex as its anchor to create F'. For each vertex v of input graph H we then compute the vector

$$\mathbf{f}_v := (\mathsf{Hom}(F', H)[\circledast \to v])_{F \in \mathsf{Spasm}(G)}$$

for some fixed order on the graphs of $\mathsf{Spasm}(G)$. We then associate each $\mathbf{f}_v$ to node v as additional features. The details in which this vector is added depend on the experiment and is described in the respective experiment section.

The motivation for using this approach is that it splits the number of total homomorphisms up "disjointly," i.e., no homomorphism is counted twice at different vertices. In particular this means that

$$\sum_{v \in V(H)} \mathbf{f}_v = (\mathsf{Hom}(F, H))_{F \in \mathsf{Spasm}(G)}. \tag{3.7}$$

That is, the global counts are simply a sum over the individual feature vectors, without the need to correct for potential double counting of homomorphisms. This thus presents a compromise of preserving the global counts as truthfully as possible, while still providing some local node-level information. That said, as discussed in Section 3.3.4, precise node-level information requires more fine-grained homomorphism information.

Below, we provide the series of experimental results that led us to this formulation. Aligning with the overall theme of focusing on molecular applications, all the experiments below are conducted on the ZINC molecular property prediction dataset, which we will now describe.

Dataset Setup. ZINC [33, 41] is a graph regression task for predicting the constrained solubility of molecules. The constrained solubility of a molecule is given by the formula: *logP* - *SAS* - *cycles*, where *logP* is the water-octanol partition coefficient, *SAS* is the synthetic accessibility score, and *cycles* denotes the number of cycles with more than six atoms.

For each graph, the original vertex features denote the type of heavy atom for a given node, and the edge features denote the type of bond between two nodes. In all experiments, we follow the experimental protocol from Dwivedi et al. [33] and use a subset of the ZINC dataset that contains 12,000 individual graphs, which are split into 10,000 graphs for training, 1,000 graphs for validation, and 1,000 graphs for testing. For the following set of experiments in this subsection, we use the version of the ZINC dataset that *does* include edge features. This distinction is important, since there are two different ZINC benchmarks, one that uses edge features and one that does not.

Selection of Graph Motif Parameters. We experiment with a range of different graph motif parameters and exclusion criteria. Since we are performing a molecular property prediction task, and it is known that cycles (i.e. rings) play an important role in molecular structures, we choose to evaluate the performance of injecting the Spasm of different cycle lengths. These include C_5, C_6, C_7, and C_8. We also experiment with different unions, such as $C_6 \cup C_7$ and $C_7 \cup C_8$, and specific

basis pattern exclusion criteria. These include excluding 1-WL basis patterns to evaluate the practical relevance of Section 3.3.3. We also experiment with some more fine-grained exclusion, such as excluding the  basis element from $\mathsf{Spasm}(C_7)$ and excluding the  basis element from $\mathsf{Spasm}(C_8)$. These motifs resemble the structures of some known chemical motifs, so their exclusion may impact performance. A full set of graphs within each of these Spasm sets is provided in Appendix A.2 for reference.

Model Selection and Training. All results in this section use the GIN-E model, which is a variation of GIN that accounts for edge features [109]. Since this is a graph regression task, our evaluation metric is the mean absolute error (MAE) of the predicted results, and lower is better. All models are run with the same general hyperparameters, which are reported in Appendix A.1.1. All results are reported as the average of 5 runs at different seeds.

Experiment 1: Linear Encodings with Variable Dimension Sizes. We begin by first applying a simple Linear encoding on our homomorphism basis features. Recall the formulation of a graph motif parameter from Equation (2.1), which contains both α coefficient terms and raw Hom counts:

$$\Gamma(G) = \sum_i \alpha_i \mathsf{Hom}(F_i, G)$$

First, we experiment with several different variations of how to encode the respective α and raw Hom count values using a simple Linear encoder. These features are then directly concatenated to the initial node feature embeddings before being passed through the rest of the model. Embeddings from the initial node features given by the dataset (i.e. atom types) are denoted as $h^{(0)}$. Note that the initial node and edge features are commonly passed through a one-hot encoding, then a Linear layer [33]. After the message passing layers, an MLP is often used at the end as a decoder.

For our first set of experiments, we introduce four different encoding techniques: $\text{AlphaHom}^{\text{Lin}}$, $\text{MultHom}^{\text{Lin}}$, $\text{PosHom}^{\text{Lin}}$, and $\text{SubHom}^{\text{Lin}}$. In $\text{AlphaHom}^{\text{Lin}}$, the homomorphism counts and alpha coefficients are separately passed through a Linear layer, then concatenated into a single feature vector at the end. In $\text{MultHom}^{\text{Lin}}$, we perform an element-wise multiplication of the α and Hom features before passing the resulting vector through a Linear layer. For $\text{PosHom}^{\text{Lin}}$, we apply the sine/cosine positional encoding technique from Vaswani et al. [6] to normalize our homomorphism features. In this initial formulation, the positional encodings

are reduced down to the original dimension size of the homomorphism count vector using an additional Linear layer.

Finally, for $\text{SubHom}^{\text{Lin}}$, we use the total subgraph counts for the entire pattern graph as well, since we are able to recover this information from the homomorphism basis. Note that we use the global subgraph counts, not the node-level ones (i.e. orbitals). Therefore, each vertex will have the same subgraph count features for each respective pattern. This mean that for basis pattern C_5, each node will have one feature denoting the number of 5-cycles in the entire graph. For unions, such as $C_6 \bigcup C_7$, each node will have 2 features, where one denotes the number of 6-cycles and one denotes the number of 7-cycles in the entire graph. We again use a positional encoding for $\text{SubHom}^{\text{Lin}}$ features in the same manner as described for $\text{PosHom}^{\text{Lin}}$.

We summarize these encoding methods below. Note that transformations act on the features within the square brackets, the + operator denotes a vector concatenation, and the $*$ operator denotes an element-wise vector multiplication. The array of raw homomorphism basis count values are denoted as Hom, the array of alpha coefficient values are denoted as α, and the array of subgraph count values are denoted as Sub. In the results tables, d_h denotes the dimension of the additional embeddings for each of the corresponding models (i.e. the output dimension of the Linear layer).

- $\text{AlphaHom}^{\text{Lin}}$: $h^{(0)}$ + Linear[Hom] + Linear[α]
- $\text{MultHom}^{\text{Lin}}$: $h^{(0)}$ + Linear[$\alpha * \mathsf{Hom}$]
- $\text{PosHom}^{\text{Lin}}$: $h^{(0)}$ + Linear[Linear[PE[$\alpha * \mathsf{Hom}$]]]
- $\text{SubHom}^{\text{Lin}}$: $h^{(0)}$ + Linear[Linear[PE[$\alpha * \mathsf{Hom}$]]] + Linear[Sub]

Results. In general, we see that using the basis features from larger cycles tends to improve performance (Table 3.1). This is likely because the basis sets of larger cycles contain the basis sets for some smaller cycles as well (e.g. $\mathsf{Spasm}(C_6) \subset \mathsf{Spasm}(C_8)$), so the model is able to inherently capture more information (Proposition 3.3.8).

That said, the results are not entirely consistent with the expected behavior across all models. This is reflected in the $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ basis features performing worse than the $\mathsf{Spasm}(C_6) \cup \mathsf{Spasm}(C_7)$ basis features in $\text{MultHom}^{\text{Lin}}$ and $\text{PosHom}^{\text{Lin}}$, even though the latter set fully contains the former. This may be due to the fact that for the larger feature sets, the model is not sufficiently large enough to process the extra information.

Table 3.1: Different encoding techniques using a Linear layer, where the output embedding dimension (d_h) depends on the size of the homomorphism basis features.

Basis		**AlphaHom$^{\text{Lin}}$**		**MultHom$^{\text{Lin}}$**		**PosHom$^{\text{Lin}}$**		**SubHom$^{\text{Lin}}$**	
Pattern	Excl.	d_h	MAE	d_h	MAE	d_h	MAE	d_h	MAE
C_5	None	6	0.1717±0.0144	3	0.1748±0.0034	3	0.1848±0.0074	4	0.1883±0.0051
C_6	None	20	0.1670±0.0060	10	0.1775±0.0087	10	0.1781±0.0070	11	0.1804±0.0082
C_6	1-WL	12	0.1729±0.0042	6	0.1931±0.0046	6	0.1653±0.0029	7	0.1830±0.0101
C_7	None	24	0.1203±0.0029	12	0.1235±0.0024	12	0.1755±0.0051	13	0.1316±0.0070
C_7		22	0.1267±0.0062	11	0.1238±0.0027	11	0.1309±0.0193	12	0.1305±0.0079
C_8	None	70	0.1560±0.0086	35	0.1614±0.0109	35	0.1639±0.0075	36	0.1535±0.0124
C_8		68	0.1542±0.0041	34	0.1765±0.0065	34	0.1688±0.0056	35	0.1540±0.0044
C_8	1-WL	56	0.1517±0.0096	28	0.1542±0.0063	28	0.1533±0.0071	29	0.1550±0.0079
$C_6 \cup C_7$	None	44	0.1117±0.0095	22	0.1260±0.0082	22	0.1264±0.0034	24	0.1211±0.0052
$C_6 \cup C_7$	1-WL	36	0.1239±0.0093	18	0.1193±0.0077	18	0.1201±0.0102	20	0.1275±0.0043
$C_7 \cup C_8$	None	94	0.1125±0.0034	47	0.1532±0.0056	47	0.1423±0.0034	49	0.1274±0.0003
$C_7 \cup C_8$	1-WL	80	0.1108±0.0024	40	0.1436±0.0074	40	0.1498±0.0085	42	0.1189±0.0036

Furthermore, there is also no consistent trend when analyzing the exclusion basis elements. For instance, sometimes excluding specific feature elements helps, whereas other times it hurts performance. This speaks to the difficulty of specific feature engineering and suggests that a simple Linear encoding technique may be too elementary to realize the full potential of the theoretical results presented above.

Experiment 2: Linear Encodings with Fixed Dimension Sizes. Given the results from Experiment 1, we next adjust all models slightly to use a fixed output dimension size for all homomorphism feature encodings. We select $d = 16$ for our output dimension size across all models. However, for models such as AlphaHom$^{\text{Lin}}$, which include separate feature vectors for Hom and α, we end up with a total additional feature size of 32 due to the concatenation of the two embeddings. Similar adjustments are applied for SubHom$^{\text{Lin}}$. Note that on average, there is much less variance in dimension size between different basis sets with this setup. Recall that all models still use a simple Linear encoding. We restate the model definitions below for reference:

- AlphaHom$^{\text{Lin}}$: $h^{(0)}$ + Linear[Hom] + Linear[α]
- MultHom$^{\text{Lin}}$: $h^{(0)}$ + Linear[$\alpha * \mathsf{Hom}$]
- PosHom$^{\text{Lin}}$: $h^{(0)}$ + Linear[Linear[PE[$\alpha * \mathsf{Hom}$]]]
- SubHom$^{\text{Lin}}$: $h^{(0)}$ + Linear[Linear[PE[$\alpha * \mathsf{Hom}$]]] + Linear[Sub]

Table 3.2: Different encoding techniques using a Linear layer, where the output encoding dimensions are fixed to 16. Total feature embedding sizes (d_h) are provided.

Basis		**AlphaHom$^{\text{Lin}}$**		**MultHom$^{\text{Lin}}$**		**PosHom$^{\text{Lin}}$**		**SubHom$^{\text{Lin}}$**	
Pattern	Excl.	d_h	MAE	d_h	MAE	d_h	MAE	d_h	MAE
C_5	None	32	0.1770±0.0017	16	0.1694±0.0057	16	0.1709±0.0179	17	0.1783±0.0057
C_6	None	32	0.1660±0.0032	16	0.1742±0.0033	16	0.1744±0.0086	17	0.1707±0.0015
C_6	1-WL	32	0.1572±0.0052	16	0.1832±0.0024	16	0.1816±0.0103	17	0.1717±0.0072
C_7	None	32	0.1287±0.0084	16	0.1197±0.0030	16	0.1259±0.0026	17	0.1283±0.0106
C_7		32	0.1304±0.0098	16	0.1175±0.0072	16	0.1520±0.0236	17	0.1391±0.0057
C_8	None	32	0.1498±0.0043	16	0.1581±0.0059	16	0.1652±0.0077	17	0.1508±0.0043
C_8		32	0.1485±0.0052	16	0.1771±0.0054	16	0.1571±0.0079	17	0.1587±0.0010
C_8	1-WL	32	0.1509±0.0021	16	0.1703±0.0136	16	0.1606±0.0066	17	0.1603±0.0057
$C_6 \cup C_7$	None	32	0.1160±0.0053	16	0.1707±0.0072	16	0.1262±0.0060	18	0.1282±0.0052
$C_6 \cup C_7$	1-WL	32	0.1142±0.0042	16	0.1339±0.0109	16	0.1174±0.0058	18	0.1315±0.0039
$C_7 \cup C_8$	None	32	0.1118±0.0042	16	0.1456±0.0096	16	0.1269±0.0126	18	0.1193±0.0039
$C_7 \cup C_8$	1-WL	32	0.1093±0.0045	16	0.1404±0.0059	16	0.1458±0.0040	18	0.1147±0.0029

Results. We see from Table 3.2 that using a fixed dimension size yields results that are more stable than the ones reported in Table 3.1. The best AlphaHom$^{\text{Lin}}$ configuration in Table 3.2 outperforms the best AlphaHom$^{\text{Lin}}$ configuration in Table 3.1 as well. Furthermore, results for AlphaHom$^{\text{Lin}}$ and PosHom$^{\text{Lin}}$ are starting to align better with expected behavior. Specifically, we see that excluding 1-WL basis components generally does not have a significant impact, while excluding the  basis element from $\mathsf{Spasm}(C_7)$ produces a drop in performance.

This result highlights the fact that smaller feature vectors seem to be somewhat preferable for these simple 4-layer GIN-E models. Additionally, it is clear that the practical choices for how these features are encoded greatly influence the empirical results. As mentioned above, a Linear layer is a very elementary embedding technique, so it may not be enough to fully take advantage of the homomorphism basis features being provided.

Experiment 3: Encoding Homomorphism Basis Features with an MLP. In general, a simple Linear embedding layer cannot capture many of the intricate interactions between features. Therefore, we adjust AlphaHom$^{\text{Lin}}$ and MultHom$^{\text{Lin}}$ models from above and replace the Linear encoder with a 2-layer MLP. These two models are selected since they are the best performing models in Tables 3.1 and 3.2 that do not include explicit subgraph information (i.e. SubHom$^{\text{Lin}}$). For AlphaHom$^{\text{MLP}}$, we also evaluate a version when we concatenate the α and Hom vectors before passing the concatenated vector through the MLP to reduce the

total feature dimension size, since our results from Experiment 2 suggest that this may help performance. This modified model is denoted as $\text{AlphaHom}_{+}^{\text{MLP}}$. Our experimental setup is as follows:

- $\text{AlphaHom}^{\text{MLP}}$: $h^{(0)} + \text{MLP}[\mathsf{Hom}] + \text{MLP}[\alpha]$
- $\text{AlphaHom}_{+}^{\text{MLP}}$: $h^{(0)} + \text{MLP}[\mathsf{Hom} + \alpha]$
- $\text{MultHom}^{\text{MLP}}$: $h^{(0)} + \text{MLP}[\alpha * \mathsf{Hom}]$

We also provide the results from the following models as reference:

- $\text{AlphaHom}^{\text{Lin}}$: $h^{(0)} + \text{Linear}[\mathsf{Hom}] + \text{Linear}[\alpha]$ (from Experiment 1 (Table 3.1))
- $\text{MultHom}^{\text{Lin}}$: $h^{(0)} + \text{Linear}[\alpha * \mathsf{Hom}]$ (from Experiment 1 (Table 3.1))

Table 3.3: Experimental results using a 2-layer MLP instead of a single-layer Linear encoding. Note that there are two slightly different versions of $\text{AlphaHom}^{\text{MLP}}$, which affect the resulting feature encoding dimension sizes.

Basis		**AlphaHom$^{\text{Lin}}$**		**AlphaHom$^{\text{MLP}}$**		**AlphaHom$_{+}^{\text{MLP}}$**		**MultHom$^{\text{Lin}}$**		**MultHom$^{\text{MLP}}$**	
Pattern	Excl.	d_h	MAE	d_h	MAE	d_h	MAE	d_h	MAE	d_h	MAE
C_5	None	6	$0.1717_{\pm 0.0144}$	6	$0.1658_{\pm 0.0046}$	3	$0.1779_{\pm 0.0052}$	3	$0.1748_{\pm 0.0034}$	3	$0.1777_{\pm 0.0114}$
C_6	None	20	$0.1670_{\pm 0.0060}$	20	$0.1602_{\pm 0.0073}$	10	$0.1565_{\pm 0.0060}$	10	$0.1775_{\pm 0.0087}$	10	$0.1918_{\pm 0.0091}$
C_6	1-WL	12	$0.1729_{\pm 0.0042}$	12	$0.1764_{\pm 0.0125}$	6	$0.1859_{\pm 0.0094}$	6	$0.1931_{\pm 0.0046}$	6	$0.1814_{\pm 0.0045}$
C_7	None	24	$0.1203_{\pm 0.0029}$	24	$0.1243_{\pm 0.0076}$	12	$0.1170_{\pm 0.0097}$	12	$0.1235_{\pm 0.0024}$	12	$0.1253_{\pm 0.0045}$
C_7		22	$0.1267_{\pm 0.0062}$	22	$0.1195_{\pm 0.0046}$	11	$0.1212_{\pm 0.0054}$	11	$0.1238_{\pm 0.0027}$	11	$0.1162_{\pm 0.0073}$
C_8	None	70	$0.1560_{\pm 0.0086}$	70	$0.1453_{\pm 0.0047}$	35	$0.1538_{\pm 0.0074}$	35	$0.1614_{\pm 0.0109}$	35	$0.1628_{\pm 0.0052}$
C_8		68	$0.1542_{\pm 0.0041}$	68	$0.1429_{\pm 0.0058}$	34	$0.1478_{\pm 0.0124}$	34	$0.1765_{\pm 0.0065}$	34	$0.1758_{\pm 0.0091}$
C_8	1-WL	56	$0.1517_{\pm 0.0096}$	56	$0.1489_{\pm 0.0094}$	28	$0.1464_{\pm 0.0099}$	28	$0.1542_{\pm 0.0063}$	28	$0.1578_{\pm 0.0070}$
$C_6 \cup C_7$	None	44	$0.1117_{\pm 0.0095}$	44	$0.1219_{\pm 0.0144}$	22	$0.1138_{\pm 0.0062}$	22	$0.1260_{\pm 0.0082}$	22	$0.1559_{\pm 0.0294}$
$C_6 \cup C_7$	1-WL	36	$0.1239_{\pm 0.0093}$	36	$0.1140_{\pm 0.0026}$	18	$0.1181_{\pm 0.0087}$	18	$0.1193_{\pm 0.0077}$	18	$0.1607_{\pm 0.0302}$
$C_7 \cup C_8$	None	94	$0.1125_{\pm 0.0034}$	94	$0.1091_{\pm 0.0029}$	47	$0.1073_{\pm 0.0043}$	47	$0.1532_{\pm 0.0056}$	47	$0.1411_{\pm 0.0076}$
$C_7 \cup C_8$	1-WL	80	$0.1108_{\pm 0.0024}$	80	$0.1136_{\pm 0.0063}$	40	$0.1015_{\pm 0.0010}$	40	$0.1436_{\pm 0.0074}$	40	$0.1316_{\pm 0.0061}$

Results. Examining the results in Table 3.3, we see that $\text{AlphaHom}^{\text{MLP}}$ generally produces better results compared to $\text{AlphaHom}^{\text{Lin}}$, especially when using the new pre-concatenated vector in $\text{AlphaHom}_{+}^{\text{MLP}}$. However, the results for $\text{MultHom}^{\text{MLP}}$ do not differ much from those for $\text{MultHom}^{\text{Lin}}$. The performance difference between the AlphaHom and MultHom models is also interesting because in some respects, both formulations are given the same α and Hom information, but the way in which this information is processed is different.

One possible explanation for this discrepancy is that multiplying the α coefficients with the raw Hom counts directly could be distorting the signal in some way that is unfavorable for the model. We will thus explore this interplay between α coefficients and raw Hom counts further in the next experiment.

Experiment 4: Encoding Different Alpha Coefficient Splits. Recall from Section 2.1.1 that the alpha coefficients roughly take the form:

$$\alpha_F \approx \frac{f(\text{lattice})}{\mathsf{Aut}(F)}$$

where $f(\text{lattice})$ is a function determined by the lattice structure of the basis pattern F, and $\mathsf{Aut}(F)$ denotes the number of automorphisms of F. In order to investigate the respective roles of these two components, we decompose the alpha coefficients and apply them to the homomorphism features Hom in different ways. Note that the coefficients have alternating signs, and this is reflected in the $f(\text{lattice})$ values. The resulting encoding schemes can be summarized as follows:

- AutHom$^{\text{MLP}}$: $h^{(0)}$ + MLP[$\mathsf{Hom} * \frac{1}{\mathsf{Aut}(F)} + f(\text{lattice})$]
- LatHom$^{\text{MLP}}$: $h^{(0)}$ + MLP[$\mathsf{Hom} * f(\text{lattice}) + \frac{1}{\mathsf{Aut}(F)}$]

We also provide the results from the following model as reference:

- AlphaHom$^{\text{MLP}}$: $h^{(0)}$ + MLP[Hom] + MLP[α] (from Experiment 3 (Table 3.3))

Table 3.4: Results from encodings using different α coefficient splits.

Basis			**AlphaHom$^{\text{MLP}}$**	**AutHom$^{\text{MLP}}$**	**LatHom$^{\text{MLP}}$**
Pattern	Excl.	d_h	MAE	MAE	MAE
C_5	None	6	0.1658±0.0046	0.1778±0.0056	0.1694±0.0038
C_6	None	20	0.1602±0.0073	0.1814±0.0079	0.1657±0.0054
C_6	1-WL	12	0.1764±0.0125	0.1797±0.0073	0.1626±0.0061
C_7	None	24	0.1243±0.0076	0.1688±0.0069	0.1250±0.0083
C_7		22	0.1195±0.0046	0.1736±0.0050	0.1207±0.0063
C_8	None	70	0.1453±0.0047	0.1782±0.0121	0.1483±0.0052
C_8		68	0.1429±0.0058	0.1792±0.0111	0.1512±0.0074
C_8	1-WL	56	0.1489±0.0094	0.1736±0.0076	0.1523±0.0072
$C_6 \cup C_7$	None	44	0.1219±0.0144	0.1522±0.0170	0.1145±0.0070
$C_6 \cup C_7$	1-WL	36	0.1140±0.0026	0.1691±0.0101	0.1214±0.0055
$C_7 \cup C_8$	None	94	0.1091±0.0029	0.1765±0.0110	0.1279±0.0173
$C_7 \cup C_8$	1-WL	80	0.1136±0.0063	0.1633±0.0112	0.1261±0.0137

Results. We see in Table 3.4 that $\text{LatHom}^{\text{MLP}}$ performs better than $\text{AutHom}^{\text{MLP}}$. This is expected since pre-multiplying by $f(\text{lattice})$ essentially stretches the signal provided by the Hom values, whereas pre-multiplying by $\frac{1}{\mathsf{Aut}(F)}$ squeezes the signal. The $\text{LatHom}^{\text{MLP}}$ results are also roughly comparable to the $\text{AlphaHom}^{\text{MLP}}$ results in Table 3.3. Overall, these results suggest that how the homomorphism count values are scaled or normalized plays an important role in empirical performance. As such, we revisit the positional encoding scheme introduced in Experiment 1 to identify if there is a smarter way to normalize the homomorphism count values.

Experiment 5: Evalutaing Different Positional Encoding Dimensions. Recall that the positional encoding experiments from Experiment 1 reduced the expanded positional encoding vectors down to a single dimension using an additional Linear encoder. In this experiment, we will not perform this reduction, but instead directly pass the generated positional encoding vectors into a 2-layer MLP. This in turn should help preserve more of the intricate feature signals that are captured by the positional encoding (i.e. the count differences both within graphs and across graphs in the dataset). The resulting encoding scheme is as follows:

- $\text{PosHom}^{\text{MLP}}$: $h^{(0)} + \text{MLP}[\text{PE}[\alpha * \mathsf{Hom}]]$

We also provide the results from the following model as reference:

- $\text{PosHom}^{\text{Lin}}$: $h^{(0)} + \text{Linear}[\text{Linear}[\text{PE}[\alpha * \mathsf{Hom}]]]$ (from Experiment 1 (Table 3.1))

Table 3.5: Results from $\text{PosHom}^{\text{MLP}}$ using different positional encoding dimension sizes.

Basis			**$\text{PosHom}^{\text{Lin}}$**	**$\text{PosHom}^{\text{MLP}}_{\text{PE=2}}$**	**$\text{PosHom}^{\text{MLP}}_{\text{PE=4}}$**	**$\text{PosHom}^{\text{MLP}}_{\text{PE=6}}$**	**$\text{PosHom}^{\text{MLP}}_{\text{PE=10}}$**
Pattern	Excl.	d_h	MAE	MAE	MAE	MAE	MAE
C_5	None	3	0.1848±0.0074	0.2717±0.0290	0.2393±0.0265	0.2235±0.0113	0.2011±0.0053
C_6	None	10	0.1781±0.0070	0.1897±0.0072	0.1753±0.0082	0.1703±0.0070	0.1663±0.0100
C_6	1-WL	6	0.1653±0.0029	0.2021±0.0060	0.1873±0.0080	0.1894±0.0102	0.1726±0.0072
C_7	None	12	0.1755±0.0051	0.1318±0.0029	0.1265±0.0062	0.1205±0.0061	0.1186±0.0048
C_7		11	0.1309±0.0193	0.1360±0.0027	0.1249±0.0043	0.1209±0.0068	0.1203±0.0046
C_8	None	35	0.1639±0.0075	0.1640±0.0083	0.1712±0.0073	0.1684±0.0033	0.1784±0.0052
C_8		34	0.1688±0.0056	0.1700±0.0095	0.1660±0.0077	0.1651±0.0061	0.1785±0.0162
C_8	1-WL	28	0.1533±0.0071	0.1723±0.0075	0.1751±0.0068	0.1681±0.0050	0.1630±0.0030
$C_6 \cup C_7$	None	22	0.1264±0.0034	0.1206±0.0067	0.1186±0.0030	0.1219±0.0066	0.1268±0.0103
$C_6 \cup C_7$	1-WL	18	0.1201±0.0102	0.1130±0.0044	0.1090±0.0038	0.1120±0.0080	0.1229±0.0067
$C_7 \cup C_8$	None	47	0.1423±0.0034	0.1103±0.0042	0.1137±0.0051	0.1253±0.0041	0.1468±0.0062
$C_7 \cup C_8$	1-WL	40	0.1498±0.0085	0.1097±0.0037	0.1120±0.0027	0.1155±0.0031	0.1372±0.0056

Results. From the results in Table 3.5, we see that using a smaller positional encoding dimension yields better performance for more complex basis sets, likely because the model is unable to properly process the higher dimensional encodings. However, compared to the $\text{PosHom}^{\text{Lin}}$ results from Table 3.1, $\text{PosHom}^{\text{MLP}}$ still consistently performs better overall. The behavior of the pattern exclusion results also remains roughly in line with theoretical expectations.

The motivation for using positional encodings is because unlike subgraph counts, homomorphism counts can grow very large, particularly for large graphs. This leads to unbounded feature values, which the model may interpret as noise. By applying a positional encoding scheme, we can normalize the Hom features in a principled manner. As we show in Section 3.4.4, this normalization yields tangible benefits in practical, real-world settings.

Experiment 6: Only Encoding Homomorphism Basis Counts. So far, all previous experiments have incorporated some notion of α coefficients as input features for the model. Now, we will conduct experiments with only the Hom count features and no α coefficients, which aligns closely with the approach taken by Barceló et al. [27] (although note that they did not have access to α coefficient values). We experiment with two different versions of this: one that uses a positional encoding, and one that does not. These models can be summarized as follows:

- OnlyHom: $h^{(0)}$ + MLP[Hom]
- $\text{OnlyHom}_{\text{PE}}$: $h^{(0)}$ + MLP[PE[Hom]]

Results. From Table 3.6, we see that using Hom features without incorporating α coefficients actually performs quite well. Although the results are slightly worse than the best-performing $\text{AlphaHom}_{+}^{\text{MLP}}$ model above (from Experiment 3), they are still better than many of the other variations tried. When considering the $\text{OnlyHom}_{\text{PE}}$ results, we again see that lower positional encoding dimensions tend to perform better.

Given these results, we will continue to exclude α coefficients for the rest of our experiments. Not only does this better align with the existing literature, but it also helps us avoid the sensitivity of encoding α coefficients, which we found to be quite inconsistent. Furthermore, we can treat this approach as a reliable lower bound for the performance of our method.

Table 3.6: Results when only encoding Hom features, without incorporating α coefficients.

Basis			OnlyHom	$\textbf{OnlyHom}_{\textbf{PE=2}}$	$\textbf{OnlyHom}_{\textbf{PE=4}}$
Pattern	Excl.	d_h	MAE	MAE	MAE
C_5	None	3	$0.1713_{\pm 0.0016}$	$0.1650_{\pm 0.0018}$	$0.1634_{\pm 0.0057}$
C_6	None	10	$0.1786_{\pm 0.0106}$	$0.1616_{\pm 0.0067}$	$0.1545_{\pm 0.0087}$
C_6	1-WL	6	$0.1782_{\pm 0.0108}$	$0.1626_{\pm 0.0057}$	$0.1578_{\pm 0.0044}$
C_7	None	12	$0.1249_{\pm 0.0051}$	$0.1176_{\pm 0.0046}$	$0.1250_{\pm 0.0031}$
C_7	C_5+edge	11	$0.1248_{\pm 0.0069}$	$0.1144_{\pm 0.0028}$	$0.1235_{\pm 0.0036}$
C_8	None	35	$0.1623_{\pm 0.0100}$	$0.1623_{\pm 0.0106}$	$0.1744_{\pm 0.0057}$
C_8	C_6+edge	34	$0.1597_{\pm 0.0147}$	$0.1659_{\pm 0.0057}$	$0.1672_{\pm 0.0082}$
C_8	1-WL	28	$0.1494_{\pm 0.0102}$	$0.1651_{\pm 0.0050}$	$0.1713_{\pm 0.0057}$
$C_6 \cup C_7$	None	22	$0.1199_{\pm 0.0072}$	$0.1113_{\pm 0.0085}$	$0.1095_{\pm 0.0026}$
$C_6 \cup C_7$	1-WL	18	$0.1182_{\pm 0.0040}$	$0.1117_{\pm 0.0057}$	$0.1157_{\pm 0.0123}$
$C_7 \cup C_8$	None	47	$0.1297_{\pm 0.0057}$	$0.1157_{\pm 0.0057}$	$0.1295_{\pm 0.0035}$
$C_7 \cup C_8$	1-WL	40	$0.1192_{\pm 0.0116}$	$0.1167_{\pm 0.0056}$	$0.1240_{\pm 0.0031}$

Results Summary. From the suite of experiments conducted above, we have gained a number of valuable practical insights on how to properly capture local structural information from homomorphism basis features and realize the implications of our theoretical results. These insights can be summarized as follows:

1. In general, excluding α coefficients does not significantly harm performance. As such, our node feature formulation will focus primarily on Hom basis features moving forward (see Equation (3.7)).

2. Adding too much information (i.e. having a higher dimensional feature vector) may actually harm model performance for smaller GNN models (see Appendix A.1.1 for details).

3. Using a sine/cosine positional encoding can help normalize Hom values in a principled way if the raw counts are too large. This helps mitigate against having an unbounded feature space, while also preserving some measure of "distance" for the different Hom features, both within and between graphs.

4. When using a positional encoding, fewer dimensions usually performs slightly better, supporting the dimensionality concerns from Point 2. However, there is minimal differences between different positional encoding dimensions.

5. For GIN-E, excluding the 1-WL basis components generally does not seem to impact performance significantly if the model is parametrized correctly. This aligns with our theoretical analysis in Section 3.3.3, and we will therefore exclude 1-WL basis components from our GIN experiments moving forward.

6. Excluding specialty basis elements (e.g.  or) sometimes hurts performance, but these results are somewhat inconsistent. This indicates that the other basis features may still provide enough signal to make up for the exclusion of these singular elements. However, in general it is very difficult to perform specific feature engineering, which motivates our more general approach. We provide a more in-depth analysis of this in Section 3.4.2.

We now use these findings to conduct a more extensive set of experiments on a real-world graph learning tasks. Keeping with the theme of molecular applications, we will begin by evaluating our method on two different molecular property prediction datasets. Then, we will move on to other graph learning tasks, such as link prediction, and finish with a targeted expressivity evaluation.

3.4.2 Graph Regression Experiments on ZINC

ZINC Dataset Setup. Recall that ZINC [33, 41] is a graph regression task for predicting the constrained solubility of molecules. For the following experiments, we follow the experimental guidelines from Dwivedi et al. [33], and continue to use a subset of the dataset that contains 12,000 graphs (split into 10,000 for training, 1,000 for validation, and 1,000 for testing). Note that Dwivedi et al. [33] do not use edge features when performing model benchmarking, so for comparability sake, we adhere to their protocol and omit edge features from our experiments in this section.

Selection of Graph Motif Parameters. We select cycles as our patterns of interest because it is well established that rings are an important substructure in molecules. We use cycles up to length 8 for our experiments in Table 3.7 because Bouritsas et al. [26] found that it yielded the best performance for their GSN-v model on the ZINC dataset. Note that we take the homomorphism counts for $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ since this contains all the component elements that make up the bases for $C_3 - C_8$. Additionally, as mentioned in Section 3.3.2, we can use these Spasm and $\mathsf{Spasm}^{\circledast}$ counts to quickly calculate graph-level or node-level subgraph counts, respectively. Therefore, we also include these subgraph counts as additional features since this information directly follows from computing the homomorphism

counts of the entire basis of interest. In each model, the count features are encoded using a 2-layer MLP and concatenated to the initial node feature embeddings.

This 2-layer MLP that we use to encode the count features differs from the approach taken by Barceló et al. [27], who use the z-score of the log-normalized count values. However, we found that this normalization method actually degraded the performance of using Sub counts. Therefore, we decide to pass the additional count features into an MLP, which would yield more fair and comparable results.

Model Selection and Training. For ZINC, we select GAT [21], GCN [19], GIN [23], and BasePlanE [85] as our baseline models. This is because GAT, GCN, and GIN are all standard architectures, and we want to evaluate how effective the homomorphism basis counts are at elevating the performance and expressivity of these models. We select BasePlanE because it is a recent strong architecture that is maximally expressive over planar graphs (all graphs in ZINC are planar), and it has been shown to perform very well on several molecular benchmarks. Note that our reported results for GIN are slightly stronger than those presented in the original benchmark by Dwivedi et al. [33] because we use a more standard implementation of GIN from the PyTorch Geometric library.

In terms of training, we do not perform any additional model tuning and use the same model hyperparameters for GAT, GCN, and GIN as reported by Dwivedi et al. [33], and the same hyperparameters for BasePlanE as reported by Dimitrov et al. [85]. These hyperparameters are reported in Table A.1 in Appendix A.1.2. Each model is evaluated using the mean average error (MAE). All results are reported as the average of 5 runs at different seeds.

Experimental Setup. Our experiment compares the following configurations:

- Base: The baseline performance of the GAT, GCN, GIN, and BasePlanE model architectures.
- Sub: The baseline models enriched with *subgraph* counts of cycles $\{C_3, ..., C_8\}$.
- Hom: The baseline models enriched with *homomorphism* counts of cycles $\{C_3, ..., C_8\}$.
- Spasm: The baseline models enriched with homomorphism counts of $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$[3].
- $\mathsf{Spasm}^{\circledast}$: The baseline models enriched with homomorphism counts of $\mathsf{Spasm}^{\circledast}(C_7) \cup \mathsf{Spasm}^{\circledast}(C_8)$.

[3]Note that $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ also contain the basis graphs for C_3, C_4, C_5, and C_6 (recall Proposition 3.3.8).

In all cases, the homomorphism (or subgraph) count features are directly concatenated to the original node features (i.e. atom type) for each node. This makes Sub and Hom correspond directly to GSN [26] and LGP-GNN [27], respectively, which allows us to precisely compare our approach to these previous works. Following the analysis of Section 3.3.3, we exclude counts for the acyclic graphs in the basis for GIN and BasePlanE experiments but add them for GAT and GCN. This is because both GIN and BasePlanE theoretically reach 1-WL expressiveness, so we are able to demonstrate the fine-grained expressiveness of our approach by adding only the higher-order basis terms to the model.

Results. Our results are summarized in Table 3.7. In all models, adding the Spasm$^{\circledast}$ counts as additional node features yields the best performance. Despite ZINC being a graph-level task, we observe that precise node-level subgraph information is critical for top performance as Spasm$^{\circledast}$ outperforms Spasm. Additionally, in GAT and GCN, we clearly see the improvements in expressiveness from Theorem 3.3.1 in effect as Spasm already clearly outperforms Sub. This is further confirmed by the effectively equivalent performance of GAT+Spasm$^{\circledast}$, GIN+Spasm$^{\circledast}$, and GIN+Sub. That is, their performance is intuitively the performance of 1-WL together with node-level subgraph counts (up to C_8). Any additional expressiveness provided by Spasm$^{\circledast}$ is seemingly not relevant to the ZINC task. Since GIN is as expressive as 1-WL, little is gained, whereas GAT manages to match GIN through the increased expressiveness from the homomorphism basis. Moreover, the gap between Spasm and Spasm$^{\circledast}$ illustrates the discussion in Section 3.3.4 and in particular the effect of Spasm$^{\circledast}$ as predicted in Theorem 3.3.12.

Table 3.7: We report the mean absolute error (MAE) for graph regression on the ZINC dataset (without edge features). Using the Spasm and Spasm$^{\circledast}$ counts yields very substantial improvements in performance of every model compared to their respective baselines and other approaches using $C_3, ..., C_8$.

	GAT	GCN	GIN	BasePlanE
Base	0.457$\pm$0.004	0.417$\pm$0.007	0.294$\pm$0.012	0.124$\pm$0.004
Sub	0.210$\pm$0.006	0.206$\pm$0.006	0.147$\pm$0.006	0.108$\pm$0.002
Hom	0.269$\pm$0.033	0.254$\pm$0.017	0.208$\pm$0.025	0.106$\pm$0.004
Spasm	0.155$\pm$0.006	0.166$\pm$0.003	0.158$\pm$0.004	0.104$\pm$0.005
Spasm$^{\circledast}$	**0.147**$\pm$0.004	**0.165**$\pm$0.004	**0.146**$\pm$0.005	**0.100**$\pm$0.002

Table 3.8: Extended MAE results on ZINC graph regression (without edge features) with $C_3, ..., C_8$ patterns. Note that we are using a subset of the ZINC dataset that contains 12k graphs.

Model	Layers	Base	Sub	Hom	Spasm	Spasm$^{\circledast}$
GAT	4	0.457±0.004	0.210±0.006	0.269±0.033	0.155±0.006	**0.147**±0.004
	16	0.380±0.009	0.201±0.004	0.229±0.008	0.155±0.008	**0.152**±0.007
GCN	4	0.417±0.007	0.206±0.006	0.254±0.017	0.166±0.003	**0.165**±0.004
	16	0.282±0.007	0.198±0.003	0.235±0.005	0.167±0.007	**0.166**±0.005
GIN	4	0.294±0.012	0.147±0.006	0.208±0.025	0.158±0.004	**0.146**±0.005
	16	0.246±0.019	**0.143**±0.002	0.197±0.015	0.157±0.005	**0.143**±0.004
BasePlanE	3	0.124±0.004	0.108±0.002	0.106±0.004	0.104±0.005	**0.100**±0.002

Additional Results with Deeper Models. To expand on these results, we also evaluate the 16 layer versions of GCN, GAT, and GIN from Dwivedi et al. [33] in Table 3.8. Our results show that the expressiveness gains from adding Spasm and Spasm$^{\circledast}$ homomorphism counts for $C_3, ..., C_8$ hold for deeper GNNs that have more layers as well.

That said, for GAT and GIN, the best performing 16-layer model with Spasm$^{\circledast}$ seems be perform slightly worse than the best-performing 4-layer models with Spasm$^{\circledast}$. This may indicate that the additional layers are leading to over-smoothing of the anchored homomorphism features. For the 16-layer version of GIN, we again see that the GIN+Spasm$^{\circledast}$ and GIN+Sub models perform roughly the same.

BasePlanE Analysis. For BasePlanE, we observe that using the homomorphism basis instead of the parameter itself results in improved performance. This is insightful since even though BasePlanE can distinguish planar graphs (and all ZINC graphs are planar), distinguishing graphs is a weaker notion than capturing graph motif parameters, so injecting these parameters leads to empirical improvements even for stronger models specifically designed for planar graphs.

In fact, from Table 3.9, we can see that enriching BasePlanE with the homomorphism basis of these graph motif parameters yields extremely competitive results when compared to other leading GNN models for molecular property prediction on the ZINC dataset. These models include PNA [110], GSN [26], and CIN [87]. These results show the strong expressive power of using the homomorphism basis of graph motif parameters.

Table 3.9: Comparison of BasePlanE results with other leading models on the ZINC dataset. Note that this is on the 12K subset without using edge features.

Model	MAE
GCN	$0.278_{\pm 0.003}$
PNA	$0.320_{\pm 0.032}$
GSN	$0.140_{\pm 0.006}$
CIN	$0.115_{\pm 0.003}$
BasePlanE	$0.124_{\pm 0.004}$
BasePlanE + Sub	$0.108_{\pm 0.002}$
BasePlanE + Hom	$0.106_{\pm 0.004}$
BasePlanE + Spasm	$0.104_{\pm 0.005}$
BasePlanE + $\mathsf{Spasm}^{\circledast}$	$\mathbf{0.100}_{\pm 0.002}$

Using the Entire Basis. To study the effect of the homomorphism basis in more detail, we perform experiments where we inject increasingly more (in order of the number of vertices, lowest first) elements of the basis as features. To avoid the effect of the complex interplay of odd and even cycles on the benchmark, we use only $\mathsf{Spasm}(C_8)$. Figure 3.5 clearly shows that the performance of every model improves as more of the basis is included. Moreover, just as in Table 3.7, GAT and GIN converge in performance, again confirming that their expressiveness (as relevant to this task) matches with this additional information.

This plot is interesting, because we are able to see at which point the homomorphism basis features are able to surpass the subgraph count features. In particular, for the more expressive GIN model, only a fraction of the $\mathsf{Spasm}(C_8)$ homomorphism basis is enough to improve performance beyond that of GIN+$\mathsf{Sub}(C_8)$, which can be viewed of as an instatiation of GSN [26]. However, for weaker models such as GAT and GCN, more of the homomorphism basis must be included ($\sim 75\%$) before the performance is able to surpass that of GIN+$\mathsf{Sub}(C_8)$.

These results, combined with the ones above, show that using the homomorphism basis of a graph motif parameter is the key to achieving more expressive models. In particular, we see that the theoretical results from Section 3.3.1, which state that injecting the homomorphism basis of a graph motif parameter is more expressive than injecting the parameter itself, does indeed translate to practical applications, especially for molecular property prediction.

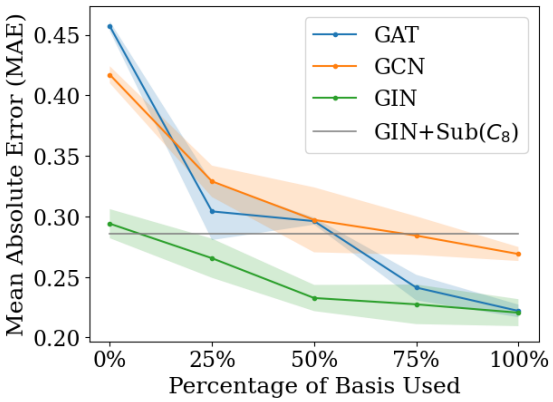


Figure 3.5: We report the effect of using incrementally more of $\mathsf{Spasm}(C_8)$ as node features for each model. The performance of GIN with $\mathsf{Sub}(C_8, \cdot)$ at node-level is also included for reference.

3.4.3 Graph Regression Experiments on QM9

Now that we have shown the empirical success of using the homomorphism basis in the ZINC molecular dataset, we will expand our analysis to other molecular property prediction datasets. In this next experiment, we will predict multiple molecular properties and explore other ways of encoding homomorphism counts.

QM9 Dataset Setup. In the QM9 graph regression benchmark, the objective is to predict 13 different molecular properties [40]. The entire dataset contains 130,831 graphs, which are split into 110,831 for training, 10,000 for validation, and 10,000 for testing. The original node features denote atom type as well as other chemical information about the atom. Similarly to ZINC, the edge features denote the type of bond between two nodes.

Selection of Graph Motif Parameters. Because there are multiple different target tasks for the QM9 dataset, we follow Proposition 3.3.13 and decide to take the homomorphism counts for all connected components that contain up to 5 vertices, $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$. In addition to this set, we also include $\mathsf{Hom}(C_6)$ because it is well established that 6-cycles are prevalent within molecules. Also, this ensures that $\mathsf{Spasm}(C_6)$ is contained within this set of features. Since we use R-GCN as our baseline model, we include all homomorphism counts for all components in this set, including the 1-WL ones. For each graph, all counts are encoded as graph-level features. Specifically, we concatenate the raw homomorphism counts at the very end after the message passing layers to the final feature representation vector for the graph before the final MLP decoder.

Model Selection and Training. We select R-GCN [111] as our base model and report the baseline results from Brockschmidt [112]. We then adapt the codebase from Abboud, Dimitrov and Ceylan [113] to perform our experiments. We choose R-GCN as our base model due to its comparatively stable behavior in the original benchmark [112]. Other than adding our homomorphism count features, we do no further hyperparameter tuning. Specific hyperparameter details are in Appendix A.1.3.

Experimental Setup. Contrasting Section 3.4.2, we inject only graph-level homomorphism counts rather than node-level counts. These homomorphism features are concatenanted directly to the model embeddings at the final layer before the MLP decoder is applied. This is to show the flexibility of our approach, since it does not matter if the features are incorporated at the node-level or graph-level to introduce the extra expressivity. Our results in Table 3.10 are reported as an average across 5 different runs. On average, the model training time for R-GCN on one property took roughly 5 hours to complete, which is significantly longer than the time it took to compute all of the homomorphism feature counts for the entire QM9 dataset (see Table 3.14). This further highlights the negligible compute cost to generate the homomorphism counts.

Table 3.10: We report the mean absolute error (MAE) for graph regression on QM9 for all 13 proprieties. Using the homomorphism counts yields significant improvements over the baseline model.

Property	R-GCN	+ $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5} \cup \{C_6\})$
mu	3.21±0.06	**2.29**±0.03
alpha	4.22±0.45	**1.77**±0.05
HOMO	1.45±0.01	**1.30**±0.03
LUMO	1.62±0.04	**1.41**±0.02
gap	2.42±0.14	**2.00**±0.04
R2	16.38±0.49	**10.29**±0.35
ZPVE	17.40±3.56	**3.03**±0.38
U0	7.82±0.80	**1.09**±0.18
U	8.24±1.25	**1.21**±0.17
H	9.05±1.21	**1.22**±0.14
G	7.00±1.51	**1.14**±0.13
Cv	3.93±0.48	**1.46**±0.08
Omega	1.02±0.05	**0.81**±0.02

Results. Table 3.10 shows that using the homomorphism counts for $\Omega^{\mathsf{con}}_{\leq 5}$ and C_6 significantly improves the performance of the base R-GCN model on all properties, even without additional model tuning. Moreover, we see that the addition of graph level homomorphism counts alone are sufficient for substantial performance improvements. This is particularly noteworthy as it simplifies integration with more complex models, where node-level injection of many features might affect hyperparameters significantly. To the best of our knowledge, this is also the first study demonstrating the potential of injecting graph-level homomorphism information to a GNN.

Additional Results with Fully-Connected Layers. Alon and Yahav [114] also presented a slightly modified version of R-GCN that uses a fully-adjacent (FA) layer at the very end of the model. They show that this helps counteract the signal loss due to over-squashing effects in a GNN. We benchmark our approach against this model configuration and find that using the homomorphism basis counts continue to improve the performance of the R-GCN+FA model (see Table 3.11).

We see that for all but one property, the standard R-GCN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5}, C_6)$ model outperforms the R-GCN + FA model. This suggests that for tasks where the molecular structure is important, the graph-level homomorphism counts are still important enough to provide the model with useful information, despite potential signal loss from over-squashing. The fact that we apply our global homomorphism features at the end may also help mitigate some of the information loss due to over-squashing, and further speaks to the flexibility of our approach.

From Table 3.11, it is clear that the best performing model is when we use both the FA layer and the homomorphism basis counts. This demonstrates that including homomorphism counts provides complementary information to the fully-adjacent layer, which helps boost total performance of an originally weak model. This is an interesting result because it shows how homomorphism basis features can be used as an easy drop-in component to lift the expressivity of several different architectures. Overall, these observations demonstrate the well-foundedness of our approach as it confirms the practical relevance of Section 3.3.5 and Proposition 3.3.13.

Furthermore, although we did not evaluate our method on the synthetic TreeNeighborsMatch benchmark from Alon and Yahav [114], homomorphism basis features may be effective for capturing the long-range information required to solve this task. By selecting graph motifs that correspond to long paths or deep trees, the homomorphism basis count features can capture information from distant regions of the graph. While this may not directly combat potential over-squashing, our results above show that adding these features at the end of the model can partially circumvent some of its effects and yield significant performance gains.

Table 3.11: We report the mean absolute error (MAE) for graph regression on QM9. Using the homomorphism counts yields significant improvements over the baseline models (both with and without FA). Best results are shown in red, second best in blue.

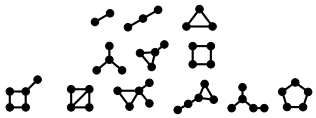
Property	R-GCN	+ $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5} \cup C_6)$	+ FA	+ FA/Hom
mu	3.21±0.06	2.29±0.03	2.91±0.07	2.15±0.02
alpha	4.22±0.45	1.77±0.05	2.14±0.08	1.70±0.05
HOMO	1.45±0.01	1.30±0.03	1.37±1.41	1.22±0.02
LUMO	1.62±0.04	1.41±0.02	1.41±0.01	1.27±0.00
gap	2.42±0.14	2.00±0.04	2.03±0.03	1.81±0.03
R2	16.38±0.49	10.29±0.35	13.55±0.50	9.99±0.33
ZPVE	17.40±3.56	3.03±0.38	5.81±0.61	2.86±0.31
U0	7.82±0.80	1.09±0.18	1.75±0.18	1.03±0.05
U	8.24±1.25	1.21±0.17	1.88±0.22	1.07±0.04
H	9.05±1.21	1.22±0.14	1.85±0.18	1.14±0.12
G	7.00±1.51	1.14±0.13	1.76±0.15	1.19±0.18
Cv	3.93±0.48	1.46±0.08	1.90±0.07	1.46±0.07
Omega	1.02±0.05	0.81±0.02	0.75±0.04	0.71±0.02

3.4.4 Link Prediction on COLLAB

So far, we have demonstrated multiple ways in which our approach is effective on graph-level regression tasks for small molecular graphs. In this experiment, we contrast this by performing a link prediction task on a graph where direct subgraph counting is intractable for most patterns. This also provides an example of how our approach extends beyond just the molecular domain.

COLLAB Dataset Setup. COLLAB is a collaboration network graph from Open Graph Benchmark that presents a link prediction task [115]. In total, the entire undirected graph contains over 235,000 nodes and 1,285,465 edges, but 23,367 of the edges are used for validation, and 294,466 edges are used for testing. Nodes represent scientists, and edges represent collaborations between two scientists. The nodes also contain 128-dimensional node features that correspond to the average word embeddings for each scientist's papers. Edge features include the year and the edge weight, representing the number of co-authored papers published that year. Here, the task is to predict if there exists an edge (i.e. a collabroation) between two given nodes (i.e. scientists) in the network.

Table 3.12: Hits@50 results for link prediction over the COLLAB dataset.

Add. Basis	Γ	GCN	GraphSAGE	GAT
	—	46.13%±2.10	48.85%±0.43	48.27%±1.05
	K_3	49.41%±0.42	49.55%±0.66	49.43%±0.73
	K_4	47.76%±0.53	49.29%±0.30	48.35%±0.85
	K_5	48.01%±0.87	49.61%±0.24	49.75%±0.16
	P_4	49.59%±0.23	50.01%±0.57	50.76%±0.51
	P_5	49.60%±0.29	49.50%±0.47	51.55%±0.94
	P_6	**50.35%**±0.21	**50.18%**±0.14	**51.62%**±0.66

Selection of Graph Motif Parameters. Traditional substructure-style approaches [27] have used cliques (i.e., K_3, K_4, K_5) as patterns of interest for the COLLAB dataset. The Spasm of a clique contains only itself, and the clique experiments that we conduct thus match the experiments conducted by Barceló et al. [27]. However, because COLLAB presents a link prediction task that may require multi-hop reasoning, paths could also be a useful pattern. Paths are interesting because counting the number of n-vertex paths in a graph cannot be done by a 1-WL model due to the cyclic components in the homomorphism basis. Also, standard subgraph counting techniques [99] are mostly intractable for graphs as large as COLLAB, whereas our method is practically feasible.

Model Selection and Training We select GCN, GAT, and GraphSAGE [77] as our default baseline models. For GCN and GraphSAGE, we use the exact same model and hyperparameters as the default example provided by Hu et al. [115]. For GAT, we construct a model that closely aligns with the overall framework of the GCN and GraphSAGE implementations. All results are reported as an average of 4 runs at different seeds, using the Hits@50 metric. Specific model details are outlined in Table A.2.

Experimental Setup. Following our experiments from Section 3.4.2, we again enrich the baseline models with the homomorphism basis counts of our parameter of interest at the node level. Due to the large graph size, we normalize our raw homomorphism counts using the sine and cosine positional encoding technique from Vaswani et al. [6]. These mirror the experiments conducted above in Section 3.4.1. We also experiment with positional encoding dimensions of size 2, 4, 6, and 8.

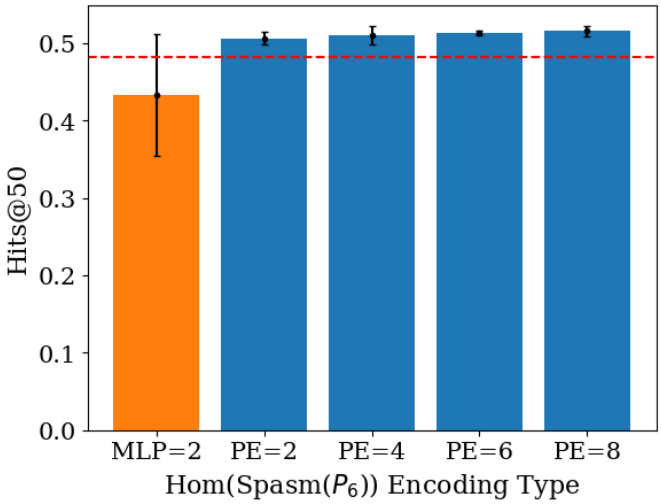


Figure 3.6: We report the effect of using different encoding methods and dimensions for the $\mathsf{Spasm}(P_6)$ homomorphism count features with GAT on COLLAB. The dashed red line indicates the performance of GAT without additional features. Using a positional encoding (PE) boosts performance of the baseline model, whereas a standard MLP encoder without positional encodings degrades performance.

Results. From Table 3.12 we see that our approach of injecting the basis homomorphism counts for paths outperforms the previously studied injection of clique counts in both models. This further demonstrates the targeted gain of expressiveness, irrespective of task, that can be realized through our approach. We note that while short paths are seemingly simple structures, actually counting them is not expressible in 1-WL as illustrated by the cyclic graphs in the basis (and thus also in the tested, weaker models). Despite the large graph size, we compute all necessary node-level homomorphisms counts in negligible time (see Section 3.4.6).

Ablation with Positional Encoding. To study the impact of normalizing homomorphim counts with a sinusoidal positional encoding [6], we perform an ablation with GAT on the COLLAB dataset. We compare the performance of using only an MLP encoder to using a positional encoder for the $\mathsf{Hom}(\mathsf{Spasm}(P_6))$ features. From Figure 3.6, we see that including counts directly via an MLP encoder actually degrades performance of the baseline model and introduces high variance. This is likely because homomorphism count values can grow extremely large in larger graphs, causing the model to treat the extra information as noise. Conversely, using a positional encoding allows us to normalize the raw homomorphism count values in a way that preserves the relative distances between different patterns across different graphs. However, adjusting the positional encoding dimension presented minimal effects in our ablation.

From our COLLAB experiments, we see that our approach is practically applicable to large-scale graphs as well as on different graph learning tasks (e.g. link prediction). Finally, we will round out our experimental section with results on a synthetic expressivity dataset to highlight the true expressive power of homomorphism basis counts.

3.4.5 Expressiveness Evaluation on BREC

In this experiment, we will focus on evaluating the expressivity of our proposed approach. Recall that our definition of expressivity pertains to how well a model is able to distinguish between two non-isomorphic graphs. This makes BREC an ideal dataset to pinpoint the exact expressivity gains and advantages of our homomorphism basis counts.

BREC Dataset Setup. BREC is a recent synthetic expressiveness dataset, where the task is to *distinguish* 400 pairs of non-isomorphic graphs [116]. The graphs span four different primary categories: Basic, Regular, Extension and CFI graphs. There are 60 pairs of Basic graphs, 140 pairs of Regular graphs, 100 pairs of Extension graphs, and 100 pairs of CFI graphs. The Regular graphs are further subdivided into 50 pairs of simple regular graphs, 50 pairs of strongly regular graphs, 20 pairs of 4-vertex condition graphs, and 20 pairs of distance regular graphs. None of the graphs in this dataset contain any initial node or edge features.

Selection of Graph Motif Parameters. Since this is a synthetic expressivity task, there is no specific pattern of interest that we want to capture. In this case, we again opt against choosing a particular pattern and instead inject the homomorphism counts of $\Omega^{\mathsf{con}}_{\leq 5}$ as node features. This approach is similar to the one used in Section 3.4.3 and again supports the claims made in Proposition 3.3.13 and Section 3.3.5.

Model Selection and Training. We select GIN and PPGN [78] as the two models to extend with node-level homomorphism counts. For GIN+$\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq k})$, we use the exact same model construction and hyperparameters as were used to benchmark GSN on this dataset. For PPGN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$ we normalize features in accordance with Maron et al. [78]. We also sightly tweak the hyperparameters of the model used in the BREC benchmark by changing the dimensions of the blocks from 32 to 48 in order to accommodate our extra features. Specific hyperparameters are provided in Appendix A.1.5.

Experimental Setup. We follow the evaluation protocol of Wang and Zhang [116] and report the best results from 10 seeds to get an upper bound on the expressiveness of each model. The effect of this protocol on our results is minimal. For the top performing PPGN+$\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq k})$ model, we only observe minimal variance over changing seeds. The number of solved CFI graph pairs varied between 22% and 25%, and we observed no change in the other categories.

Table 3.13: Full results for the BREC expressiveness experiment, comparing the performance of all models tested by Wang and Zhang [116]. The numbers in the parentheses are the total number of graph pairs to distinguish within that category. Num denotes the number of graph pairs solved, and Acc denotes the percentage solved.

	Basic (60)		Reg. (140)		Ext. (100)		CFI (100)		Total (400)	
Model	Num	Acc	Num	Acc	Num	Acc	Num	Acc	Num	Acc
2-WL	60	100%	50	35.7%	100	100%	60	60%	270	67.5%
SPD-WL	16	26.7%	14	11.7%	41	41%	12	12%	83	20.8%
S_3	52	86.7%	48	34.3%	5	5%	0	0%	105	26.2%
S_4	60	100%	99	70.7%	84	84%	0	0%	243	60.8%
N_1	60	100%	99	85%	93	93%	0	0%	252	63%
N_2	60	100%	138	98.6%	100	100%	0	0%	298	74.5%
M_1	60	100%	50	35.7%	100	100%	41	41%	251	62.8%
NGNN	59	98.3%	48	34.3%	59	59%	0	0%	166	41.5%
DE+NGNN	60	100%	50	35.7%	100	100%	21	21%	231	57.8%
DS-GNN	58	96.7%	48	34.3%	100	100%	16	16%	222	55.5%
DSS-GNN	58	96.7%	48	34.3%	100	100%	15	15%	221	55.2%
SUN	60	100%	50	35.7%	100	100%	13	13%	223	55.8%
SSWL_P	60	100%	50	35.7%	100	100%	38	38%	248	62%
GNN-AK	60	100%	50	35.7%	97	97%	15	15%	222	55.5%
KP-GNN	60	100%	106	75.7%	98	98%	11	11%	275	68.8%
I^2-GNN	60	100%	100	71.4%	100	100%	21	21%	281	70.2%
PPGN	60	100%	50	35.7%	100	100%	23	23%	233	58.2%
δ-k-LGNN	60	100%	50	35.7%	100	100%	6	6%	216	54%
KC-SetGNN	60	100%	50	35.7%	100	100%	1	1%	211	52.8%
GSN	60	100%	99	70.7%	95	95%	0	0%	254	63.5%
DropGNN	52	86.7%	41	29.3%	82	82%	2	2%	177	44.2%
OSAN	56	93.3%	8	5.7%	79	79%	5	5%	148	37%
Graphormer	16	26.7%	12	10%	41	41%	10	10%	79	19.8%
GIN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$	60	100%	120	85.7%	96	96%	0	0%	276	69%
PPGN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$	60	100%	120	85.7%	100	100%	25	25%	305	76.25%

Results. We present our results on BREC in the full context of the benchmark provided by Wang and Zhang [116] in Table 3.13. In particular, we compare our results to the previous state of the art I^2-GNN and closely related models, such as PPGN and GSN. We achieve state of the art results using PPGN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$. Notably, in the category of regular graphs, where PPGN was initially lacking, the inclusion of our approach enables the model to distinguish over twice as many graphs. This experimentally illustrates our analysis in Section 3.3.3 in the context of higher-order models: $\Omega^{\mathsf{con}}_{\leq 5}$ contains graphs with treewidth greater than 2 (in particular 4 graphs containing a 4-clique). These homomorphism counts thus improve the 2-WL[4] expressiveness of PPGN. GIN + $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$ also performs well by ranking 4th overall, outperforming many higher-order models.

We note that the 2-WL row in Table 3.13 is labeled as "3-WL" in the original benchmark [116] due to a mismatch in which style of k-WL is considered in the paper. Since we use k-WL to refer to the *folklore* version of k-WL, we have adapted the label from their table accordingly. Furthermore, although PPGN is theoretically as expressive as 2-WL, note that it performs slightly worse. This is because the large radii in CFI graphs requires more layers of GNNs (i.e. more WL iterations). However, using deeper networks can cause over-smoothing, which would explain the gap between the theoretical expectations and the actual empirical performance [116]. This again speaks to the practical benefits of our approach, since we are able to recover some of the signal that may have been lost from the original PPGN model.

In summary, these results show that our framework for injecting homomorphism basis counts leads to more expressive architectures, even for higher-order k-WL models, such as PPGN. This again speaks to the flexibility and power of homomorphism counts for capturing local structure (and to some extent, global structure) in graph learning.

3.4.6 Practical Runtime for Counting Homomorphisms

Recall our complexity analysis from Section 3.3.2, where we stated that using the homomorphism basis is the most efficient way of computing a graph motif parameter. Here, we will see the practical implications of this. We report the time required to compute all the homomorphism counts that were used for each experiment in Table 3.14. We also provide additional details regarding the size of the graphs in each dataset and the amount of compute used to generate these counts. From Table 3.14, we can clearly see that the time required to generate these homomorphism counts

[4]Recall that we always refer to the folklore version of k-WL.

Table 3.14: Time to compute all the homomorphism counts used in each experiment in Section 3.4 reported as elapsed real time (wall time). n_G denotes the number of graphs in the dataset, and $\overline{|V|}$ and $\overline{|E|}$ indicate the average number of nodes and edges respectively. Note that the time reported is normalized per graph.

Dataset	n_G	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	Basis	Basis size	Time
ZINC	12,000	∼23.2	∼49.8	$\mathsf{Spasm}(C_7)$	12	0.007s
				$\mathsf{Spasm}(C_8)$	35	0.024s
				$\mathsf{Spasm}^{\circledast}(C_7, C_8)$	118	0.096s
QM9	130,831	∼18.0	∼37.3	$\Omega^{\mathsf{con}}_{\leq 5}$	30	0.019s
				$\bar{C}_6$	1	0.001s
COLLAB	1	235,868	967,632	K_3	1	3m48s
				K_4	1	5m30s
				K_5	1	6m29s
				$\mathsf{Spasm}(P_4)$	4	4m16s
				$\mathsf{Spasm}(P_5)$	8	39m23s
				$\mathsf{Spasm}(P_6)$	15	3h6m
BREC	51,200	∼34.7	∼286.9	$\Omega^{\mathsf{con}}_{\leq 5}$	30	0.063

is almost negligible, especially since this counting step only needs to be performed once for a given parameter and dataset. In other words, once the counts have been obtained, they can be reused and added to any existing model or architecture.

The computational efficiency of performing this homomorphism counting is especially apparent for QM9, where we are able to generate all $\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq 5})$ counts in less than 0.02 seconds per graph. For comparison, training a single run of R-GCN on QM9 to predict a single target property took several hours. For ogbl-COLLAB, we only perform counting on the training graph, which is a subset of the entire dataset. This is because for link prediction, a subset of the edges are removed from training and used for validation and testing. Due to the memory constraints of our compute resources, we use a slightly modified counting method for counting the number of 5-clique homomorphisms in COLLAB. Finally, we note that we do not perform counting for $\mathsf{Hom}(K_5)$ on the 4-vertex graphs in BREC because the graphs are constructed in a way that makes counting them infeasible.

From these reported times, we can see that computing homomorphism basis counts is indeed extremely fast, which makes our approach feasible for many widespread applications. It is also well known that homomorphism counting is closely related to database theory, where homomorphism counts can be formulated as SQL JOIN queries [117]. This unlocks the possibility of integrating more complex

graph motif parameters from relational or tabular data into GNNs. It also allows for the computation of homomorphism basis counts on larger datasets, which may currently be infeasible with other tools at hand.

3.5 Summary and Outlook

In this chapter, we have proposed the injection of the homomorphism basis of graph motif parameters into GNNs as a flexible framework for increasing expressiveness in a targeted and efficient manner. We experimentally validate that this increase in expressiveness is reflected in significant performance gains in real-world graph representation learning tasks. This is particularly beneficial for molecular property prediction, where being able to capture important local structural information is essential to the problem at hand.

More specifically, in terms of theoretical contributions, we have proven that enriching GNNs with the homomorphism basis of a graph motif parameter is more expressive than enriching a GNN with the parameter itself (Theorem 3.3.1). For subgraph counting, this approach allows GNNs to capture the subgraph counts of patterns in the basis of the target graph motif parameter as well (Proposition 3.3.8), at no additional computational cost. We also prove that it is never necessary to consider homomorphism counts from disconnected graphs (Proposition 3.3.10). This result holds for more advanced, higher-order GNNs as well (Theorem 3.3.5). Furthermore, we have proven that this applies at both the graph-level and node-level, where we have introduced a method for obtaining local orbital-style subgraph counts (Theorem 3.3.12).

In terms of empirical contributions, we show that we are able to realize the implications of our theoretical results on a series of different real-world tasks. These include notable performance gains on molecular graph-level tasks for the ZINC and QM9 datasets (Section 3.4.2 and Section 3.4.3), an edge-level task on the COLLAB dataset (Section 3.4.4), and a specifically curated graph expressivity task on the BREC dataset (Section 3.4.5).

This chapter lays the foundation for a wide range of further applications for both node-level and graph-level homomorphism counts. Although our experiments have primarily focused on standard architectures, our results motivate further analysis on combining homomorphism basis counts with more complex models, which we will touch upon in the next chapter. Furthermore, the use of homomorphism bases induced by logical formulas (Section 3.3.5) has been beyond the scope of this thesis but presents numerous interesting possibilities.

Another open direction of research would be to use the homomorphism basis of typed motifs that can represent structural information and their associated features simulaneously. In this case, the homomorphism basis would differ based on the motif of interest, but our framework can easily be extended to incorpate node and edge types (e.g. atom types and bond types in molecular graphs) and also directed edges for heterogeneous knowledge graphs. An example of this is provided in Figure A.6 of Appendix A.

The broad applicability of our approach also raises a number of technical questions that require more detailed analysis. The question of how to best encode these homomorphism count features to realize their full theoretical potential still remains an open question. Additionally, the choice of basis to inject depends on the desired expressivity, and further work is needed to explore the effects of different bases in common GNN tasks. Although we have proposed using the homomorphism counts of all connected components up to a certain size as a general, catch-all set, it is possible that for some tasks, a more specific basis may be more effective. Therefore, it would be interesting to leverage techniques such as reinforcement learning to learn the best set of graph motif parameters for a given task and dataset. We can also inspect the model weights (or attention weights if applicable) to see which specific homomorphism count features are most important for the model's performance. This could provide insights into the underlying graph structure of the data, and also surface meaningful anchors within the graph as well.

Overall, graph motif parameters and their homomorphism bases present a strong theoretical framework for increasing the expressivity of GNN models in a clear and principled way. This approach comes with strong theoretical guarantees, which can also be realized in real-world applications, such as molecular property prediction. This chapter therefore highlights the importance of informed local structural representations and the benefits of having a more general, unified methodology.

4

Motif Structural Encodings for Graph Learning

In the previous chapter, we showed that using the homomorphism basis of graph motif parameters is an effective and principled way of improving local structural awareness and expressivity in GNNs. Now, we will show that homomorphism counts can also be used in a more general approach to construct powerful graph structural encodings for graph transformers and other models that have inherently weak graph inductive biases, such as MLPs. Our new structural encoding method, Motif Structural Encoding (MoSE), performs especially well on molecular graphs, which highlights its ability to capture complex local structure information. This chapter contains both theoretical results pertaining to the expressivity of MoSE and other popular structural encodings, as well as experimental results on molecular property prediction and other domains.

The content of this chapter is largely based off of the work published in Bao et al. [2], with the theoretical results in Section 4.3 completed in collaboration with Matthias Lanzinger and Linus Bao, and the ZINC experimental results presented in Section 4.4.2 completed in collaboration with Linus Bao.

4.1 Introduction

Graph Neural Networks (GNNs) have been the prominent approach to graph machine learning in the last decade. Most conventional GNNs fall under the framework of Message Passing Neural Networks (MPNNs), which incorporate graph structure – the so-called *graph inductive bias* – in the learning and inference process

by iteratively computing node-level representations using "messages" that are passed from neighboring nodes [22]. More recently, Transformer architectures [6] have been applied to the graph domain and achieved impressive empirical performance [29, 30], especially in molecular property prediction.

While MPNNs operate by exchanging messages between adjacent nodes in a graph, Transformers can be seen as a special type of GNN that operates on complete graphs. This helps alleviate some of the key limitations of MPNNs, such as over-squashing, over-smoothing, and the difficulty of capturing long-range interactions [118]. On the one hand, this allows for direct communication between all node pairs in a graph, regardless of whether there exists an edge between two nodes in the original input graph. On the other hand, since the node adjacency information is omitted, the Transformer lacks any "built-in" graph inductive bias. Instead, the underlying graph structure is usually provided by combining Transformer layers with local message-passing layers [29, 119, 120] or by incorporating additional pre-computed features that encode the topological context of each node. These additional features are referred to as *positional or structural encodings*. Common encodings include Laplacian positional encoding (LapPE) [33] and random-walk structural encoding (RWSE) [32]. The quality of these encodings are a key ingredient in the success of Transformers on graphs [29, 32].

Motivation. While empirical studies have observed the impact of structural or positional encodings on model performance [29, 32], our theoretical understanding of the expressive power of different encodings remains limited. This represents an important gap in the literature, especially since the expressive power of Transformer-based architectures heavily rely on the specific choice of the structural or positional encoding [121]. Furthermore, precise theoretical results that quantify the strength of an encoding's graph inductive bias, known as its *expressive power*, remain open-ended.

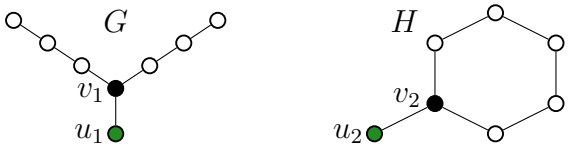
We begin by providing a short exposition on RWSE, which empirically has been shown to perform well on molecular benchmarks [29].

How Expressive is RWSE? The expressive power of MPNNs is upper bounded by the *1-dimensional Weisfeiler Leman graph isomorphism test (1-WL)* [23, 25]. This means that the node invariants computed by MPNNs are at most as powerful as the node invariants computed by 1-WL. As a result, an MPNN can distinguish two nodes *only if* 1-WL can distinguish them. Some MPNN architectures, such as Graph Isomorphism Networks (GIN) [23], can match this expressiveness bound

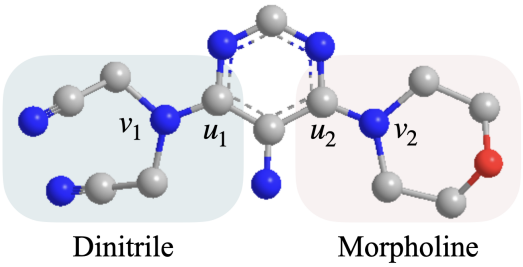
and distinguish any pair of nodes that can be distinguished by 1-WL. We relate the expressive power of RWSE to that of the Weisfeiler Leman hierarchy by proving that node invariants given by RWSE are strictly weaker than node invariants computed by 2-WL (Proposition 4.3.4).[1] Additionally, RWSE node invariants are incomparable to node invariants computed by 1-WL (Proposition 4.3.7). In fact, there are simple node pairs which can be distinguished by 1-WL but not by RWSE (and vice versa).

We now illustrate a serious limitation of RWSE in its power to distinguish nodes.

Example 4.1.1. Consider the nodes from the two non-isomorphic graphs G and H.



Observe that the nodes u_1 and u_2 can be distinguished by 1-WL. Interestingly, however, they are indistinguishable to RWSE for *all* lengths ℓ of the considered random walk. This observation also applies to the nodes v_1 and v_2. Moreover, this limitation is not merely of theoretical interest, as it readily applies to real-world molecules, where the use of RWSE is prominent. Below, we show a molecule from the ZINC dataset, with Carbon, Nitrogen, and Oxygen atoms.



The nodes u_1 and u_2 are indistinguishable by RWSE and so are v_1 and v_2, although they can be distinguished by MPNNs. In practical terms, this implies that RWSE cannot distinguish key nodes in a Dinitrile group from those in a Morpholine group. △

This example suggests the importance of a broader study pertaining to structural and positional encodings.

[1] Rceall that throughout this thesis, we refer to the folklore version of the WL hierarchy [63].

Motif Structural Encoding as a Flexible and Powerful Approach. In this chapter, we propose *motif structural encoding* (MoSE) as a method of leveraging homomorphism count vectors to capture graph structure. Homomorphism counts have been investigated in the context of MPNNs to overcome well-known theoretical limitations in their expressivity [1, 27] with promising theoretical and empirical findings (see Section 4.2). Building on the existing literature, we show that MoSE is a general, flexible, and powerful alternative to existing positional and structural encoding schemes in the context of Transformers. In fact, we show that unlike RWSE, MoSE cannot be strictly confined to one particular level of the WL hierarchy (Proposition 4.3.3), and MoSE can provide significant expressiveness gains exceeding those achieved by RWSE (Theorem 4.3.6).

Example 4.1.2. Consider the graphs G and H from our running example, and observe that H has a cycle of length six, while G has no cycles. The node-level homomorphism counts $\mathsf{Hom}_{\to v_1}(\hexagon, G) \neq \mathsf{Hom}_{\to v_2}(\hexagon, H)$ (defined formally in Section 4.3.2) provide sufficient information to distinguish the nodes v_1 and v_2. Analogous statements can also be made for u_1 and u_2. △

MoSE also allows for known theoretical results on MPNNs [1] to be extended to the graph transformer domain. Our work therefore aims to address some of the unanswered questions related to the expressive power of structural encoding methods and bridge the gap between the theoretical study of MPNNs and graph transformers.

Main Results. The key results in this chapter can be summarized as follows:

- We introduce MoSE and detail the expressiveness guarantees that can be achieved by using homomorphism counts as a graph inductive bias (Section 4.3).
- We compare MoSE to RWSE in terms of expressivity and relate them to the expressive power of MPNNs through the well-known WL hierarchy (Section 4.3.2).
- More specifically, we show that MoSE strictly generalizes RWSE (Theorem 4.3.6). In fact, RWSE is simply a special case of MoSE (Proposition 4.3.5).
- We empirically validate our theoretical findings and demonstrate the efficacy of MoSE over existing encoding types on a variety of molecular datasets (Sections 4.4.2 to 4.4.4).
- We demonstrate the general nature of our approach by reporting consistent performance gains over other real-world and synthetic benchmarks as well (Sections 4.4.5 to 4.4.7).

4.2 Related Work

(Graph) Transformers and Encodings. Vaswani et al. [6] first highlighted the power of self-attention when they introduced the Transformer architecture. Dwivedi and Bresson [28] generalized the concepts presented in Vaswani et al. [6] to the graph domain. In recent years, several different graph transformer architectures have arisen. Yun et al. [119] first combined message-passing layers with Transformer layers, and other models have followed this approach [29, 120, 122]. Shehzad et al. [123] present a survey of existing graph transformer models and how they compare to each other.

A key component in the success of graph transformer models is the use of effective graph inductive biases [32]. Laplacian positional encodings (LapPE), which were introduced by Dwivedi and Bresson [28], are a popular choice, but they break invariance[2] as they are dependent on an arbitrary choice of eigenvector sign and basis [125, 126]. Rampásek et al. [29] present a framework for building general, powerful, and scalable graph transformer models, and they perform ablations with a suite of different encoding types, promoting the use of random-walk structural encoding (RWSE) on molecules. Ying et al. [127] introduced Graphormer, which uses an attention mechanism based on shortest-path (and related) distances in a graph. Additional encoding techniques based on distance [128] and hierarchical distance [129] have also been introduced. Ma et al. [30] present GRIT as a novel graph transformer that relies on relative random walk probabilities (RRWPs) for its graph inductive bias.

Expressive Power of GNNs. MPNNs capture the vast majority of GNNs [22] and their expressive power is upper bounded by 1-WL [23, 25]. This limitation has motivated a large body of work, including *higher-order* GNN architectures [25, 78–80], GNNs with unique node features [81], and GNNs with random features [82, 83], to achieve higher expressive power. The expressivity of GNNs has also been evaluated in terms of their ability to detect graph components, such as biconnected components [84], and in terms of universal approximation on permutation invariant functions [130]. Fully expressive architectures have been designed for special graph classes, e.g., for planar graphs [85].

Our work is most closely related to approaches that design more expressive architectures by injecting the counts of certain substructures, widely referred to as motifs. Bouritsas et al. [26] present an early work that enhances node feature

[2]By sacrificing invariance, it becomes much easier to design very expressive architectures. In fact, graph transformer architectures with LapPE are universal [124], but so is a 2-layer MLP and a single-layer GNN using LapPE (see Proposition 3.1 of Rosenbluth et al. [121]).

encodings for GNNs by counting subgraph isomorphisms for relevant patterns. Barceló et al. [27] follow up on this work by proposing to instead count the number of homomorphism from a set of pertinent substructures. Several other works have identified homomorphism counts as a key tool for understanding the expressive power of MPNNs [98, 106, 116]. Zhang et al. [94] also recently proposed a new framework that characterizes the expressiveness of GNNs in terms of homomorphism counts. Through tight connections between counting homomorphisms and counting (induced) subgraphs [53, 102, 107], these expressiveness results can be lifted to a wide range of functions [106]. As such, Jin et al. [1] built upon these theoretical observations to establish a general framework for using homomorphism counts in MPNNs. In particular, they show that homomorphism counts that capture certain "bases" of graph mappings provide a theoretically well-founded approach to enhance the expressivity of GNNs. Maehara and NT [131] reiterate the benefits of using homomorphism in GNNs.

4.3 Motif Structural Encodings (MoSE)

In this section, we formally define Motif Structural Encoding (MoSE). Just as with other node-level structural or positional encodings, MoSE provides a way to encode the structural characteristics of a node in terms of a numerical vector. Such encodings then provide graph inductive bias to models, such as Transformers, that cannot (or only in a limited fashion) take graph structure into account.

The *motif structural encoding* (MoSE) scheme is parameterized by a choice of a finite[3] pattern graph family $\mathcal{G} = \{G_1, \ldots, G_d\}$, as well as a choice of node weighting scheme ω which sends any (v, H) to a non-negative weight $\omega(v, H) \in \mathbb{R}_{\geq 0}$. For each node v of graph H, the node-level $\text{MoSE}_{\mathcal{G},\omega}$ label of v is:

$$\text{MoSE}_{\mathcal{G},\omega}(v, H) = \left[\omega\text{-}\mathsf{Hom}_{\to v}(G_i, H)\right]_{i=1}^{d} \in \mathbb{R}^d \tag{4.1}$$

We will notate $e_v^{\mathcal{G}} := \text{MoSE}_{\mathcal{G},\omega}(v, H)$ for shorthand when ω and H are either clear from context, or any arbitrary choice can be made. Unless otherwise specified, we use the constant ω weighting which sends all nodes in all graphs to 1, aligning $\text{MoSE}_{\mathcal{G},\omega}$ with the un-weighted homomorphism counts used in previous works [1, 27]. In this case, we omit ω from our notation, using $\text{MoSE}_{\mathcal{G}}$ to denote our encoding. When we reference $\text{MoSE}_{\mathcal{G},\omega}$ at the graph-level, we mean the multiset of node-labels $\text{MoSE}_{\mathcal{G},\omega}(H) = \{\!\{e_v^{\mathcal{G}} : v \in V(H)\}\!\}$. It holds that $\text{MoSE}_{\mathcal{G},\omega}(\cdot)$ is a graph invariant

[3]We require that $\mathcal{G}$ be a finite set, and that each graph within $\mathcal{G}$ have finite size.

because for any two graphs H and F with isomorphism $\iota : V(H) \to V(F)$, we have $e_v^{\mathcal{G}} = e_{\iota(v)}^{\mathcal{G}}$ for every vertex $v \in V(H)$ [56].

MoSE offers a highly general and flexible approach to structural encoding on account of two key parameters: the graphs $\mathcal{G}$ and the vertex weight function ω. Both of these parameters can be adapted to fit the problem domain and the model architecture as desired. The choice of $\mathcal{G}$ can be informed in precise terms by desired levels of expressiveness as shown below in Proposition 4.3.3. Furthermore, the choice of $\mathcal{G}$ can build on a range of empirical studies on structural information in MPNNs [1, 26, 27, 116]. Additional choice of a non-trivial weight function ω adds further power and flexibility. In particular, we show that even a simple weight function that maps nodes to the reciprocal of their degree is enough to exactly express RWSE in terms of MoSE (Proposition 4.3.5).

4.3.1 Expressiveness of MoSE

In the following, we will provide insight into the expressivity of MoSE by leveraging established connections between homomorphism counts and MPNNs. For Transformer architectures, similar frameworks of expressivity are not yet established. Recent work by Rosenbluth et al. [121] shows that Transformer architectures do not inherently contribute to expressivity, and that the positional/structural encoding is instead the key ingredient of their expressivity. We therefore study the expressivity of the encodings themselves, which in turn translates to the expressivity of models that utilize – and heavily depend on – these encodings. In particular, any typical graph transformer architecture with MoSE will naturally be at least as expressive as MoSE, and thus inherit all lower bounds from our analysis.

Our first two propositions establish that MoSE is, in a sense, incomparable to the WL-hierarchy. For every fixed set of graphs $\mathcal{G}$ that induces an encoding, we can provide an upper bound in terms of k-WL for some k dependent on $\mathcal{G}$. At the same time, given any fixed choice of k, the expressiveness of even MoSE encodings induced by a single pattern graph cannot be confined to the expressive power of k-WL (Proposition 4.3.2).

Proposition 4.3.1. *Let $\mathcal{G}$ be a finite set of graphs and let k be the maximum treewidth of a graph in G. Then, MoSE$_{\mathcal{G}}$ is at most as distinguishing as k-WL. That is, MoSE$_{\mathcal{G}} \preceq k$-WL.*[a]

[a] With respect to the definition of $\preceq$, both sides in this case represent singleton families of graph invariants.

Proof of Proposition 4.3.1. Dvorák [97] showed that if G and H are equivalent under k-WL, then every graph F with treewidth at most k has $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$. This extends also to the vertex level. Let $v \in V(G)$ and $v' \in V(H)$ such that v and v' have equivalent k-WL labels, and $\mathsf{Hom}(F, G, v) = \mathsf{Hom}(F, H, v')$ (see Lanzinger and Barceló [106], Lemma 12). Hence, if the maximum treewidth in $\mathcal{G}$ is k, then equivalence under k-WL implies also that they are equivalent under $\mathrm{MoSE}_{\mathcal{G}}$. □

Although we cannot distinguish *all* graphs distinguished by k-WL using $\mathrm{MoSE}_{\mathcal{G}}$ with a finite $\mathcal{G}$, we can distinguish any particular pair of graphs distinguished by k-WL. In fact, we only need a single graph in $\mathcal{G}$ in order to do this.

Proposition 4.3.2. *For $k \geq 1$, let G and H be two non-isomorphic graphs that are equivalent under k-WL. Then, there exists a graph F such that $\mathrm{MoSE}_{\{F\}}$ distinguishes G and H.*

Proof of Proposition 4.3.2. It is well known [54] that for every two non-isomorphic graphs G and H, there is a graph F such that $\mathsf{Hom}(F, G) \neq \mathsf{Hom}(F, H)$. We have that $\mathrm{MoSE}_{\{F\}}$ will distinguish the two graphs. □

Lanzinger and Barceló [106] and Jin et al. [1] studied the expressivity of MPNNs with respect to functions that map graphs to numbers. For a large class of these *graph motif parameters*, the distinguishing power of such a function relates closely to homomorphism counts from its homomorphism basis [1]. Building on these results, we establish a very broad lower bound for MoSE-enhanced Transformers while also providing a practical method for selecting $\mathcal{G}$.

Proposition 4.3.3. *Let f be a graph motif parameter with basis $\mathcal{G}_f \subseteq \mathcal{G}$. Then $\mathrm{MoSE}_{\mathcal{G}}$ is at least as distinguishing as f. That is, if $\mathcal{F}$ is the family of all graph motif parameters, we have $\mathcal{F} \preceq \mathrm{MoSE}$.*

Proof of Proposition 4.3.3. Suppose the opposite, i.e., there is a pair of graphs G, H such that $f(G) \neq f(H)$, but MoSE_{Γ_f}, where Γ_f is set of graphs in the basis of f, creates the same multiset of labels on both graphs. If this were the case, then $\mathsf{Hom}(F, G) = \mathsf{Hom}(F, H)$ for every $F \in \Gamma_f$. But since f is a graph motif parameter, $f(G) = f(H)$, and we arrive at a contradiction. □

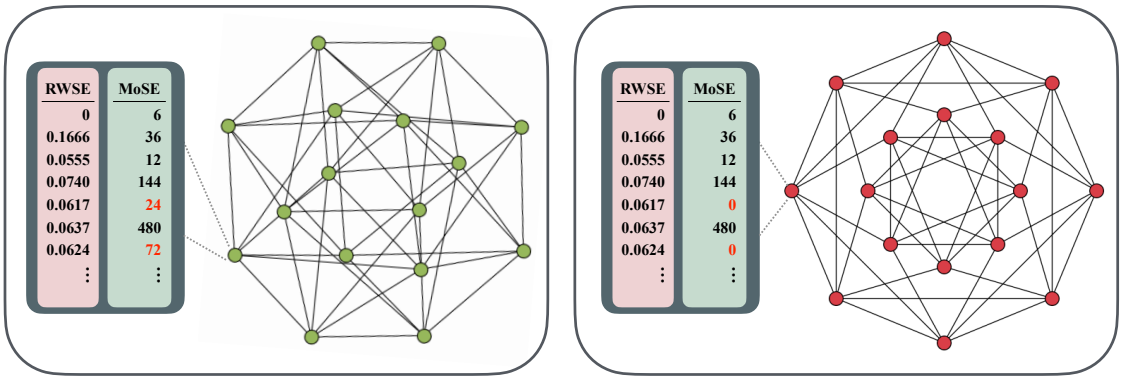


Figure 4.1: The 4×4 Rook's Graph (left) and the Shrikhande Graph (right) are non-isomorphic strongly regular graphs that have the same regularity parameters. RWSE produces the same vector on every vertex of both graphs whereas a simple construction of MoSE using $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ homomorphism counts easily distinguishes the two graphs.

4.3.2 Comparing the Expressivity of MoSE to RWSE

From above, we see that MoSE is closely aligned to the k-WL hierarchy, and we can use this alignment to inform the choice of $\mathcal{G}$. However, it is not possible to entirely contain MoSE within the WL hierarchy. In the following, we show that this is not the case for RWSE. In fact, the expressiveness of RWSE, regardless of length parameter, is fully contained within 2-WL.

Proposition 4.3.4. *For every $\ell \geq 2$, any graph which can be distinguished by RWSE$_\ell$ can be distinguished by 2-WL. That is, RWSE $\preceq$ 2-WL.*

Proof of Proposition 4.3.4. We first recall the walk refinement procedure from Lichter, Ponomarenko and Schweitzer [132] such that we can relate it directly to RWSE.

The scheme at its basis assumes a directed complete colored graph $G = (V, H, \chi)$ where the coloring χ always assigns different colors to self-loops than to other edges. For tuples of m vertices $(v_1, v_2, \ldots, v_m) \in V^m$, we define

$$\bar{\chi}(v_1, v_2, \ldots, v_m) = ((\chi(v_1, v_2), \chi(v_2, v_3), \ldots, \chi(v_{m-1}, v_m)).$$

Ultimately, we want to use the scheme on undirected uncolored graphs. An undirected (and uncolored) graph G, is converted into the setting above by adding the coloring $\chi : V(G) \to \{-1, 0, 1\}$ as follows: for all self-loops χ assigns -1, i.e., $\chi(v, v) = -1$. For all distinct pairs v, u, we set $\chi(v, u) = 1$ if $(v, u) \in E(G)$ and $\chi(v, u) = 0$ if $(v, u) \notin E(G)$. For the directed edges recall that we consider complete directed graphs and hence $E = V^2$.

With this in hand we can define the walk refinement procedure of Lichter, Ponomarenko and Schweitzer [132]. Formally, for $k \geq 2$, the k-walk refinement of a colored complete directed graph $G = (V, E, \chi)$ as the function that gives the new coloring $\chi_{W[k]}$ to every edge is as follows

$$\chi_{W[k]}(v, u) = \{\!\{\bar{\chi}(v, w_1, \ldots, w_{k-1}, u) \mid w_i \in V\}\!\}$$

Intuitively, the refinement takes into account all walks of length k in the graph. Note that walks of length shorter than k are also captured via self-loops (because of their different weights we are also always aware that a self-loop is taken and that a specific tuple in the multi-set represents a shorter walk. As with the Weisfeiler-Leman color refinement procedure, k-walk refinement can be iterated until it reaches a stable refinement (after finitely many steps). We will write $\chi^i(v, u)$ to designate the color obtained after i steps of the refinement (with $\chi(v, u) = \chi^0(v, u)$). Importantly, Lichter, Ponomarenko and Schweitzer [132] (Lemma 4) show that the stable refinements of the k-walk refinement procedure is always reached after finitely many steps and produces the same partitioning of vertices as 2-WL. That is, any pairs (u, v) and (x, y) that receive the same label by 2-WL refinement, also have $\chi^\infty(v, u) = \chi^\infty(x, y)$.

All that is then left is to show that the stable $W[k]$ refinement uniquely determines the RWSE_k feature vector for every vertex v. In particular, we show that every entry of the RWSE_k vector of any vertex v can be uniquely derived from the $W[k]$ labeling of the pair (v, v). As we will show below in Proposition 4.3.5, the k'-th entry of the RWSE_k vector for v, is uniquely determined by the number of k'-hop paths beginning and ending in v, together with the degrees along every such path. In the following, we observe that this combination of information can be directly computed from the $W[k]$ labeling of (v, v).

We first observe that $\chi^1_{W[k]}(v, v)$ uniquely determines the degree of v. It is equivalent to the number of k-walks that move to an adjacent vertex and continue with self-loops for the $k - 1$ remaining steps. That is:

$$\mathsf{degree}(v) = \mathsf{degree}(\chi^1_{W[k]}(v, v)) := |\{(1, -1, -1, \ldots, -1) \in \chi^1_{W[k]}(v, v)\}| \tag{4.2}$$

Similarly, it is straightforward to recognize the original label $\chi(v, u)$ from $\chi^1_{W[k]}(v, u)$:

$$\chi(v, u) = -1 \iff \{-1\}^k \in \chi^1_{W[k]}(v, u) \tag{4.3}$$

that is $v = u$ if and only if u can be reached from v only through self-loops. Similarly we can determine the other labels, and we only state the case for an edge existing here:

$$\chi(v, u) = 1 \iff (1, -1, -1, \ldots, -1) \in \chi^1_{W[k]}(v, u). \tag{4.4}$$

We can similarly retrieve the label $\chi(v, u)$ from $\chi^2_{W[k]}(v, u)$, since in any tuple of the form

$$((\chi(v_1, v_2), \chi(v_2, v_3), \ldots, \chi(v_{m-1}, v_m))$$

we know by Equation (4.3) and Equation (4.4) whether the respective pair of vertices are adjacent or equal. We can then naturally iterate the definitions from Equation (4.3) and Equation (4.4). Thus, for every $1 \leq k' \leq k$, we can directly enumerate all paths from v to v by inspecting the multiset $\chi^2(v, v)$.

To complete the argument outlined above, we need to show that for every v, u, $\chi^2(v, u)$ determines the degree of v. Now as above we can determine $\chi(v, u)$ and for each case observe how to obtain the degree. We have $\chi(v, u) = -1$, if and only if there is a tuple for the walk $(v, v, \ldots, v) \in \chi^2(v, u)$ (and the tuple can be recognized). The degree is then directly determined by the first tuple through Equation (4.2). Similarly, $\chi(v, u) = 1$, if and only if there is a tuple $(v, u, \ldots, u) \in \chi^2(v, u)$ that corresponds to original labels $(-1, -1, \ldots, -1, 1)$. We can thus again obtain the degree from the first position of this tuple, and analogously for the 0 label.

In summary, we see that after 2 steps of k-walk refinement, every label of (v, v) has enough information to uniquely determine all 1 to k-walks from v to v, together with the degree of every node of the walk. As seen in the proof of Proposition 4.3.5, this determines the RWSE_k vector. □

We note that the proof in fact shows a stronger statement than RWSE_k being at most as expressive as 2-WL. It shows that RWSE_k is at most as expressive as 2-steps of k-walk refinement. This can be further related to equivalence under a certain number of 2-WL steps through the work of Lichter, Ponomarenko and Schweitzer [132].

This in itself has wide-ranging consequences. For instance, it is known that 2-WL cannot distinguish strongly regular graphs [133], and therefore neither can RWSE. For MoSE, there exist no such limitations (Figure 4.1) beyond those dependent on the choice of $\mathcal{G}$ (Proposition 4.3.1).

While MoSE cannot fully capture 2-WL, it is in fact possible to express RWSE_ℓ for every $\ell \geq 2$ as a special case of MoSE. In contrast to previous results, our result here requires a node-weighting scheme that is not constant. However, even straightforward node-weighting by degrees – which comes at no cost in practice – is enough for our proof. Letting $\mathcal{G}$ consist of cycle graphs with size at most ℓ gives us the desired result.

Proposition 4.3.5. *For any $\ell \in \mathbb{N}$, there exists a finite family of graphs $\mathcal{G}$ and a weighting scheme ω such that the node-level $RWSE_\ell(v, H)$ label is uniquely determined by $MoSE_{\mathcal{G},\omega}(v, H)$ for all v and H. In fact, $RWSE_\ell$ is simply the special case of $MoSE_{\mathcal{G},\omega}$ where $\omega : v \mapsto 1/d(v)$ and $\mathcal{G} = \{C_i\}_{i=1}^{\ell}$.*

Proof of Proposition 4.3.5. We prove a statement stronger than our claim that MoSE can uniquely determine the RWSE vector: we show that – for an appropriate parametrization of MoSE – the MoSE vector and the RWSE vector of any node are *equivalent*. That is, RWSE is a special case of MoSE.

Take any graph H and notate $V(H) = \{1, 2, ..., n\}$. For $v, v' \in V(H)$, define $P(v \xrightarrow{i} v')$ to be the probability of starting a length i random walk at node v and ending it at node v'. Here, "length i" refers to the number of edges in the random walk where we allow repeat edges, and "random walk" refers to a walk where each step (say we are currently at node v) assigns a uniform probability $1/d(v)$ to moving to any of the neighbors in $\mathcal{N}(v)$.

As shorthand, define $\boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{A}$ where $\boldsymbol{D}$ and $\boldsymbol{A}$ are the diagonal degree matrix and adjacency matrix of H respectively. Let us first prove that $(\boldsymbol{M}^i)_{v,v'} = P(v \xrightarrow{i} v')$ for any nodes $v, v' \in V$ and for all powers $i \in \mathbb{N}$ by performing induction on i. The base case $i = 1$ holds trivially because we assume that random steps are taken uniformly over neighbors. Assume as the inductive hypothesis that our claim holds for some $i \geq 1$, and note that:

$$(\boldsymbol{M}^{i+1})_{v,v'} = (\boldsymbol{M}^1\boldsymbol{M}^i)_{v,v'} = \sum_{x=1}^{n} \boldsymbol{M}_{v,x} \cdot \boldsymbol{M}^i_{x,v'}$$

but our base case and inductive hypothesis tell us that

$$\sum_{x=1}^{n} \boldsymbol{M}_{v,x} \cdot \boldsymbol{M}^i_{x,v'} = \sum_{x=1}^{n} P(v \xrightarrow{1} x) \cdot P(x \xrightarrow{i} v').$$

By the law of total probability:

$$\sum_{x=1}^{n} P(v \xrightarrow{1} x) \cdot P(x \xrightarrow{i} v') = P(v \xrightarrow{i+1} v')$$

This completes the induction. Hence, for any node v in any graph H, the node-level RWSE_k vector is equivalent to:

$$\text{RWSE}_k(v, H) = \left[(\boldsymbol{M}^i)_{v,v}\right]_{i=1}^{k} = \left[P(v \xrightarrow{i} v)\right]_{i=1}^{k} \in \mathbb{R}^k.$$

Now, let us show that we can recover this random-walk vector using MoSE. Consider the family of all cycles with up to k nodes $\mathcal{G} = \{C_i\}_{i=1}^{k}$, and let ω be the vertex

weighting which maps any node to the reciprocal of its degree. Of course, the "cycles" C_1 and C_2 are not well-defined, so let us set C_1 to be the empty graph (with no nodes and no edges) and C_2 to be the graph with two nodes connected by an edge. We say that there are zero homomorphisms from C_1 into any other graph purely as a notational convenience aligning with the fact that $(\boldsymbol{M}^1)_{v,v}$ is always 0.

For each $C_i \in \mathcal{G}$ with $i > 1$, enumerate the nodes $V(C_i) = \{u_0, u_1, ..., u_{i-1}\}$ such that $(u_\ell, u_{\ell+1}) \in E(C_i)$ for all $\ell = 0, ..., i-2$ and $(u_{i-1}, u_0) \in E(C_i)$. Take any node v in any graph H, and define $\mathcal{H}_i$ to be the set of all homomorphisms from C_i to G which map node $u_0 \in C_i$ to node $v \in G$. Each homomorphism in $f \in \mathcal{H}_i$ can be constructed by making $i-1$ choices of node image $f(u_\ell) \in \mathcal{N}(f(u_{\ell-1}))$ for $\ell = 1, ..., i-1$ such that $f(u_{(i-1)}) \in \mathcal{N}(f(u_0))$ where we have fixed $f(u_0) = v$. In other words, each homomorphism $f \in \mathcal{H}_i$ corresponds to a i-edge walk (allowing edge repeats) starting and ending at node v, which we will call W_f. This correspondence is described by taking the induced subgraph $W_f = G[f(C_i)]$ where $f(C_i)$ is the image of C_i under homomorphism f. In the reverse direction, any i-edge walk

$$W_f : \quad v = v^{(0)}, v^{(1)}, ..., v^{(i-1)}, v^{(i)} = v \quad \text{in } H$$

that starts and ends at node v corresponds to a homomorphism $f \in \mathcal{H}_i$ which maps $f(u_\ell) = v^{(\ell)}$. Hence, we have described a bijective correspondence between $\mathcal{H}_i$ and the set of i-edge walks starting/ending at v.

For any $f \in \mathcal{H}_i$, we have:

$$\prod_{v \in V(C_i)} \omega(f(v)) = \prod_{v \in W_f} 1/d(v) = \prod_{\ell=0}^{i-1} 1/d(v^{(\ell)})$$

Since the probability of stepping from node $v^{(\ell)} \in V(W_f)$ to node $v^{(\ell+1)} \in V(W_f)$ in a random walk is simply $1/d(v^{(\ell)})$, it holds that:

$$\prod_{\ell=0}^{i-1} 1/d(v^{(\ell)}) = P(W_f)$$

where $P(W_f)$ is the probability of performing the random walk W_f. Thus,

$$\omega\text{-}\mathsf{Hom}_{u_0 \to v}(C_i, G) = \sum_{f \in \mathcal{H}_i} P(W_f) = P(v \xrightarrow{i} v)$$

giving us:

$$\mathrm{MoSE}_{\mathcal{G},\omega}(v, H) = \Big[\omega\text{-}\mathsf{Hom}_{\to v}(C_i, H)\Big]_{i=1}^{d} = \Big[P(v \xrightarrow{i} v)\Big]_{i=1}^{k} = \mathrm{RWSE}_k(v, H)$$

□

Proposition 4.3.5 immediately shows that RWSE $\preceq$ MoSE, and the example from Figure 4.1 demonstrates that this inclusion is in fact strict, giving us the following theorem:

Theorem 4.3.6. *Motif structural encoding is strictly more expressive than random walk structural encoding: RWSE $\prec$ MoSE.*

Furthermore, we can extend the analysis above to RRWP encodings. In fact, RRWP is a special case of pair-wise MoSE, where the patterns are paths with the same weighting scheme as for RWSE.

Finally, we note that on the node-level, there are even cases where RWSE is weaker than 1-WL. Specifically, there are nodes for which 1-WL assigns different labels, but RWSE_ℓ assigns the same label for every $\ell \geq 1$ (Example 4.1.1). There are also, of course, instances the other way around.

Proposition 4.3.7. *RWSE is incomparable to 1-WL in terms of node-level expressiveness.*

Proof of Proposition 4.3.7. One direction is shown in Example 4.1.1, which shows two graphs G and H, with highlighted green and black nodes each. For every step length ℓ, RWSE produces the same node features for both green vertices (u_1, u_2) and both black vertices (v_1, v_2), respectively. At the same time, it is straightforward to see that the 1-WL labeling of the green/black nodes is different in the two graphs.

For the other direction, we use the classic example comparing the graph G consisting of two triangles and the graph H containing the 6-vertex cycle as drawn below (Figure 4.2). It is well known that all nodes in both graphs have the same 1-WL label. In contrast, RWSE_3 labels the nodes in G differently from the nodes in H, and the respective vectors are given below. □

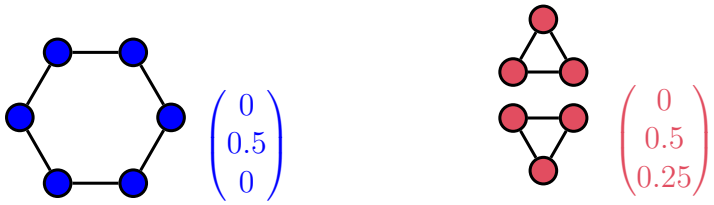


Figure 4.2: The 6 vertex cycle (left) and the disjoint sum of 2 triangles (right) are indistinguishable by 1-WL, but distinguishable by RWSE already after 3 steps. The RWSE vectors in the respective graphs are all the same and are given next to the graphs.

Now that we have presented several theoretical results pertaining to the expressivity of MoSE, we will supplement this discussion with a complexity analysis for computing these encodings.

4.3.3 Complexity of Computing MoSE

From a theoretical perspective, the complexity of MoSE is very intricate. With constant uniform weighting, counting homomorphisms from H to G is feasible in time $O(|G|^{tw(H)})$ where $tw(H)$ is the treewidth of H. It is known that this exponent cannot be substantially improved upon [101]. The complexity of computing MoSE encodings thus fundamentally depends on the choice of patterns and their treewidth.

As we show in Proposition 4.3.5, RWSE is simply a limited special case of MoSE for one specific set of pattern graphs (cycles) and one specific weighting scheme ($\omega(v) = 1/d(v)$). Cycles always have treewidth 2, and thus RWSE is always feasible in quadratic time. However, RWSE cannot capture other treewidth 2 motifs that MoSE can accommodate for essentially the same cost.

Practically, computing the homomorphism counts for MoSE is highly parallelizable since the homomorphisms from each pattern can be counted independently from one another. In our experiments, computing the MoSE encodings took only a fraction of a second per graph (see Table 4.12 in Section 4.4.8). By contrast, training various Transformer models on these datasets took between 15-30 hours for a single run. Hence, the time to compute MoSE is almost negligible overall.

In summary, we have provided theoretical results that relate the expressivity of graph transformers with MoSE to the WL hierarchy. We show that MoSE provides a strong inductive bias and is a principled and powerful way of capturing local graph structure. More specifically, we prove that MoSE fully generalizes the popular RWSE and RRWP encodings. We have also provided additional complexity analysis for MoSE encodings. Now, we will evaluate the practical utility of these theoretical results on a set of molecular datasets, as well as benchmarks from other domains.

4.4 Experimental Evaluation

We empirically demonstrate the efficacy of MoSE as a graph inductive bias on several real-world datasets across a range of models. First, we provide a broad comparison of MoSE with other positional and structural encoding methods using the GPS framework [29] in Section 4.4.1. Next, we specifically contrast the performance of MoSE and RWSE (and RRWP) on the ZINC molecular dataset in Section 4.4.2 and provide additional results on the PCQM4Mv2 dataset [134] in Section 4.4.3. After that, we conduct experiments on the LRGB Peptides dataset [135] to show the efficacy of MoSE on different types of molecules (Section 4.4.4).

To further highlight the flexibility of MoSE across multiple domains, we perform additional experiments on the CIFAR10 [33, 136] and MNIST [33, 137] image

classification datasets in Sections 4.4.5 and 4.4.6, respectively. Finally, we conduct a synthetic experiment to further evaluate the expressivity of MoSE and its ability to capture structural biases in Section 4.4.7. Unfortunately, we were unable to evaluate MoSE on the COLLAB dataset [115] used in Section 3.4.4 due to the high computational cost of training graph transformer models on graphs of this scale. Nevertheless, computing MoSE embeddings for COLLAB remains tractable for suitable choices of motif families, suggesting that MoSE could still be effectively combined with more computationally efficient architectures, such as MLPs or other non-graph transformer models.

In an attempt to establish the viability of MoSE in the most general sense and to align with previous works [1, 27], we the use the trivial vertex-weighting scheme in all our experiments where ω always maps every vertex to 1. Additional experimental details are reported in Appendix B.

4.4.1 Comparing MoSE to Other Encodings

We begin by extending the positional and structural encoding analysis from GPS [29] to include MoSE. We evaluate GPS+MoSE on three different benchmarking datasets, including ZINC [33, 41], PCQM4Mv2 [134], and CIFAR10 [33, 136]. ZINC and PCQM4Mv2 are both molecular datasets, whereas CIFAR10 is an image classification dataset. Following the protocol from Rampásek et al. [29], we use a subset of the PCQM4Mv2 dataset.

Dataset Setup. Recall that ZINC is a molecular dataset that presents a graph-level regression task to predict the constrained solubility of a molecule. Following Dwivedi et al. [33], we continue to use a subset of the ZINC dataset that contains 12,000 molecules. For this experiment, we also align with Rampásek et al. [29] and use the version of the ZINC dataset that includes edge features.

Similarly to ZINC, PCQM4Mv2 is a molecular dataset from the Large-Scale Open Graph Benchmark [134], where each graph represents a molecule, with nodes denoting atoms and edges denoting bonds. The dataset presents a graph-level regression task to predict the DFT-calculated HOMO-LUMO energy gap of a given molecule. PCQM4Mv2-subset is a subset of PCQM4Mv2. As mentioned in Rampásek et al. [29], this subset samples the original PCQM4Mv2 dataset as follows: 10% of the original training set, 33% of the original validation set, the entire test set. This results in 322,869 training graphs, 50,000 validation graphs, and 73,545 test graphs, for a total dataset size of 446,405 molecular graphs. In

Table 4.1: Performance of different positional (PE) and structural (SE) encoding types with the GPS model. MoSE consistently outperforms other encodings at relatively low computational cost. MAE stands for mean absolute error, and Acc. stands for accuracy.

Encoding	PE/SE	ZINC-12K	PCQM4Mv2-subset	CIFAR10
		MAE ↓	MAE ↓	Acc. ↑
none	-	0.113±0.007	0.1355±0.0035	71.49±0.19
PEG^{LapEig}	PE	0.161±0.006	0.1209±0.0003	72.10±0.46
LapPE	PE	0.116±0.009	0.1201±0.0003	72.31±0.34
SignNet^{MLP}	PE	0.090±0.007	0.1158±0.0008	71.74±0.60
SignNet^{DeepSets}	PE	0.079±0.006	0.1144±0.0002	72.37±0.34
RWSE	SE	0.070±0.002	0.1159±0.0004	71.96±0.40
MoSE	SE	**0.062**±0.002	**0.1133**±0.0014	**73.50**±0.44

order to align with Rampásek et al. [29], we use the exact same PCQM4Mv2-subset graphs that they use in their original ablation study.

The original CIFAR10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images in each class [136]. From this, Dwivedi et al. [33] generate a graph benchmarking dataset by constructing an 8 nearest-neighbor graph of SLIC (Simple Linear Iterative Clustering) superpixels for each image. The graph benchmarking dataset maintains the same splits as the original image dataset, which contains 45,000 training, 5,000 validation, and 10,000 test graphs.

Experimental Setup. For ZINC, we construct MoSE using $\mathcal{G} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ in accordance with Jin et al. [1]. For the PCQM4Mv2-subset, we use $\mathcal{G} = \Omega^{con}_{\leq 5} \cup C_6$, which is the set of all connected graphs with at most 5 nodes and the 6-cycle. For the CIFAR10 dataset, we simply use $\mathcal{G} = \Omega^{con}_{\leq 5}$, since there is no prior domain knowledge that indicates a 6-cycle would be useful in this context. We use the exact same model and training setup as described by Rampásek et al. [29] and constrain our reported model to corresponding parameter budgets, which are ∼500K for ZINC, ∼6M for PCQM4Mv2-subset, and ∼100K for CIFAR10. Following Rampásek et al. [29], we use 4 seeds for ZINC and CIFAR10, and 3 seeds for PCQM4Mv2-subset.

Results. Our results are presented in Table 4.1. The reported baselines for all encodings except MoSE are taken from Rampásek et al. [29]. We see that MoSE consistently outperforms other encoding types, including a number of spectral methods. Note that MoSE does not contain limitiations of LapPE such as sign ambiguity. When compared to RWSE, a prominent structural encoding, MoSE

achieves over 10% relative improvement on ZINC. Also, MoSE's strong performance on the CIFAR10 image classification benchmark highlights its effectiveness beyond molecular datasets.

4.4.2 Graph Regression on ZINC

To expand upon our evaluation on the ZINC molecular dataset, we provide a more in-depth set of experiments comparing MoSE to RWSE across a wide variety of architectures with varying degrees of inductive bias (Tables 4.2 and 4.3). We also compare MoSE to the RRWP encodings proposed in the GRIT architecture [30]. All reported results are the average of 4 runs with different random seeds, and specific hyperparameter details are provided in Appendix B.1.1.

For all experiments on ZINC, we follow Rampásek et al. [29] and use a random-walk length 20 for our RWSE encodings on ZINC. We continue to use $\mathcal{G} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ for MoSE encodings, which we also denote as $\text{MoSE}_{\mathcal{S}}$. Recall that we are using the trivial vertex-weighting scheme for MoSE, where ω always maps every vertex to 1. Although this implementation does not fully recover RWSE, we show that even this weaker version is enough to outperform RWSE in practice. Intuitively, any reasonably motivated ω increases the power of the encoding, and we therefore consider the current experiments a robust baseline for the performance of MoSE from which future work to explore the effect of more specialized vertex-weighting schemes can build upon.

Furthermore, while it might seem natural to experimentally validate Proposition 4.3.5, our proof shows that the use of $\mathcal{G} = \{C_i\}_{i=1}^{\ell}$ and $\omega : v \mapsto 1/d(v)$ will lead to the feature vectors of $\text{MoSE}_{\mathcal{G},\omega}$ being exactly the same as those obtained from the respective RWSE_{ℓ}, making an experimental comparison unnecessary. Therefore, although the motif sets in RWSE_{20} and $\text{MoSE}_{\mathcal{S}}$ are not exactly the same, they both contain a number of patterns with a maximum treewidth of 2. While $\text{MoSE}_{\mathcal{S}}$ contains smaller, more intricate motifs, RWSE_{20} corresponds to using significantly longer cycles, which can traverse more distant regions of the graph. As such, this comparison is both fair and insightful.

Comparing MoSE to RWSE across Multiple Architectures. Because Rampásek et al. [29] find that RWSE performs best on molecular datasets, we provide a detailed comparison of MoSE and RWSE across a range of architectures on ZINC. We select a baseline multi-layer perceptron (MLP-E), a simple self-attention Graph Transformer (GT-E), and GPS [29] for our models. Note that all models are adapted to account for edge features, and we will outline the changes below. For

reference, we include MPNN results for GIN-E [109] and its virtual-node extension GIN-E+VN, which is inspired by recent results on virtual nodes [138].

Model Formulations. Here, we give definitions for the MLP-E and GT-E architectures which are not taken directly from the literature.

MLP-E. Given a graph G and a node $v \in V(G)$, let the node representation $\boldsymbol{h}_v^{(0)} \in \mathbb{R}^d$ be a vector concatenation of the initial node label and the structural or positional encoding of v. To apply an L-layer MLP on G, we update each node representation in parallel for $t = 1, ..., L$:

$$\boldsymbol{h}_v^{(t)} = \text{ReLU}\left(\boldsymbol{W}^{(t)}\boldsymbol{h}_v^{(t-1)} + \boldsymbol{b}^{(t)}\right), \qquad \forall\ v \in V(G) \tag{4.5}$$

When edge features are available, MLP-E applies two MLPs in parallel to process node representations and edge representations separately. That is, node representations $\boldsymbol{h}_v^{(t)}$ are learned according to Equation (4.5) using a set of weights $\{\boldsymbol{W}_{\text{node}}^{(t)}, \boldsymbol{b}_{\text{node}}^{(t)}\}_{t=1}^{L}$, while the edge representations $\boldsymbol{e}_{u,v}^{(t)} \in \mathbb{R}^{d_e}$ are learned using a different set of weights:

$$\boldsymbol{e}_{u,v}^{(t)} = \text{ReLU}\left(\boldsymbol{W}_{\text{edge}}^{(t)}\boldsymbol{e}_{u,v}^{(t-1)} + \boldsymbol{b}_{\text{edge}}^{(t)}\right), \qquad \forall\ (u,v) \in E(G)$$

for $t = 1, ..., L_e$. Note that we *do not* require that $d = d_e$ and $L = L_e$.

GT-E. The GT model updates node representations by using scaled dot-product attention [6]:

$$\boldsymbol{h}_v^{(t)} = \text{MLP}^{(t)}\left(\boldsymbol{W}_O^{(t)} \bigoplus_{h=1}^{n_H} \boldsymbol{W}_{V,h}^{(t)} \sum_{u \in V(G)} \alpha_{v,u,h}^{(t)} \boldsymbol{h}_u^{(t-1)}\right) \tag{4.6}$$

$$\alpha_{v,u,h}^{(t)} = \text{softmax}_u\left(\frac{\boldsymbol{W}_{Q,h}^{(t)}\boldsymbol{h}_v^{(t-1)} \cdot \boldsymbol{W}_{K,h}^{(t)}\boldsymbol{h}_u^{(t-1)}}{\sqrt{d}}\right) \tag{4.7}$$

for $t = 1, ..., L$. Here, the $\oplus$ symbol denotes vector concatenation, and softmax is computed relative to the vector formed by iterating the expression inside the softmax argument over $u \in V(G)$. When we are provided with edge labels that come from a finite collection of discrete edge types, the GT-E models uses a learnable look-up embedding to learn a representation $\boldsymbol{e}_{u,v,h}^{(t)} \in \mathbb{R}^d$ for every node pair $u \neq v \in V(G)$, every attention head $h = 1, ..., n_H$, and every layer $t = 1, ..., L$. Non-adjacent node

pairs $(u, v) \notin E(G)$ are assigned some pre-defined "null edge-type." Then, these edge representations bias the attention as:

$$\alpha_{v,u,h}^{(t)} = \text{softmax}_u \left(\frac{\text{dot}\left[\boldsymbol{W}_{Q,h}^{(t)} \boldsymbol{h}_v^{(t-1)}, \boldsymbol{W}_{K,h}^{(t)} \boldsymbol{h}_u^{(t-1)}, \boldsymbol{e}_{u,v,h}^{(t)} \right]}{\sqrt{d}} \right) \tag{4.8}$$

where

$$\text{dot}[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}] = \sum_i \boldsymbol{a}_i \cdot \boldsymbol{b}_i \cdot \boldsymbol{c}_i$$

The node update equation of GT-E is defined by simply plugging in the edge-biased attention values $\alpha_{v,u,h}^{(t)}$ given by Equation (4.8) into Equation (4.6), the node update equation. Our method of "biasing" attention values with edge representations (Equation (4.8)) is inspired by the edge-type-aware graph transformer models of Dwivedi and Bresson [28] and Kreuzer et al. [124].

The final node representations are sum pooled in order to produce a graph representation $\boldsymbol{h}_G$. The MLP-E model additionally pools edge representations and concatenates the edge pool to the node pool:

$$\boldsymbol{h}_G = \sum_{v \in V(G)} \boldsymbol{h}_v^{(L)} \qquad \text{for MLP, GT, and GT-E}$$

$$\boldsymbol{h}_G = \left(\sum_{v \in V(G)} \boldsymbol{h}_v^{(L)} \right) \oplus \left(\sum_{(u,v) \in E(G)} \boldsymbol{e}_{u,v}^{(L_e)} \right) \qquad \text{for MLP-E}$$

Finally, an MLP prediction head is applied on $\boldsymbol{h}_G$ to produce an output label. Note that Equations 4.5–4.8 only describe the essential aspects of our models; when we implement these models in practice, we additionally include the usual regularization/auxiliary modules at each layer (e.g., batch normalization and dropout), as specified by the hyperparameter details in Appendix B.1.

Table 4.2: We report mean absolute error (MAE) for graph regression on ZINC-12K (with edge features) across several models. MoSE yields substantial improvements over RWSE for every architecture.

Model	Encoding Type		
	none	RWSE	**MoSE**
MLP-E	0.606±0.002	0.361±0.010	**0.347**±0.003
GIN-E	0.243±0.006	0.122±0.003	**0.118**±0.007
GIN-E+VN	0.151±0.006	0.085±0.003	**0.068**±0.004
GT-E	0.195±0.025	0.104±0.025	**0.089**±0.018
GPS	0.119±0.011	0.070±0.004	**0.062**±0.002

Results. MoSE consistently outperforms RWSE across all models in Table 4.2. Even though the graphs in ZINC are relatively small and a random-walk length of 20 for RWSE already traverses most of the graph, our experiments show that the structural information provided by MoSE yields superior performance.

We also note that GIN-E+VN becomes competitive with leading graph transformer models when using MoSE. This highlights the effectiveness of MoSE in supplementing the benefits of virtual nodes, an MPNN-enhancement aimed at capturing long-range node interactions [138]. This is consistent with our theoretical findings and aligns with recent results that suggest the use of virtual nodes can improve the performance of MPNNs [138].

Combining MoSE with RWSE. We also evaluate the combination of MoSE and RWSE on ZINC with edge features. Namely, we report RWSE+MoSE results where we concatenate the RWSE vector to the MoSE vector for a combined encoding. We use the same ZINC dataset setup as described above. All reported results are the mean of 4 runs with different random seeds.

Table 4.3: We report mean absolute error (MAE) for graph regression on ZINC-12K (with edge features) across several models. MoSE yields substantial improvements over RWSE for every architecture. Best results are shown in red, second best in blue.

Model	Encoding Type			
	none	RWSE	MoSE	RWSE+MoSE
MLP-E	0.606±0.002	0.361±0.010	0.347±0.003	0.349±0.003
GIN-E	0.243±0.006	0.122±0.003	0.118±0.007	0.117±0.005
GIN-E+VN	0.151±0.006	0.085±0.003	0.068±0.004	0.064±0.003
GT-E	0.195±0.025	0.104±0.025	0.089±0.018	0.090±0.005
GPS	0.119±0.011	0.070±0.004	0.062±0.002	0.065±0.002

Results. From Table 4.3, we see that the comparison between MoSE and the combination of RWSE+MoSE is less consistent. In models where there is global information flow, such as MLP-E, GT-E, and GPS, using MoSE alone performs better than the combination of RWSE+MoSE. In the more conventional GNN architectures such GIN-E and GIN-E+VN, the inclusion of MoSE improves performance over RWSE, but due to the information propogation issues of GNNs, MoSE performs slightly worse than the combination of RWSE+MoSE.

Although RWSE+MoSE contains more information than MoSE alone, there are other practical limitations of the combined encoding which may explain its comparatively-worse performance on some models – e.g., the extra information provided by *two* positional/structural encodings requires drastic changes in the model hyperparameters, such as an increase in model width, which may degrade performance more than the additional encoding information improves it (especially if the two encodings contain redundant information).

Regardless, the differences between MoSE and RWSE+MoSE are minimal across all models, and our experiment shows that MoSE alone is still highly competitive.

Comparing MoSE to RRWP. GRIT [30] is a recent graph transformer architecture that incorporates graph inductive biases without using message passing. GRIT utilizes relative random walk positional encodings (RRWP) at both the node representation level as well as the node-pair representation level. In our formulation, GRIT+MoSE, we remove all RRWP encodings for both the node and node-pair representations, and replace them with MoSE encodings at the node level.

Table 4.4: We achieve performance gains on ZINC by replacing the random walk encoding from GRIT with MoSE. MP denotes if the model contains a local message passing component or not.

Model	MAE ↓	MP
GSN	$0.101_{\pm 0.010}$	✓
CIN	$0.079_{\pm 0.006}$	✓
GPS	$0.070_{\pm 0.002}$	✓
Subgraphormer+PE	$0.063_{\pm 0.001}$	✓
GT-E	$0.195_{\pm 0.025}$	✗
GRIT+RRWP	$0.059_{\pm 0.002}$	✗
GRIT+MoSE	$\mathbf{0.056}_{\pm 0.001}$	✗

Results. Using MoSE in conjunction with the GRIT architecture [30] yields better results on ZINC, as detailed in Table 4.4. Recall that we substituted MoSE for RRWP only at the node level, completely omitting pairwise encodings. This suggests that even without explicit node-pair encodings, the relevant structural information captured by MoSE still surpasses that of RRWP. This is especially interesting given that GRIT is intentionally designed to exploit the information provided by RRWP, insofar that Ma et al. [30] present theoretical results ensuring that GRIT is highly expressive when equipped with RRWP encodings. Despite this, we show that GRIT achieves superior empirical performance when MoSE is used instead of RRWP.

4.4.3 Graph Regression on PCQM4Mv2-Subset

To supplement our reported results on PCQM4Mv2-Subset in Table 4.1, we provide additional results for GPS+MoSE on PCQM4Mv2-Subset using different pattern families. Specifically, we report results for MoSE constructed from $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$, denoted as $\text{MoSE}_{\mathcal{S}}$, in addition to the results for MoSE using $\Omega^{con}_{\leq 5} \cup C_6$, denoted as MoSE_{Ω}.

Results for the other encoding types presented in Table 4.5 are directly taken from Rampásek et al. [29]. In particular, Rampásek et al. [29] specify that they use a random-walk length of 16 for RWSE on the PCQM4Mv2-Subset dataset. Similarly to Section 4.4.2, although RWSE_{16}, $\text{MoSE}_{\mathcal{S}}$, and MoSE_{Ω} do not contain the exact same underlying motif sets, they still provide a meaningful comparison.

Table 4.5: We benchmark GPS+MoSE on PCQM4Mv2-Subset. Our pattern families are $\mathcal{S} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega = \Omega^{con}_{\leq 5} \cup C_6$. Results for other encoding types with GPS are included for reference [29]. Both constructions of MoSE outperform existing encoding types, especially RWSE. Best results are in red, second best in blue.

Encoding	MAE ↓
none	$0.1355_{\pm 0.0035}$
PEG^{LapEig}	$0.1209_{\pm 0.0003}$
LapPE	$0.1201_{\pm 0.0003}$
SignNet^{MLP}	$0.1158_{\pm 0.0008}$
$\text{SignNet}^{DeepSets}$	$0.1144_{\pm 0.0002}$
RWSE	$0.1159_{\pm 0.0004}$
$\text{MoSE}_{\mathcal{S}}$	$0.1137_{\pm 0.0034}$
MoSE_{Ω}	$0.1133_{\pm 0.0014}$

Results. Experimenting with different pattern families, we see in Table 4.5 that both formulations of MoSE outperform existing encoding methods on PCQM4Mv2-Subset. In general, it is also interesting to see that RWSE does not actually perform the best of the existing methods (it does worse than both SignNet models), even though PCQM4Mv2 is a molecular dataset. This suggests that perhaps longer cycles are less relevant for this molecular task, and may explain why MoSE_{Ω} performs slightly better than $\text{MoSE}_{\mathcal{S}}$. Recall that MoSE_{Ω} can capture many different graph motif parameters that are not restricted to the Spasm of cycles.

4.4.4 Graph Regression and Classification on Peptides

So far, we have evaluated MoSE on molecular datasets consisting of relatively small molecules (e.g., ZINC and PCQM4Mv2). In order to further validate the effectiveness of MoSE on a different class of molecular graphs, we conduct experiments on the Peptides-func and Peptides-struct datasets. The Peptides-func and Peptides-struct datasets were introduced in the Long Range Graph Benchmark [135]. Similar to other chemistry benchmarks, each graph represent a peptide, where nodes denote atoms and edges denote the bonds between atoms. However, peptides are generally much larger than the small drug-like molecules in datasets such as ZINC.

Experimental Setup. Peptides-func presents a 10-class multilabel classification task, whereas Peptides-struct presents an 11-target regression task. We compare the performance of MoSE with motif patterns $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$, MoSE with motifs $\Omega^{con}_{\leq 5} \cup C_6$, and RWSE of length 8 because each of these three encoding have similar "reach" (i.e., the edge-radius of each node encoding's receptive field). In order to isolate the power of our encodings, we decide to use a standard self-attention Graph Transformer [28] as our reference model. Since this model does not contain any built-in graph inductive biases, all expressive power is derived from the encodings. Also, note that GT does *not* utilize the edge features provided in the Peptides dataset because GT does not receive the adjacency matrix as an explicit input. Reported results are the average of 4 runs with different random seeds.

Table 4.6: We benchmark on the peptides-struct and peptides-func datasets using a self-attention Graph Transformer to isolate the performance gains of MoSE vs RWSE. Our pattern families are $\mathcal{S} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega = \Omega^{con}_{\leq 5} \cup C_6$. Best results are in red, second best in blue.

Model	Peptides-struct (MAE ↓)	Peptides-func (AP ↑)
GT+none	0.454±0.027	0.447±0.017
GT+RWSE	0.344±0.009	0.625±0.012
GT+MoSE$_{\mathcal{S}}$	0.339±0.007	0.635±0.011
GT+MoSE$_{\Omega}$	0.318±0.010	0.629±0.011

Results. We see from Table 4.6 that MoSE outperforms RWSE on both datasets. Due to our decision to use GT as our reference model, these results isolate the performance gains of MoSE over RWSE, since there are no other graph inductive biases (i.e. message passing layers) included in the GT model. Furthermore, we see that different constructions for MoSE yield varying performances on the two

datasets, although both pattern families outperform RWSE. This highlights the flexibility of MoSE in adapting to different tasks at hand. While we do not evaluate non-graph transformer models on the Peptides dataset, we would expect to see similar trends from Section 4.4.2, where MoSE outperforms RWSE across a variety of different model architectures. The results in this section mainly serve to demonstrate the effectiveness of MoSE on larger molecular graphs, but additional experiments comparing MoSE and RWSE using an MLP are presented later on in Section 4.4.7.

4.4.5 Image Classification on CIFAR10

Beyond molecular benchmarks, we also supplement our CIFAR10 results from Table 4.1 with a more in depth expreimental analysis. Recall that CIFAR10 is an image classification dataset, where the goal is to classify images into 10 different classes [136].

Our reported results in Table 4.1 use homomorphism counts from the family $\Omega^{con}_{\leq 5}$. We now experiment with a different pattern family constructed from $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$. We also evaluate how performance changes when we use a larger model that exceeds the 100K parameter budget. Results are reported as the average of 4 runs with different random seeds.

Table 4.7: We benchmark GPS+MoSE on CIFAR10 with pattern families $\mathcal{S} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega = \Omega^{con}_{\leq 5}$. GPS+LapPE results are included for reference [29]. Best results are in red, second best in blue.

Model	Accuracy ↑	Parameters
GPS+LapPE	72.31±0.34	112,726
$\text{GPS}_{\text{small}}$+$\text{MoSE}_{\mathcal{S}}$	73.34±0.50	110,188
GPS_{big}+$\text{MoSE}_{\mathcal{S}}$	75.00±0.59	267,134
$\text{GPS}_{\text{small}}$+$\text{MoSE}_{\Omega}$	73.50±0.44	114,330
GPS_{big}+MoSE_{Ω}	74.73±0.64	215,794

Results. Experimenting with different model sizes and pattern families, we see that MoSE consistently outperforms LapPE on CIFAR10 (Table 4.7). We use GPS+LapPE for reference here, since Rampásek et al. [29] found that it performed the best with their GPS model on this dataset. Furthermore, we see that larger models tend to perform better, likely because they can better leverage the additional information provided by the MoSE encodings.

We provide an additional comparison of our best GPS+MoSE configuration and compare it against other leading GNN and graph transformer models in

Table 4.8: Comparison of GPS+MoSE on CIFAR10 against leading GNN and graph transformer based methods. GPS+MoSE performs the best across all models.

Model	Accuracy ↑
GCN	55.71±0.38
GIN	55.26±1.53
GAT	64.22±0.46
GatedGCN	67.31±0.31
PNA	70.35±0.63
DGN	72.84±0.42
CRaWl	69.01±0.26
GIN-AK+	72.19±0.13
EGT	68.70±0.41
GPS(+LapPE)	72.31±0.34
GPS+MoSE	**75.00**±0.59

Table 4.8. These results further solidify the effectiveness of MoSE on domains beyond molecular modelling.

4.4.6 Image Classification on MNIST

Due to the success of MoSE on CIFAR10, we also evaluate GPS+MoSE on the MNIST image classification dataset. The original MNIST dataset consists of 70,000 28x28 black and white images in 10 classes, representing handwritted digits [137]. From this, Dwivedi et al. [33] follow the same procedure as for CIFAR10 and generate a graph benchmarking dataset by constructing an 8 nearest-neighbor graph of SLIC superpixels for each image. The graph dataset also has the same original dataset splits of 55,000 training, 5,000 validation, and 10,000 test graphs.

We conduct similar experiments on MNIST as we did for CIFAR10 in terms of pattern families and model sizes. We do not grid-search GPS+MoSE on MNIST, and instead we simply extrapolate from our hyperparameters on CIFAR10 to infer our model configurations. All results for GPS+MoSE are the mean of 4 runs with different random seeds, with GPS+LapPE provided for reference, since that was found to perform best previously [29].

Results. Once again, we see that larger models tend to perform better (Table 4.9). However, since MNIST is a simpler dataset, there is minimal performance gain, as this benchmark seems quite saturated already. Comparing GPS+MoSE with other leading GNN and graph transformer models in Table 4.10, we see that GPS+MoSE

remains competitive with leading GNNs, which contain message passing modules, on MNIST image classification.

Table 4.9: GPS+MoSE on MNIST with various $\mathcal{S} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega = \Omega^{con}_{\leq 5}$ configurations. Best results are in red, second best in blue.

Model	Accuracy ↑	∼Parameters
GPS+(LapPE)	98.051±0.126	100K
$\text{GPS}_{\text{small}}$+$\text{MoSE}_{\mathcal{S}}$	98.090±0.050	100K
GPS_{big}+$\text{MoSE}_{\mathcal{S}}$	98.273±0.075	400K
$\text{GPS}_{\text{small}}$+$\text{MoSE}_{\Omega}$	98.135±0.002	100K
GPS_{big}+MoSE_{Ω}	98.198±0.180	200K

Table 4.10: Comparison of GPS+MoSE on MNIST against leading GNN and graph Transformer based methods. GPS+MoSE performs the best across all models.

Model	Accuracy ↑
GCN	90.71±0.22
GIN	96.49±0.25
GAT	95.54±0.21
GatedGCN	97.34±0.14
PNA	97.94±0.12
CRaWl	97.94±0.05
EGT	98.17±0.09
GPS(+LapPE)	98.05±0.13
GPS+MoSE	**98.27**±0.08

4.4.7 Predicting Fractional Domination Numbers

To finalize our experimental evaluation, we generate a new synthetic dataset where the task is to predict each graph's *fractional domination number*. This target is an inherently global proprety, so it relies on complex long-range interactions, unlike other popular metrics (e.g., clustering coefficients) which are aggregations of local properties. The target distribution of our dataset is complex (Figure 4.3), making the task highly challenging, and fractional domination is important for a range of applications, such as optimizing resource allocation [139].

Dataset Setup. Our sythetic dataset contains 10,000 randomly-generated Erdős-Rényi graphs, where edge density varies between [0.25, 0.75] and the number of nodes varies between [16, 32]. There are *no* initial node features or edge features provided.

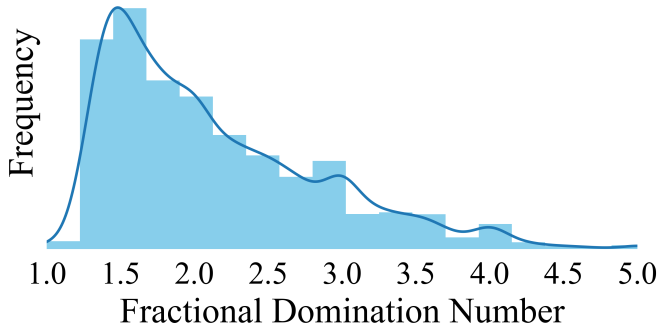


Figure 4.3: Plot of the distribution of fractional domination numbers for the graphs in our synthetic dataset.

A fractional dominating function of a graph G is a weight assignment $\alpha : V(G) \to [0, 1]$ such that for every vertex v, the sum of the weights assigned to v and its neighbors is at least 1. The total weight of α is given by $\sum_{v \in V(G)} \alpha(v)$. The *fractional domination number* of G is the smallest total weight of any fractional dominating function on G [140].

Experimental Setup. We focus on isolating the performance of MoSE compared to LapPE and RWSE. For this, we select a 4-layer MLP and a basic Graph Transformer [28] as our reference models. Since MLP and GT do not receive the adjacency matrix, initial node features, or initial edge features as an input: the model must infer the fractional domination number using *only* the node-level positional/structural encoding vectors provided. Hence, all structure-awareness comes from the encodings. We choose $\mathcal{G} = \mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ for MoSE$_{\mathcal{G}}$, random walk length 20 for RWSE, and dimension 8 for LapPE. We report the mean of 4 runs with different random seeds. For each model, we do not perform any hyperparameter tuning. In order to control for the large homomorphism count values generated by MoSE on this dataset, we scale the raw counts down by taking the $\log_{10}$ of the true values.

Table 4.11: MAE results for our synthetic dataset. MoSE consistently performs the best, with MLP+MoSE even outperforming the Graph Transformer with LapPE and RWSE.

Encoding	MLP	GT
LapPE	0.482±.003	0.084±.001
RWSE	0.075±.001	0.061±.001
MoSE	**0.058**±.001	**0.055**±.001

Results. Table 4.11 shows that MoSE outperforms LapPE and RWSE across both models. Strikingly, even the simple MLP+MoSE model outperforms the more advanced GT+RWSE and GT+LapPE models. These results suggest that MoSE is able to capture complex graph information in a context broader than just the molecule and image domains. We also see that MLP+LapPE performs significantly worse than all other configurations. This indicates that the self-attention mechanism of GT is a key component in allowing the model to leverage spectral information provided by LapPE. Furthermore, the comparable performance of MLP+MoSE and GT+MoSE shows that, for certain tasks, MoSE provides a sufficiently informative graph inductive bias, and complex architectures such as graph transformers are not required to achieve strong performance. More broadly, this highlights the fact that expressive structural encodings can enable simpler models, such as MLPs, to perform competitively while benefiting from improved scalability and efficiency.

This thus concludes our experimental evaluation of MoSE. Across 3 diferent molecular datasets and 3 datasets from other domains, we have shown that MoSE is an effective and flexible graph inductive bias for capturing local (and to some extent even global) structure. This in turn leads to stronger empirical performance in many architectures with varying degrees of inherent bias.

4.4.8 Practical Runtime for Computing MoSE Encodings

We report the practical runtime for computing MoSE encodings on all datasets used in our experiments in Table 4.12. From these reported times, we can see that computing homomorphism basis counts is indeed extremely fast, which makes our approach feasible for many widespread applications. Recall that it is also well known that homomorphism counting is closely related to database theory, where homomorphism counts can be formulated as SQL JOIN queries [117]. This unlocks the possibility of integrating more complex graph motif parameters from relational or tabular data into GNNs. It also allows for the computation of MoSE encodings on larger datasets, which may currently be infeasible with other tools at hand.

Since MoSE generalizes popular encodings, we see it also as a way to trade off more compute to achieve a more informative (and thus ideally better performing and generalizing) encoding. For example, it is possible to start with quickly computing RWSE-like encodings (since we have shown in Proposition 4.3.5 that MoSE recovers RWSE) using matrix multiplication, and then incrementally adding only the desired motif patterns to $\mathcal{G}$ until a target expressivity level for MoSE is reached. This kind of adaptability is not possible in RWSE or other popular encodings.

Table 4.12: Time to compute MoSE embeddings used in all experiments reported as elapsed real time (wall time). Reported times are normalized by dataset size (i.e. per graph). $\Omega^{con}_{\leq 5}$ refers to the set of all connected graphs with at most 5 nodes, and C_i is the cycle graph on i nodes. n_G denotes the number of graphs in the dataset, and $\overline{|V|}$ and $\overline{|E|}$ indicate the average number of nodes and edges respectively. Note that [a] denotes both $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega^{con}_{\leq 5} \cup C_6$ basis sets, and [b] denotes both $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$ and $\Omega^{con}_{\leq 5}$ basis sets.

Dataset	n_G	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	Basis	Time
ZINC	12,000	23.2	49.8	$\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$	0.03s
PCQM4Mv2-subset	446,405	14.1	14.6	Both[a]	0.05s
Peptides	15,535	150.9	307.3	Both[a]	0.01s
CIFAR10	60,000	117.6	941.1	Both[b]	0.10s
MNIST	70,000	70.6	564.5	Both[b]	0.08s
Synthetic	10,000	23.5	137.9	$\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$	0.06s

4.5 Summary and Outlook

In this chapter, we have proposed Motif Structural Encoding (MoSE) as a flexible and powerful graph inductive bias based on counting graph homomorphisms (Section 4.3). We prove several theoretical results regarding the expressivity of MoSE both in general (Section 4.3.1) and in relation to other popular encoding schemes, such as RWSE, and relate these to the expressive power of MPNNs through the WL hierarchy more generally (Section 4.3.2). Empirically, we validate the practical effects of our theoretical results, achieving strong benchmark performance across a variety of real-world and synthetic datasets (Section 4.4).

In terms of specific theoretical contributions, we have proven that MoSE is supported by expressiveness guarantees that translate naturally to architectures using MoSE (Proposition 4.3.3). Additionally, we have proven that in a way, MoSE is somewhat incomparable to the WL-hierarchy (Propositions 4.3.1 and 4.3.2), whereas the expressivity of RWSE is fully contained within 2-WL (Proposition 4.3.4), no matter the random walk length. We also show that it is possible to express RWSE as a special case of MoSE (Proposition 4.3.5), and furthermore, that MoSE strictly generalizes RWSE (Theorem 4.3.6). Regarding node-level expressiveness, we also prove that RWSE is incomparable to 1-WL (Proposition 4.3.7) as there are graphs that can be distinguished by one but not the other, and vice versa.

For experimental contributions, we show that MoSE is able to achieve strong performance gains over existing encoding schemes for the GPS graph transformer

model on a variety of graph learning benchmarks (Section 4.4.1). Aligning with our overall theme of focusing on molecular applications, we provide in depth experiments on molecular property prediciton for the ZINC (Section 4.4.2) and PCQM4Mv2 (Section 4.4.3) benchmarks, showing the real-world benefits of MoSE's ability to better capture local structure. This extends to larger molecular graphs as well, with experiments on the Peptides dataset from the Long Range Graph Benchmark (Section 4.4.4).

Beyond molecular tasks, we also show that the benefits of MoSE extend to image classification tasks on CIFAR10 (Section 4.4.5) and MNIST (Section 4.4.6), demonstrating that our results generalize to other domains. Finaly, we introduce a new synthetic benchmark for predicting fractional domination numbers (Section 4.4.7), where we again see that MoSE provides strong graph inductive biases for weaker models, such as MLPs (Section 4.4.7). The results also suggest that MoSE is able to somewhat capture complex global structure as well.

Although our work provides a deeper understanding of the expressivity induced by different graph inductive biases, several open questions remain. In particular, the relationship between MoSE and spectral methods warrants further investigation, and the full potential of the node-weighting parameter in our encoding scheme has yet to be explored. For larger or more complex graphs, there may also be practical limitations to the information gained from counting increasingly large motif patterns. Additional work is also needed to investigate motif families beyond the Spasm of cycles and set of all connected graphs up to a fixed size, which are used in our current experiments. As mentioned previously, an important direction for future research lies in learning important motifs automatically rather than hand-selecting them. This may help avoid the inclusion of irrelevant motif features that could act as noise and degrade model performance.

Overall, we find that graph homomorphism counts provide a strong theoretical framework for introducing graph inductive biases into graph transformer models. MoSE not only offers solid theoretical guarantees of expressiveness, but these guarantees also translate into strong empirical performance across a variety of datasets—particularly within the molecular domain. This chapter contributes to a better understanding of how to capture local graph structure through positional and structural encodings, which are essential for graph transformers and other architectures that lack inherent graph inductive biases. In the next chapter, we consider a slightly different definition of local structure, but again highlight how informed model representation decisions can have tangible impacts for molecular modelling.

5

Leveraging Local Interactions to Capture Global Symmetry in Periodic Crystals

In the previous chapter, we showed how Motif Structural Encoding is a powerful encoding technique for capturing intramolecular local graph structure within a single molecular graph. In this chapter, we consider the second notion of local structural awareness, which pertains to intermolecular interactions *between* molecular graphs in a periodic crystal. In particular, we introduce OXtal, a new generative model for molecular crystal structure prediction. By learning local interactions between molecules, we are able to capture global crystal packing symmetries and scale to much larger models and molecular crystals. More specifically, we remove explicit model equivariance, we do not use a unit cell representation during training, and we introduce a novel data augmentation scheme, Stoichiometric Stochastic Shell Sampling, which is inspired by the crystallization process and implicitly introduces soft symmetries. Due to the applied nature of this challenge, the contributions in this chapter are more methodological and contain an in-depth chemical analysis to highlight the practical relevance of OXtal and the importance of adapting local representations to the problem at hand.

The content of this chapter is largely based off of the work published in Jin et al. [3], with theoretical results in Proposition 5.4.1 completed in collaboration with Alexander Tong, A-Transformer baseline results in Section 5.5 completed in collaboration with Andrei Nica, and chemical analysis results in Section 5.5 completed in collaboration with Chenghao Liu.

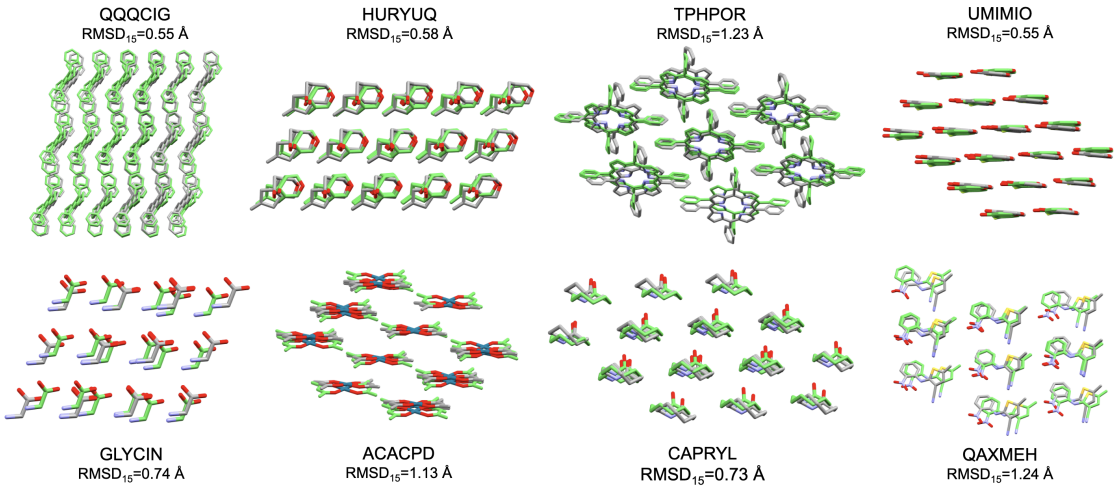


Figure 5.1: Molecular crystal structures generated by OXTAL (color) compared to ground truth (grey).

5.1 Introduction

In 1988, the editor of Nature, John Maddox, famously declared, "One of the continuing scandals in the physical sciences is that it remains in general impossible to predict the structure of even the simplest crystalline solids from a knowledge of their chemical composition" [43]. Nearly forty years later, this problem of *Crystal Structure Prediction (CSP)* remains one of the grand challenges in computational chemistry [37–39]. In particular, given only a molecule's 2D chemical graph, *ab initio* molecular CSP seeks to estimate the distribution of experimentally realizable 3D crystal packings in an accurate and scalable manner.

This 3D arrangement of molecules within a periodic lattice dictates the macroscopic behavior of organic solids. For instance, in pharmaceuticals, crystal packing governs solubility, bioavailability, and the long-term stability of active ingredients [141, 142]; in material science, intermolecular geometry dictates charge transport, porosity, and optical response—enabling applications across electronics, photonics, sensing, and energy storage [143, 144].

The complexity of CSP is underscored by the nature of crystal formation, wherein experimentally realized structures often occupy local minima of a highly non-smooth and well-separated Gibbs free energy landscape (Figure 5.3). This thermodynamic energy landscape is determined by the competition between the *intramolecular* interactions that set the molecule's own (flexible) conformation and the long-range and weak *intermolecular* forces that dictate how molecules pack together periodically [52]. As a result, classical CSP approaches combine a search procedure (e.g., enumeration or evolutionary algorithms) with oracle

access to energy ranking models, such as force fields or quantum-chemical density functional theory (DFT) [39, 145].

Unfortunately, classical CSP approaches often fail to capture realistic *kinetic* conditions that lead to the *distribution* of experimentally sampled energy minima. Consequently, these methods require the generation and optimization of ~1,000 to 100,000 structures per molecule—the majority of which struggle to go beyond unfavorable local energy minima despite extensive computation. This is especially difficult for flexible molecules, which contain many rotatable bonds that can lead to several different 3D molecular conformations.

In fact, the computational cost of these brute-force methods is so significant that the most recent 7th CSP blind test competition stated, "With more than **46 million CPU core hours** reportedly utilized for the structure generation phase of this seventh blind test alone, we cannot avoid commenting on the need for the community to carefully consider the economic and environmental impact of CSP. Scientific research, and possible future blind tests, should better allow for the ethical use of natural, computational and economic resources and focus on developing rational and efficient algorithms for CSP, rather than naïve brute force methods" [39].

Recently, *generative* approaches to modelling atomic systems–e.g., AlphaFold3 [11] for biomolecules and MatterGen [18] for inorganic materials–have demonstrated their ability to capture intricate 3D atomistic interactions directly from data. Molecular CSP generalizes both of these, as proteins and inorganic crystals have a smaller set of interactions; proteins pack into lattices under strong intramolecular constraints mostly governed by their backbones [146], and inorganic crystals possess strong covalent or ionic bonds across a smaller number of atoms (Figure 5.2a). Protein generative models also heavily rely on biological priors based on evolutionary information captured by multiple-sequence alignment [10]. In contrast, small-molecule crystals span chemically diverse scaffolds, exhibit rich conformational flexibility, and often contain many molecular copies within a unit cell (Figure 5.2b). Accurately capturing these interactions requires a large and diverse training set, highly expressive yet efficient to sample models, and training schemes that consider periodic interactions in a crystal lattice without impeding scalability for large model training.

In this chapter, we introduce OXtal, an all-atom diffusion transformer model for molecular CSP. Conditioned solely on the 2D molecular graph, OXtal learns to sample experimentally realistic crystal structures with accurate descriptions of molecular conformers as well as their periodic packing. To enable large-scale modelling, we train OXtal on roughly 600k experimental molecular crystal

structures spanning rigid and flexible molecules, co-crystals, and solvates. OXtal builds on recent advances in all-atom generative modelling [11], discarding symmetry representations of lattice vectors and associated crystal symmetries in favor of training directly on Cartesian coordinates and employing SE(3) data augmentation. We further introduce Stoichiometric Stochastic Shell Sampling (S^4), a novel lattice-free training scheme that retains the rich information of long-range interactions.

Main Results. The key results from this chapter can be summarized as follows:

- We present OXtal, the first large-scale all-atom diffusion model for molecular CSP, that samples molecular crystal packing directly from 2D molecular graphs (Section 5.4).
- We introduce a crystallization-inspired training scheme for periodic structures, S^4, which removes explicit lattice parametrization and enables more scalable training (Section 5.4.1). We further bound the error due to S^4 in Proposition 5.4.1.
- Empirically, OXtal significantly outperforms existing ML-based *ab initio* CSP methods, achieving state-of-the-art results by over a magnitude across a variety of metrics including collision rate, packing similarity rate, conformer recovery rate, and approximate solve rate (Section 5.5.2).
- Compared to traditional quantum chemical methods, OXtal remains competitive in terms of collision and packing similarity rates, while being several orders of magnitude cheaper (Section 5.5.3).
- We conduct a series of ablation studies on OXtal to understand the impact of various components, including the S^4 training scheme, S^4 radius size, model size, and impact of enforcing explicit equivariance (Section 5.5.4).
- We provide additional chemical analysis on inference and interpretability to highlight OXtal's ability to model global symmetry and capture diverse intramolecular and intermolecular interactions in crystal polymorphs, complex co-crystals, and biomolecules (Sections 5.5.5 and 5.5.6, respectively).

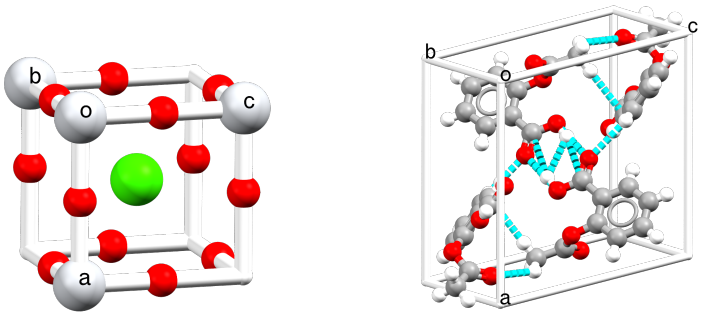
5.2 Related Work

Physical Approaches to Crystal Structure Prediction. Classical crystal structure prediction (CSP) mostly applied search-based algorithms to sample a pre-defined search space [39]. Some classical methods infused domain knowledge, such as starting with initial guesses like a unit cell, and employed varying parameters, random sampling [147–149], guidance from force-fields [150], or constructed guesses

based on chemical principles [151]. Recent methods have also applied more structured search algorithms, such as simulated annealing [152, 153], genetic algorithms [154, 155], particle swarm optimization [156], and basin-hopping [157]. While many such methods have shown varied degrees of success for different applications [147–150, 157], they fundamentally rely on many calls to an expensive evaluation function (e.g., energy computation using DFT) and struggle to leverage prior data effectively.

ML Potentials. Computational complexity of DFT and availability of large datasets [158–161] enabled the development of universal machine learning interatomic potentials (MLIP) [34, 162–166] to predict energy and forces of different atomistic systems (from small molecules to inorganic crystals) at a fraction of DFT costs. The next generation of physical approaches to CSP replaced DFT with MLIPs and showed benefits in material discovery [167] with the recent FastCSP model [168] applied to organic CSP. Other recent search and optimization approaches have also replaced DFT with machine learning interatomic potentials [165, 166, 168]. However, these methods still rely on a search-and-rank procedure. In contrast, OXtal does not require explicit energy or function calls and performs zero-shot structure prediction.

Generative Models for Inorganic Crystal Structure Prediction. Recently, generative models have emerged as a promising new paradigm for inorganic crystal structure prediction. For inorganic crystal structures, which consist of a periodic unit cell of atoms, generative models were first applied for unconditional de-novo generation of new crystals [45] and later used for structure generation conditioned on crystal composition [47, 49, 169, 170]. The use of generative models has spanned multiple modeling methods, including diffusion models with equivariant models [47, 169], symmetry-aware diffusion models [170], and flow-matching models on specialized manifolds [49]. Recent work have also applied large-language models to crystal generation and structure prediction with more varied success compared to other methods [171, 172]. In contrast to molecular crystals, which consist of discrete molecules and exhibit rich conformational and packing polymorphism, inorganic crystals typically comprise of smaller unit cells governed by strong inter-atomic bonding, resulting in rigid continuous lattices rather than flexible molecular packing (see Figure 5.2). OXtal relies on the notion of supercells and asymmetric units but tackles the more complex challenge of organic CSP, where each "asymmetric unit" contains organic molecules.



(a) $CaTiO_3$ (inorganic) **(b)** Asprin (organic)

Figure 5.2: Molecular crystals consist of distinct molecules held together via long-range, weak interactions. They typically contain many atoms per unit cell and an unknown number of molecule copies Z.

Protein Structure Prediction. Generating low-energy atomic arrangements is not unique to CSP. In computational structural biology, the analogous task of protein structure prediction (PSP) conditioned on a protein sequence has seen transformative progress with landmark models like AlphaFold [10, 11] and ESMFold [173], achieving high accuracy by generating conformations conditioned on a protein sequence. More recently, generative models of protein structure have also found great success in the de novo protein design problem [12, 174, 175]. These models work with a few dozen residue types, compared to a larger chemical space in general molecular CSP, and use evolutionary information not present in molecular crystals. Our model, OXTAL, builds on the successes in protein structure prediction and de novo generation, adapting insights from these problems to the organic crystal structure generation problem, where the lattice structure and symmetries necessitate a tailored strategy.

5.3 Molecular Crystal Structure Prediction

Crystal Representations. Formally, a periodic crystal structure $\mathcal{C}$ is defined by a pair $(L, \mathcal{B})$. The first component $L \in \mathbb{R}^{3\times3}$ defines the lattice vectors forming a parallelepiped known as the unit cell. The second component $\mathcal{B} = \{(z_i, u_i)\}_{i=1}^N$ is the basis, which consists of the N atoms within this unit cell. Each atom is described by its species $\{z_i\}_{i=1}^N$ and its fractional coordinates $\{u_i \in [0,1)^3\}_{i=1}^N$ relative to the lattice vectors, or equivalently, its Cartesian coordinates Lu_i. For molecular crystals, $\mathcal{B}$ naturally decomposes into Z molecules. The connectivity of each molecule m_k is given by a graph $\{g_k = (V_k, E_k)\}_{k=1}^Z$ (vertices/atoms V_k and edges E_k; species labels z_i are carried by the vertices). We next recall the symmetries present in the periodic crystal structure $\mathcal{C}$.

Symmetries. Given $\mathcal{C} = (L, \mathcal{B})$ where $(u_i \in \mathbb{T}^3 := \mathbb{R}^3/\mathbb{Z}^3)$, the Cartesian positions of atoms are,

$$X(L, \mathcal{B}) = \{\, L(n + u_i) : \; n \in \mathbb{Z}^3, \; i = 1, \ldots, N \,\}. \tag{5.1}$$

Furthermore, two descriptions $(L, \mathcal{B})$ and $(L', \mathcal{B}')$ encode the same structure if and only if there is a group action g in 3D, $g := (R, t) \in \mathrm{SE}(3)$ such that, $X(L', \mathcal{B}') = g \circ X(L, \mathcal{B})$.

A periodic crystal admits an *asymmetric unit* $\mathcal{A}$, the minimal subset that recovers the entire unit cell by applying symmetry transformations of the crystal's space group. Conversely, a supercell can be obtained by an integer matrix $U \in \mathbb{Z}^{3\times3}$ with $m := |\det U| \geq 1$, yielding $\mathcal{C}^{(U)} = (LU, \mathcal{B}^{(U)})$, the same infinite crystal in a differing tiling. For example, $U = mI_3$ yields a cubic $m \times m \times m$ supercell. A formal summary of crystal symmetries is outlined below.

Crystal representation invariances

(S1) *Global translation:*

$$\text{For } t \in \mathbb{T}^3, (L, \{(z_i, u_i)\}) \mapsto (L, \{(z_i, u_i + t)\}).$$

(S2) *Global rotation:*

$$\text{For } R \in \mathrm{SO}(3), (L, \{(z_i, u_i)\}) \mapsto (RL, \{(z_i, u_i)\}).$$

(S3) *Permutation (reindexing):*

$$\text{For } \zeta \in S_N, (L, \{(z_i, u_i)\}) \mapsto (L, \{(z_{\zeta(i)}, x_{\zeta(i)})\}).$$

(S4) *Unit-cell change:* Let $U \in \mathbb{Z}^{3\times3}$ with $m := |\det U| \geq 1$.

- *Unimodular basis change* ($m = 1$, $U \in \mathrm{GL}(3, \mathbb{Z})$):

  $$(L, \{(z_i, u_i)\}) \mapsto (LU, \{(z_i, U^{-1}u_i)\}), \quad (LU)(n+U^{-1}u_i) = L(Un+u_i).$$

- *Supercell expansion* ($m > 1$): Let $\mathcal{R}(U) \subset \mathbb{T}^3$ be a fixed set of coset representatives for $\mathbb{Z}^3/U\mathbb{Z}^3$ (so $|\mathcal{R}(U)| = m$). Then

  $$(L, \{(z_i, u_i)\}) \mapsto \Big(LU, \; \{(z_i, U^{-1}(u_i+r)) : \; i = 1, \ldots, N, \; r \in \mathcal{R}(U)\}\Big),$$

  since $(LU)(n + U^{-1}(u_i + r)) = L(Un + u_i + r)$.

Molecular crystallization. Let $g = \{g_k = (V_k, E_k)\}_{k=1}^{Z}$ denote the set of molecular graph(s) and $\mathcal{X}(g)$ the set of periodic, all-atom crystal structures compatible with g. Due to the invariances, the physically distinct configurations form a quotient

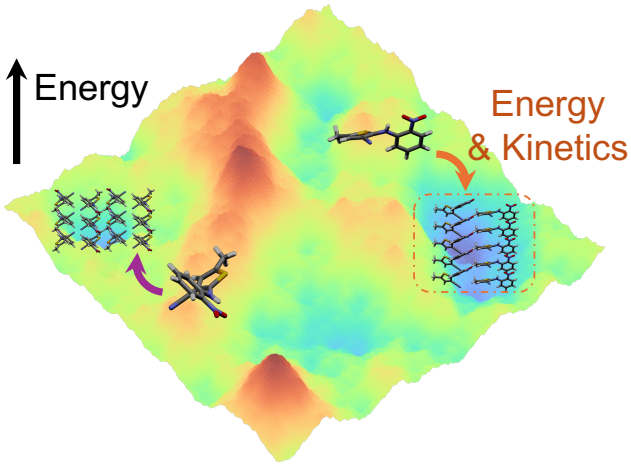


Figure 5.3: Schematic Gibbs free energy landscape for molecular crystallization, showing multiple local minima corresponding to experimentally realizable polymorphs.

space $\mathcal{M}(g) = \mathcal{X}(g)/\sim$. *Ab initio* CSP can then be posed as conditional probabilistic inference over equivalence classes $[\,\mathcal{C}\,] \equiv [\,(L, \mathcal{B})\,] \in \mathcal{M}(g)$. An approximation of the experimentally realized distribution under crystallization conditions $\aleph$ is:

$$p_{\aleph}\Big([\,\mathcal{C}\,] \mid g\Big) \;\propto\; \kappa_{\aleph}\Big([\,\mathcal{C}\,] \mid g\Big)\, \exp\Big(-\beta\, \Delta G([\,\mathcal{C}\,])\Big), \qquad \beta = \tfrac{1}{k_{\mathrm{B}} T} \tag{5.2}$$

where ΔG is the Gibbs free energy (thermodynamics), $\kappa_{\aleph}$ summarizes kinetic accessibility (nucleation and growth pathways), k_B is the Boltzmann constant, and T is temperature.

In practice, learning and sampling must respect the symmetry of $\mathcal{M}(g)$, handle strong coupling between intramolecular conformation and intermolecular packing (especially in flexible molecules), navigate rugged kinetic and energy landscapes arising from large unit cells (often > 100 atoms) and weak and long-range interactions, and marginalize over unknown Z (i.e. number of molecule copies in a unit cell).

These challenges are distinct from most inorganic crystals, which are often characterized by strong covalent or ionic bonds, contain < 30 atoms per unit cell, and lack molecular conformers (Figure 5.2).

5.4 The OXtal Model

We now describe OXtal, our all-atom diffusion model for molecular CSP. As outlined in Section 5.2, existing inorganic CSP models [49, 162] that rely on equivariant architectures and explicit unit-cell parametrizations face scalability challenges for large molecular crystals with unknown multiplicity Z. We thus introduce S^4 training, which exposes the model to long-range periodic cues without

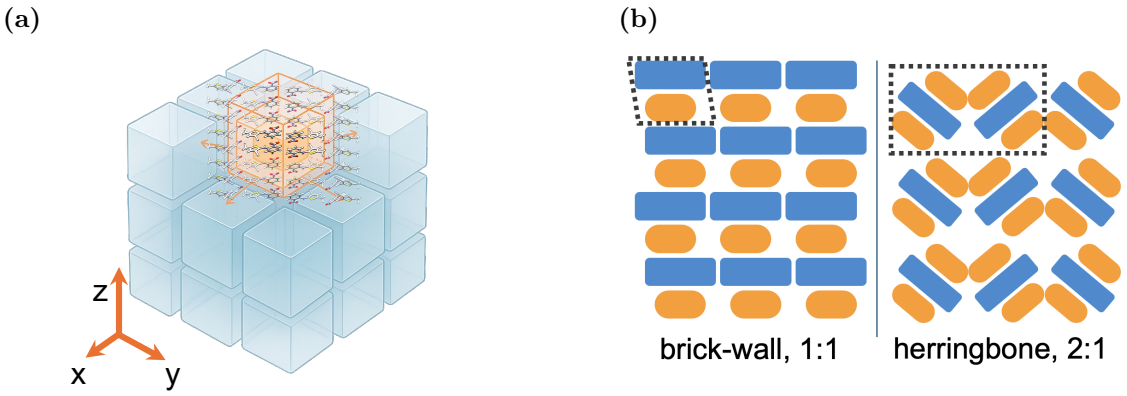


Figure 5.4: (a) Molecular crystallization, showing *nucleation* and *growth* in successive layers, which is the inspiration for S^4. (b) Common packing motifs exemplified in co-crystal polymorphs with 1:1 and 2:1 stoichiometric ratio.

ever parametrizing a lattice (Section 5.4.1). Second, we implement a high-capacity, non-equivariant Transformer with data augmentation and strong molecular embeddings to capture symmetries (Section 5.4.2). Together, these choices decouple what to generate (conformations and packing) from how the crystal is represented during training (unit cell, Z, etc.).

5.4.1 Stoichiometric Stochastic Shell Sampling (S^4)

Crystallization is a local-to-global process: once molecules approach contact distances, weak but specific interactions induce recurring motifs that propagate periodically. Learning to denoise such local consistent neighborhoods should therefore recover larger periodicity at inference time. Training on such subsampled blocks reduces token size, provides natural augmentation, and mirrors the partial observability of nucleation and growth (Figure 5.4(a)).

We formalize this idea in S^4. Let $\mathcal{C}^{(U)} = (LU, \mathcal{B}^{(U)})$ be a supercell with molecules $m \in \mathcal{M}$, where $X(m) \subset \mathbb{R}^3$ denotes m's non-hydrogen Cartesian coordinates. We next define the *minimum-image* intermolecular distance between two molecules, $d_{\min}(m, m') = \min_{x \in X(m),\, x' \in X(m')} \|x - x'\|_2$.

Given a fixed contact radius r_{cut}, S^4 builds concentric shells $\mathcal{S}$ around a uniformly-sampled central molecule $m_c \sim \text{Uniform}(\mathcal{A})$ based on the molecular contact graph induced by $d_{\min}(\cdot,\cdot) \le r_{\text{cut}}$:

$$\mathcal{S}_k(m_c) = \{m_i \in \mathcal{M} : (k) r_{\text{cut}} \le d_{min}(m_c, m_i) < (k+1) r_{\text{cut}}\}, \quad k = 0, 1, 2, \ldots \tag{5.3}$$

We sample the number of shells $K \sim \text{Uniform}(0, k_{max})$, and define the block of molecules $V_K = \cup_{j=0}^{K} S_j$. We cap block size by a token budget $T_{\max}$ (atoms). If

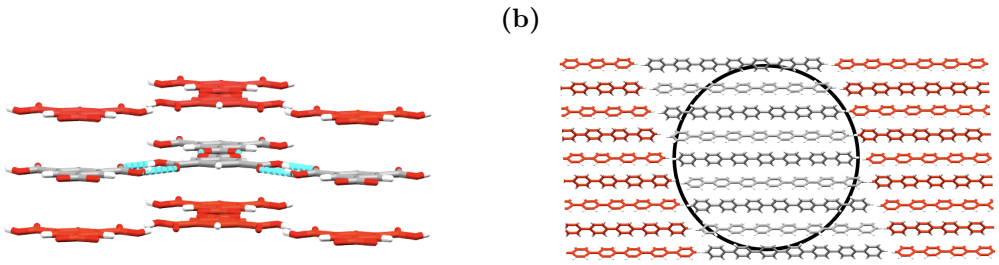


Figure 5.5: (a) kNN cropping as used in AlphaFold3 captures only the closest interactions (e.g. hydrogen bonds in trimesic acid, highlighted in blue) but does not capture more distant interactions that are equally crucial for crystallization (e.g. π-π stacking in trimesic acid, highlighted in red). (b) centroid-based approaches will create anisotropic crops in elongated molecules, for example only capturing a 1-dimensional column (grey) in the ellipsoid p-quinquephenyl, while missing peripheral interactions (red).

$|V_K| \leq T_{\text{max}}$ we accept V_K; otherwise we choose the smallest $K^\star$ with $|V_{K^\star+1}| > T_{\text{max}}$ and subsample the frontier $S_{K^\star}$ to meet the budget while preserving the crystal's molecular stoichiometry. For example, if molecule type i appears N_i^{ASU} times in $\mathcal{A}$ and N_i times in $S_{K^\star}$, we sample molecules of type i in $S_{K^\star}$ with weight $\omega_i \propto (N_i^{\text{ASU}}/N_i)$. The resulting set is our training crop, denoted $\mathbf{A}_{\text{crop}}$.

Compared to kNN- or centroid-based heuristics, this "shell cropping" better respects local interaction networks by respecting anisotropic packing motifs and interactions beyond the strongest ones (Figure 5.5), while mitigating truncation biases common in large-graph learning [176]. The result is a scalable training scheme that balances efficiency with physical fidelity, while leaving open the possibility of future distributed inference. We see in Section 5.5.4 that training with S^4 indeed outperforms kNN and centroid cropping methods empirically.

We provide the full algorithm for S^4 cropping in Algorithm 1. Recall that for a crystal $\mathcal{C}$, $\mathcal{M}$ denotes the set of molecules within the crystal, $\mathbf{A}$ denotes the tokenized atom array for all molecules $m \in \mathcal{M}$, and $X \subset \mathbb{R}^3$ denotes the Cartesian coordinates of each atom $a \in \mathbf{A}$. Note that for practical purposes, we assume $\mathcal{M}$ has a finite size. Additionally, $\mathcal{A}$ is the crystal's minimal *asymmetric unit*, and d is a pre-computed intermolecular distance matrix, where $d(i,j) = \min_{x_i \in X(m_i), x_j \in X(m_j)} ||x_i - x_j||_2$.

Given a specified shell radius r_{cut}, we first sample a central molecule $m_c \sim \text{Uniform}(\mathcal{A})$. Then, we assign all other molecules $m_i \in \mathcal{M} \setminus m_c$ to a shell layer, as defined by r_{cut}. Note that each molecule m_i can only belong to one shell layer. Next, with probability $(1 - p_{\text{max}})$, we randomly sample how many shells to keep. We then add complete shells of molecules to our selected set until the maximum token budget T_{max} is reached.

Algorithm 1 Stoichiometric Stochastic Shell Sampling

Input: $\mathcal{M}$, $\mathbf{A}$, $\mathcal{A}$, d, r_{cut}, $T_{\max}$, $p_{\max}$
$m_c \sim \text{Uniform}(\mathcal{A})$ ▷ Sample central molecule
$\mathcal{M} \leftarrow \mathcal{M} \setminus \{m_c\}$
Initialize $\mathcal{S} = \emptyset$, $k \leftarrow 1$
while $\mathcal{M} \neq \emptyset$ **do** ▷ Compute shell layers around m_c
    $S_k \leftarrow \{m_i \in \mathcal{M} : d(m_c, m_i) \leq k \cdot r_{\text{cut}}\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup S_k$, $\mathcal{M} \leftarrow \mathcal{M} \setminus S_k$, $k \leftarrow k + 1$
end while

$b_{\max} \sim \text{Bernoulli}(p_{\max})$ ▷ Sample maximum number of shells $S_{k_{\max}}$ to keep
if $b_{\max} = \text{False}$ **then**
    $k_{\max} \sim \text{Uniform}\{1, \ldots, k-1\}$
    $\mathcal{S} \leftarrow \{S_1, \ldots, S_{k_{\max}}\}$
end if

Initialize $\mathbf{A}_{\text{crop}} \leftarrow \mathbf{A}(m_c)$, $i \leftarrow 1$,
while $|\mathbf{A}_{\text{crop}}| \leq T_{\max}$ **do** ▷ Add full integer shells within token budget $T_{\max}$
    if $|\mathbf{A}_{\text{crop}}| + |\mathbf{A}(S_i)| > T_{\max}$ **then**
        break
    end if
    $\mathbf{A}_{\text{crop}} \leftarrow \mathbf{A}_{\text{crop}} \cup \mathbf{A}(S_i)$, $i \leftarrow i + 1$
end while

if $i = 1$ **then** ▷ If no full shell fits, sample molecules according to stoichiometry
    $\mathbf{A}_{\text{crop}} \leftarrow$ Adaptive Stoichiometric Sampling($\mathbf{A}$, m_c, $\mathcal{A}$, $T_{\max}$, S_1)
end if
return $\mathbf{A}_{\text{crop}}$

If no complete shell can fit within the token budget, we adaptively sample molecules in the first shell according to Algorithm 2 in order to preserve the stoichiometric ratios of the molecules in $\mathcal{A}$. Lastly, we note that it is possible to deduce the underlying Bravais lattice of a large periodic point cloud by analyzing the set of interatomic difference vectors (a Patterson analysis): in the resulting Patterson function, lattice translation vectors appear as a high-multiplicity, regularly spaced subset of the interatomic vectors, from which the lattice parameters (and hence a primitive cell) can be recovered.

Algorithm 2 Adaptive Stoichiometric Sampling

Input: $\mathbf{A}$, m_c, $\mathcal{A}$, $T_{\max}$, S
Calculate proportion of each molecule type $p_t = \frac{|\{m_i \in \mathcal{A} : \text{type}(m_i) = t\}|}{|\mathcal{A}|}$
Initialize ideal targets: $R_t \leftarrow \text{round}(p_t \cdot |S|)$ for each type t
$\mathbf{A}_{\text{crop}} \leftarrow \mathbf{A}(m_c)$, $R_{\text{type}(m_c)} \leftarrow R_{\text{type}(m_c)} - 1$ ▷ m_c is already pre-selected

while $|\mathbf{A}_{\text{crop}}| < T_{\max}$ **do**
    Bucket remaining molecules by type: $B_t = \{m \in S : \text{type}(m) = t\}$
    Compute adaptive weights for each type:
    for all $t \in T$ with $B_t \neq \emptyset$ **do**
        $w_t \leftarrow R_t / |B_t|$
    end for
    Normalize weights to probabilities: $P'_t = w_t / \sum_t w_t$
    Select type t' randomly according to probabilities P'_t
    Sample $m' \sim \text{Uniform}(B_{t'})$
    if $|\mathbf{A}_{\text{crop}}| + |\mathbf{A}(m')| > T_{\max}$ **then**
        break ▷ Stop if adding this molecule exceeds token budget
    end if
    $\mathbf{A}_{\text{crop}} \leftarrow \mathbf{A}_{\text{crop}} \cup \mathbf{A}(m')$
    $S \leftarrow S \setminus \{m'\}, B_{t'} \leftarrow B_{t'} \setminus \{m'\}$, $R_{t'} \leftarrow R_{t'} - 1$
end while
return $\mathbf{A}_{\text{crop}}$

Finally, we bound the error due to cropping. In the next proposition, we show that the error in the loss due to cropping with S^4 decreases with the cube root of the number of tokens.

Proposition 5.4.1. *Let $\partial\mathbf{A}_{crop} = \{\{u, v\} \in E : u \in \mathbf{A}_{crop}, v \notin \mathbf{A}_{crop}\}$ represent the boundary of $\mathbf{A}_{crop}$. Denote the number of atoms in a volume C as $T(C)$. Let $L_\partial(\mathbf{A}_{crop}) = \sum_{\{u,v\} \in \partial\mathbf{A}_{crop}} \mathcal{L}(u, v)$ be the boundary loss. Assuming $\exists r^0$ s.t. $L(u, v) = 0, \forall u, v$ s.t. $\|u - v\| > r^0$, i.e. $\mathcal{L}$ is local, and there exist $0 < a \leq b < \infty$ s.t. $a|S| \leq T(S) \leq b|S|$ for any $S \subseteq V$. Then,*

$$\frac{L_\partial(\mathbf{A}_{crop})}{T(\mathbf{A}_{crop})} = O((1 + r_{cut})T(\mathbf{A}_{crop})^{-1/3}) \tag{5.4}$$

To prove this proposition, we first label our assumptions.

A1 **Locality.** $\exists r_0$ s.t. $L(u, v) = 0$ for all u, v s.t. $\|u - v\| > r_0$

A2 **Uniform Density.** there exist $0 < a \leq b < \infty$ s.t. $a|S| \leq T(S) \leq b|S|$ for any $S \subseteq V$.

Next we investigate the S^4 algorithm. We first recall the definition of shells and $\mathbf{A}_{\text{crop}}$:

$$\mathcal{S}_k(g_c) = \{g_i \in \mathcal{C}^{(U)} : kr_{\text{cut}} \leq d(g_c, g_i) < (k+1)r_{\text{cut}}\}, \quad k = 0, 1, 2, \ldots \tag{5.5}$$

where d is the distance between molecules g_c and g_i in the infinitely repeating lattice, and r_{cut} is some predefined contact radius. Molecules from each subsequent full shell k are added until $|V_{k+1}| > T_{\max}$, where $T_{\max}$ is the crop budget. If at least one full shell fits, we train on the crop $\mathbf{A}_{\text{crop}} = \{g_c\} \cup \bigcup_{k \in \mathcal{K}} \mathcal{S}_k$. This implies that $\mathbf{A}_{\text{crop}}$ can be well approximated by a ball centered around g_c of radius $(k+1)r_{\text{cut}}$.

We define the ball of radius $r_k = r_{\text{cut}}k$ as $\mathcal{B}_k = \bigcup_{k \in \mathcal{K}} \mathcal{S}_k$. First, we present a short lemma to show the number of neighbors of a node is bounded.

Lemma 5.4.2. *We define the* ***edge neighborhood*** *of g as $\mathcal{N}(g) = \{\{u, v\}$ s.t. $u = g \cap \mathcal{L}(u, v) \geq 0\}$. Assuming [A1] and [A2], we have a uniform bound on the degree*

$$|\mathcal{N}(g)| \leq C \tag{5.6}$$

for some C independent of g.

Proof. This follows from considering a ball of radius r_k around g. That ball has volume $V = \frac{4}{3}r_k^3$. Using [A2], we have that the token count $T(\mathcal{B}_k(g)) \leq \frac{4b}{3}r_k^3$, and therefore $|\mathcal{N}(g)| < \frac{4b}{3}r_k^3$ which is independent of g. □

We next note that for a regular lattice structure we have the following inequality for the size of the boundary relative to the total volume. Specifically for a lattice we have:

Lemma 5.4.3. *For $0 < a \leq b < \infty$ on a 3D lattice, we have the following relationship between a sphere's surface area and volume: $a|\mathcal{B}_k|^{2/3} \leq |\partial\mathcal{B}_k| \leq b|\mathcal{B}_k|^{2/3}$.*

We note that the constants are due to discretization error and locality. As we approach the continuous limit (i.e. $k \to \infty$), the bounds become tight.

Next, we have to bound how well $\mathbf{A}_{\text{crop}}$ is approximated by a ball of radius k.

Lemma 5.4.4. *For some constant $0 < c < \infty$, $\partial\mathbf{A}_{crop} \leq \partial\mathcal{B}_k + cr_{cut}|\mathcal{B}_k|^{2/3}$.*

Proof. We first note that $\mathcal{B}_k \subseteq \mathbf{A}_{\text{crop}}$ by construction of $\mathbf{A}_{\text{crop}}$. This means that we can consider $\mathbf{A}_{\text{crop}}$ as all the molecules in $\mathcal{B}_k$ plus (possibly) some additional molecules in $\mathcal{S}_k(g_c)$. We can then bound the total surface area as

$$|\partial\mathbf{A}_{\text{crop}}| \leq |\partial\mathcal{B}_k| + |\partial\mathcal{S}_k(g_c)| \tag{5.7}$$

however we know that the surface area of molecules in $\mathcal{S}_k(g_c)$ is bounded by the number of nodes in $\mathcal{S}_k$ and some constant.

$$\begin{aligned}
|\partial\mathcal{S}_k(g_c)| &\leq c|\mathcal{S}_k| \\
&= c((r_{\text{cut}}(k+1))^3 - (r_{\text{cut}}k)^3) \\
&= cr_{\text{cut}}^3(3k^2+3k+1) \\
&\leq cr_{\text{cut}}^3k^2 \\
&\leq cr_{\text{cut}}|\mathcal{B}_k|^{2/3}
\end{aligned}$$

which combined with equation 5.7 proves the lemma. □

We are now ready to prove the main proposition.

Proof. Assuming $\mathcal{L}_\partial(\mathbf{A}_{\text{crop}})$ is based on a sum of pairwise interaction terms i.e.

$$\mathcal{L}_\partial(\mathbf{A}_{\text{crop}}) = \sum_{\{u,v\}\in\partial\mathbf{A}_{\text{crop}}} \mathcal{L}(u,v) \tag{5.8}$$

Lemma 5.4.4 allows us to first analyze $\mathcal{L}_\partial(\mathcal{B}_k)$ then use Lemma 5.4.4 for the final bound.

$$\begin{aligned}
\mathcal{L}_\partial(\mathcal{B}_k) &= \sum_{\{u,v\}\in\partial\mathcal{B}_k} \mathcal{L}(u,v) \\
&\leq c|\partial\mathcal{B}_k| \max_{\{u,v\}\in\partial\mathcal{B}_k} \mathcal{L}(u,v)
\end{aligned}$$

using Lemma 5.4.3,

$$\leq c|\mathcal{B}_k|^{2/3} \max_{\{u,v\}\in\partial\mathcal{B}_k} \mathcal{L}(u,v)$$

Assuming [A1], [A2], and Lemma 5.4.2,

$$\leq c|T(\mathcal{B}_k)|^{2/3}$$

Combining this with Lemma 5.4.4 we have the final result

$$\frac{L_\partial(\mathbf{A}_{\text{crop}})}{T(\mathbf{A}_{\text{crop}})} = O((1+r_{\text{cut}})T(\mathbf{A}_{\text{crop}})^{-1/3})$$

□

As a reminder, this proposition implies that the boundary surface becomes less of an issue as the number of tokens grows.

Implementation Details. In practice, we begin by first considering a $3 \times 3 \times 3$ crystal supercell $\mathcal{C}^{(U)}$, where $U = 3I_3$. For each input crystal, we sample a molecule m_c from the asymmetric unit of the central unit cell. After analysing the distribution of intermolecular distances for all crystals in the training set, we select $r_{\text{cut}} = 4.5$ Å (see Figure 5.6). Furthermore, in order to encourage the selection of larger $\mathbf{A}_{\text{crop}}$ blocks, we set $p_{\max} = 0.8$ and $T_{\max} = 640$. We provide additional hyperparameter ablations of r_{cut} below in Section 5.5.4.

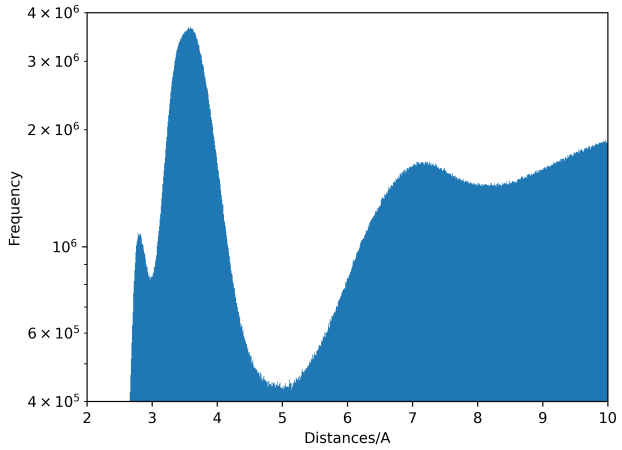


Figure 5.6: Truncated distribution of intermolecular distances in the processed training dataset. We use a cutoff radius of 4.5 Å to define the contact graph for shell growth, but experiment with other shell radius sizes in Section 5.5.4.

5.4.2 Model Architecture

OXTAL is comprised of: (1) an **Atom encoder** that embeds both physical and structural information; (2) a **Pairformer trunk** that propagates information across all atoms in the crop; (3) a **Diffusion module** that takes in the single and pairwise representations and outputs a generated crystal structure. The overall architecture of OXTAL is depicted in Figure 5.7.

Atom Encoder. Given an input SMILES sequence s, we generate a 3D conformer with RDKit ETKDG followed by relaxation by the semi-empirical quantum chemical method GFN2-xTB [177]. The atomic number, positions, formal charges, Mulliken partial charges, and bond information are used as *feature embeddings* for the model. Finally, we resolve ambiguity of identical molecular copies via relative position encoding on entity identifiers [11].

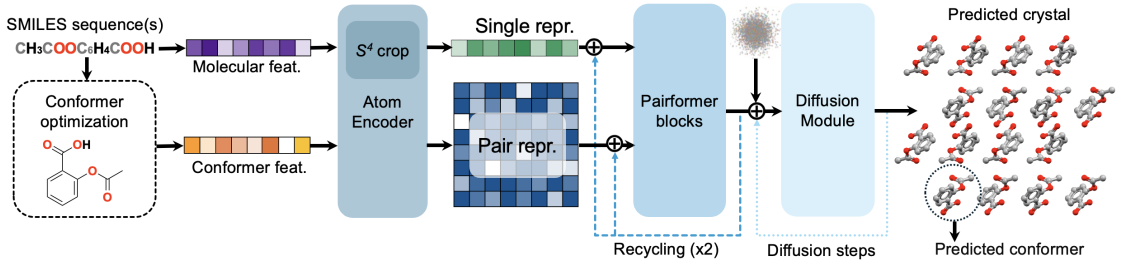


Figure 5.7: Overview of OXTAL architecture.

Pairformer Trunk. We adapt existing state-of-the-art architectures for protein folding to generate single and pair representations for each molecular crystal [11]. Instead of tokenizing protein residues, we simplify the tokenization such that each token directly represents a single atom a_i in the molecule. We then apply the Pairformer stack from AlphaFold3, which leverages triangular self-attention to update the single and pair representations. Unlike AlphaFold2, which relied on the equivariant Evoformer [10] architecture, this simpler Pairformer module is not explicitly equivariant, allowing for training on larger sequences.

Diffusion Module. The design of our diffusion module follows that of AlphaFold3 [11], consisting of an atom attention encoder which combines token information given by the Pairformer with an encoded representation of x_t. This is followed by a large 70M parameter diffusion transformer [71], before a final atom attention decoder predicts the denoised atomic positions. We broadly follow [178] for pre-conditioning of model inputs.

Architecture. Our model adopts the AlphaFold3 (AF3) architecture via the public Protenix implementation [179]. We adopt Algorithm 5 from Abramson et al. [11] to embed the reference conformer into the atom encoder. Atom-level tokens with relative position and entity encodings are processed by an AF3-style *Pairformer* trunk with recycling to produce single and pair representations; the multiple sequence alignment (MSA) and RNA language model (LM) pathways are disabled in all reported experiments. These representations condition a structure-denoising head consisting of an atom-attention encoder, a transformer-based diffusion module, and an atom-attention decoder that outputs 3D coordinates. We also provide a diffusion module size ablation in Section 5.5.4.

5.4.3 Training

OXTAL is trained using a procedure similar to those successfully employed for protein structure prediction [11]. The training objective is a composite loss designed to capture both the global structure and the accuracy of the local chemical environment. This loss is comprised of two main components: (1) a mean squared error loss $\mathcal{L}_{\text{mse}}$, (2) a smooth local difference distance test $\mathcal{L}_{\text{sLDDT}}$ as defined in Abramson et al. [11]. Both losses compare the predicted structure $\hat{x}_0 := D_\theta(x_t, t)$ and an aligned ground truth structure $x_0^{\text{align}} = \texttt{align}(x_0, \hat{x}_0)$, and the latter emphasizes the pairwise interactions within the crop via a surrogate of the interatomic distances. To round off our training, we also include a distogram loss on a separate head branching from the trunk $\mathcal{L}_{\text{dist}}(\hat{d}, d)$ to ensure the trunk output contains binned pairwise distance information. The final loss is then a weighted sum of these components:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1), x_t\sim p_t\left(x_t|x_0^{\text{align}}\right)}\left[\mathcal{L}_{\text{mse}}(\hat{x}_0, x_0^{\text{align}}) + \mathcal{L}_{\text{sLDDT}}(\hat{x}_0, x_0^{\text{align}})\right] + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}(\hat{d}, d).$$

We use a combination of losses to encourage accurate structure prediction on both the global and local scales. Below, we provide a detailed description of the SmoothLDDTLoss and distogram loss that is taken from the literature.

SmoothLDDTLoss. The smooth local distance difference test (SMOOTHLDDT) loss is a smooth form of an existing LDDT loss [180]. The LDDT measures how well two structures align over all pairs of atoms at a distance closer than some predefined threshold. For us, this is $R_0 = 15$Å. These pairs of atoms form a set of local distances L. The LDDT score is the average of fractions at four distance thresholds: [0.5, 1.0, 2.0, 4.0] Angstroms. The Smooth LDDT test, instead of using a binary test of whether or not the local distances are on the same side of the threshold, uses a sigmoid function. Let

$$\begin{aligned} L &= \|x - x^T\| \\ L^{GT} &= \|x^{GT} - (x^{GT})^T\| \\ \delta &= \text{abs}(L - L^{GT}) \\ \epsilon &= \frac{1}{4}\left[\text{sigmoid}(\frac{1}{2} - \delta) + \text{sigmoid}(1 - \delta) + \text{sigmoid}(2 - \delta) + \text{sigmoid}(4 - \delta)\right] \\ \text{mask} &= \delta < 15 \end{aligned}$$

Then the SMOOTHLDDT between two molecules of size d is:

$$\text{SMOOTHLDDT}(x, x^{GT}) = \sum(\epsilon \cdot \text{mask})/(d(d-1)) \tag{5.9}$$

Our SmoothLDDT loss is then equal to

$$\mathcal{L}_{\text{sLDDT}}(x, x^{GT}) = \text{SMOOTHLDDT}(x, \text{ALIGN}(x^{\text{GT}}, x)) \tag{5.10}$$

where $\text{ALIGN}(a, b)$ performs an optimal rigid alignment of a to b in 3D.

Distogram Loss. The distogram loss is used to supervise the prediction of pairwise inter-atomic distances by framing distance prediction as a classification problem. For an atom array of length L, a distogram is an $L \times L$ matrix in which each entry corresponds to the distance d_{ij} between atoms i and j, discretised into K distance bins.

Given a predicted probability distribution $p_{ij} \in \mathbb{R}^K$ over distance bins and a one-hot encoded ground-truth label y_{ij}, the distogram loss is defined as the categorical cross-entropy:

$$\mathcal{L}_{\text{dist}} = -\frac{1}{L^2} \sum_{i,j} \sum_{k=1}^{K} y_{ij,k} \log p_{ij,k}. \tag{5.11}$$

This formulation avoids direct regression of continuous distances, improving training stability and allowing the model to represent uncertainty over spatial relationships.

Training Data Processing. We next curate training data from the Cambridge Structural Database (CSD, including releases up to May 1, 2025) [36]. With over 1.3 million entries total, the CSD is the world's largest curated dataset of experimentally determined crystal structures. It includes structures derived from X-ray, neutron, and electron diffraction analyses, covering a wide range of organic and metal-organic compounds. The database is widely used in the pharmaceutical, agrochemical, and fine chemicals industries for various different research and development initiatives. To build our training dataset, we filter the CSD under the following criteria:

1. 3D coordinates are present
2. The conventional R-factor $< 9\%$
3. The structure is derived from single crystal diffraction at ambient pressure
4. The structure is not polymeric
5. The structure can be sanitized by RDKit with no missing heavy atoms
6. There is a known space group
7. At most 250 non-hydrogen atoms are present per unit cell

To avoid leakage, we exclude any entry belonging to a test crystal family and any entry containing a molecular component that appears in test sets. To avoid oversampling certain clusters within each crystal family, near-duplicate polymorphs are collapsed using $\text{RMSD}_{15} \leq 0.25$ Å, retaining the entry with the lowest R. Our final training dataset contains 594,202 crystals in total.

For data preparation, crystallographic disorder is resolved by selecting the disorder group with the highest occupancy. We remove hydrogens for training and extract kekulized bonds from crystallographic metadata. Prior to building supercells, molecules of the crystals are centered to lie within the unit cell, and the unit cells are transformed to their unique Niggli-reduced (i.e. most primitive) forms. Supercells are constructed by tile translation $\mathbf{T}_{ijk} = i\mathbf{a} + j\mathbf{b} + k\mathbf{c}$ for $i, j, k \in -1, 0, 1$ such that molecules with centroids inside the supercell boundary are included.

5.5 Experimental Evaluation

We evaluate OXtal on several different datasets for molecular CSP, using a diverse set of metrics to probe different aspects of generation quality (Section 5.5.1). We begin by first comparing OXtal against a range of ML *ab initio* baselines on a set of both rigid and flexible molecules (Section 5.5.2). We then evaluate OXtal against several traditional energy-based methods on the 5th, 6th, and 7th CSP blind tests (Section 5.5.3). Finally, we supplement these results with additional ablation, inference, and chemical analyses in Sections 5.5.4 to 5.5.6, respectively. Additional experimental details are provided in Appendix C.

5.5.1 Evaluation Protocol

We adopt standard CSP evaluation metrics and report both *sample-* and *crystal-level* scores for all experiments:

1. *Collision rate* (Col_S): Fraction of generated samples with any intermolecular distance $< r_w - 0.7$ Å where r_w is the sum of atomic van der Waals radii [181]. Collisions suggest physically infeasible interactions and should be minimized. We use 0.7 Å since the strongest intermolecular interactions are $\sim r_w - 0.6$ Å.

2. *Packing similarity rate* (Pac_S and Pac_C): Using CSD COMPACK, a sample *matches* if at least 8 of 15 molecules can be aligned to the experimental cluster (cluster size as defined per CSP5 [37]). Pac_S averages over all samples and Pac_C is the fraction of crystal targets with at least one match.

3. *Conformer recovery* (Rec_S and Rec_C): $\mathrm{RMSD}_1 < 0.5\,\text{Å}$ (non-hydrogen) to a solid-state conformer. $\mathrm{Rec_S}$ averages over all samples and $\mathrm{Rec_C}$ is the fraction of crystal targets with at least one match.

4. *Approximately solved* ($\widetilde{\mathrm{Sol}}_C$): *Any* collision-free, packing-similar sample with $\mathrm{RMSD}_{15} < 2\,\text{Å}$ on a 15-molecule cluster.

These four metrics jointly probe physical plausibility (Col_S), packing similarity (Pac_S, Pac_C), intramolecular fidelity (Rec_S, Rec_C), and a holistic approximate solve indicator ($\widetilde{\mathrm{Sol}}_C$). Reporting both sample-level and crystal-level aggregates is essential: the former characterizes the distributional quality of all proposals, while the latter answers whether a target was recovered at least once. Concretely, crystal-level rates are an OR-aggregation over a target's samples. Evaluating both crystal and sample level metrics together helps diagnose failure modes such as mode collapse (high Pac_C with low Pac_S) and low recall (the reverse).

Thresholds and Interpretability. We adopt community conventions for evaluation of our RMSD_k thresholds [182]:

- $\mathrm{RMSD}_{15} < 1.0\,\text{Å}$ typically indicates the predicted packing reproduces the experimental match and lies in the exact energy basin as the experimental structure. This metric is the one used by the 5th CSP blind test [37].

- $1.0-2.0\,\text{Å}$ usually indicates the prediction shares the correct topology with mild lattice strain or small reorientations. If the H-bond graph, Z/Z', density, and PXRD (powder x-ray diffraction) are consistent, this is very likely recoverable to < 1 Å with a brief local relaxation. Otherwise, the structure is often structurally related to the ground truth (i.e. in or near the correct packing motif/topology or space group but with slip, molecular misorientation, or a cell mismatch). It demonstrates that the method can correctly identify the neighborhood of the global energy minimum, even if it can't pinpoint the exact minimum itself.

- $2.0-3.0\,\text{Å}$ typically indicate a significant mismatch. At this level of deviation, key intermolecular interactions, like hydrogen bonds, may be incorrect. The overall packing symmetry might also be different. However, the prediction might still capture a general feature of the packing (e.g., identifying a layered structure vs. a herringbone packing).

- above 3.0 Å, these values are generally not considered useful. The deviation is so large that the predicted structure is almost certainly in a completely different and incorrect energy basin. Any similarity to the experimental structure is likely coincidental.

Our $\widetilde{\text{Sol}}_C$ (collision-free, packing-similar, $\text{RMSD}_{15} < 2\,\text{Å}$) metric is a thus a strict, early-stage indicator: not completely "solved," but reliably near-correct for downstream relaxation and re-ranking.

Packing Similarity for Non-Periodic Predictions. Our generators output finite blocks without periodic conditions. For large inference blocks, it is possible to identify translation vectors that map the finite cluster onto itself (e.g., by correlating molecular centroids or local environments), least-squares fit three independent vectors to define a primitive lattice, and convert atoms to fractional coordinates. Nevertheless, CSD *COMPACK* can be directly used for robust alignment of a block against a crystal structure while preserving standard CSP rigor. First, we employ standard practices of avoiding hydrogen-atom alignment and allowing mismatches in connectivity annotations (e.g., bond orders). In the absence of periodic images, greedy pruning can miss valid correspondences; therefore for all methods (including DFT baselines) we use a search time of 10 seconds with a distance threshold of 50% and angle threshold of 75 degrees. This enables reliable COMPACK outputs on non-periodic inputs *without diluting the criterion*: acceptance still hinges on multi-molecule superposition (15-molecule cluster, ≥ 8 matched) and the same RMSD_k thresholds used in periodic CSP benchmarks.

5.5.2 Molecular CSP on Rigid and Flexible Datasets

First, we compare OXtal to *ab initio* ML models on two different test sets comprised of representative rigid and flexible molecules with ground-truth structures in CSD. Specific dataset and baseline details are provided below.

Dataset Setup. We construct our rigid and flexible datasets using crystals from the Cambridge Structural Database (CSD), following the processing procedure outlined in Section 5.4.3. The rigid dataset is comprised of 50 molecular crystals, and the flexible dataset is comprised of 16 molecular crystals. Many of these crystals were selected because they appeared in previous CSP blind test competitions (i.e. CSP blind tests 1-4) [183–186], or because they are practically relevant to industrial applications. We also select a number of newly-released crystals to evaluate the

generalizability of our model to recent data. We generally define rigid molecules to contain 0-3 rotatable bonds, and flexible molecules to contain more than that. We note there is a small nuance in flexibility, as we refine it in the molecular context (by ring restriction or number of known polymorphs, etc.), this means crystals such as rubrene (QQQCIG), tetraphenylporphryin (TPHPOR), CL-20 (PUBMUU) are defined as "rigid;" ROY (QAXMEH), galunisertib (DORDUM), sulfathizaole (SUTHAZ), flufenamic acid (FPAMCA) and YOPYEL are defined as "flexible." Exact CSD identifies for each dataset are listed in Appendix C.1.

***Ab initio* Machine Learning Baselines.** Most ML methods for inorganic CSP are incompatible for molecular CSP due to the differences outlined in Section 5.3. Therefore, we compare OXtal against two *ab initio* machine learning baselines: (i) A-Transformer, a lightweight all-atom transformer trained with flow matching on our training dataset, (ii) the publicly released AssembleFlow-Atom model [187].

A-Transformer Model. A-Transformer is a lightweight transformer encoder operating on atom tokens with unit-cell features. Each atom embedding includes element and charge information, time $t \in (0, 1]$, molecular fingerprints, and, when enabled, unit-cell lengths (a, b, c) and angles (α, β, γ). Two output heads are used: a coordinate head ($\mathbb{R}^3$ per token) and, optionally, a rotation head (unit quaternion, disabled in our main runs).

The objective is rigid-cluster translation flow matching in $\mathbb{R}^3$. We linearly interpolate between ground-truth translations s_0 and random in-cell starts s_1,

$$s_t = (1 - t)\, s_0 + t\, s_1, \quad t \sim \mathrm{Unif}[t_{\min}, 1],$$

and predict denoised coordinates $\hat{x}_0$. The loss is an x_0 regression term:

$$\mathcal{L} = \lambda_{\mathrm{trans}} \sum_i \|x_0^{(i)} - \hat{x}_0^{(i)}\|^2.$$

At inference we integrate from t=1 to $t_{\min}$ with Euler steps, producing clusters in a P1 box. This predicted unit cell is then expanded for supercell evalutaion. This baseline is deliberately minimal: it ignores rotation flow matching and lattice prediction, relying solely on translation flow matching and unit-cell features. It provides a capacity-matched reference against which to measure the benefits of OXtal's architecture and training. See Appendix C.2.1 for specific hyperparameters.

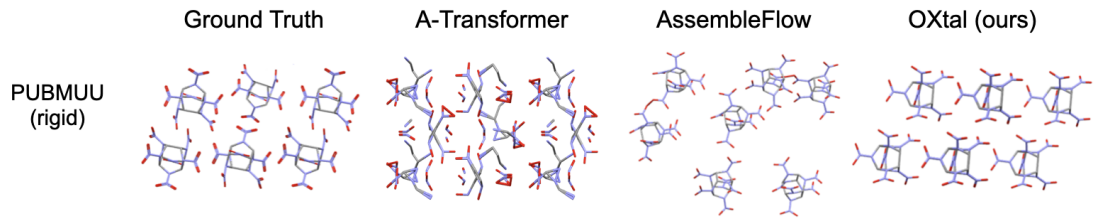


Figure 5.8: Example crystal packings generated by A-Transformer, AssembleFlow, and OXtal compared to the ground truth (left).

AssembleFlow. We evaluate the newly released AssembleFlow-Atom model [187] using the authors' checkpoints without re-training or fine-tuning. For each molecule we generate clusters of 15 rigid copies from an RDKit ETKDG conformer, applying random rotations and translations with uniform offsets $[-S, S]^3$, where $S \in \{10, 15, 20\}$ Å. Three seeds $\{42, 7, 2024\}$ and seven checkpoints yield 63 samples per molecule. To standardize evaluation, we randomly sample 30 outputs per crystal and compute metrics on those. AssembleFlow enforces rigid-molecule assembly and provides a strong baseline for rigid-packing quality, but is unable to handle flexible molecules, as reflected in the results presented in Table 5.1.

Experimental Setup. For each model, we generate and evalutae $n_S = 30$ samples for each crystal target. For OXtal and AssembleFlow, we generate a supercell cluster of 30 molecules. Recall that A-Transformer is a unit-cell based model, so we expand the predicted cell to the appropriate size for comparison. All crystals in both rigid and flexible test datasets were excluded from training for both A-Transformer and OXtal. However, we cannot guarantee the public pre-trained version of AssembleFlow did not train on any of these molecules.

Results. OXtal outperforms existing ML methods across all metrics on both datasets. Intramolecularly (Rec_C), OXtal recovers over 90% of solid-state molecular conformers in the rigid set and half of the conformers in the flexible set. Intermolecularly, Col_S is near zero on rigid targets and low on flexible ones. OXtal's predicted packings also attain a strong packing similarity rate against experimental structures, with approximate solves for both rigid and flexible molecules.

Qualitatively (Figure 5.8), A-Transformer struggles to capture meaningful conformers despite being given the number of molecule copies Z, highlighting the limitations of a unit cell based model. AssembleFlow generates molecular assemblies with large spatial separations that lack periodicity (also reflected in low Pac_C and Pac_S scores), along with frequent interatomic clashes. Overall, results for OXtal are generally an order of magnitude improvement on current ML methods, which are essentially unable to approximately resolve *any* crystal packings.

Table 5.1: Performance of *ab initio* machine learning models on both rigid and flexible molecular CSP. OXtal achieves an order of magnitude improvement and is the only model able to approximately solve any crystals in the flexible dataset.

Model	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
	Rigid Dataset					
A-Transformer	0.731	0.015	0.060	0.033	0.120	0.060
AssembleFlow	0.524	0.001	0.040	0.211	0.760	0
OXtal	**0.011**	**0.873**	**1.000**	**0.737**	**0.960**	**0.300**
	Flexible Dataset					
A-Transformer	0.874	0.002	0.063	0	0	0
AssembleFlow	0.850	0	0	0	0	0
OXtal	**0.167**	**0.410**	**0.813**	**0.056**	**0.500**	**0.125**

Zero-shot Evaluation with AlphaFold3. We additionally evaluate AlphaFold3 as a structural generative baseline (Table 5.2). For each target molecule, we generate conformations using default protein–ligand generative settings, treating each conformer as a candidate crystal packing. Inference is performed based on 30 counts of the molecule, and thirty samples per target are evaluated in the same way as for other baselines. While AlphaFold3 was not designed for crystal structure prediction, including it highlights the gap between general-purpose biomolecular structure generators and CSP-specific models.

Table 5.2: Performance of *ab-initio* machine learning models on rigid and flexible molecular CSP. For each model, results are calculated using $n = 30$ samples for each crystal target.

Model	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
	Rigid Dataset					
A-Transformer	0.731	0.015	0.060	0.033	0.120	0.060
AssembleFlow	0.524	0.001	0.040	0.001	0.020	0
AlphaFold3	0.114	0.089	0.340	0.133	0.480	0
OXtal	**0.011**	**0.873**	**1.000**	**0.737**	**0.960**	**0.300**
	Flexible Dataset					
A-Transformer	0.874	0.002	0.063	0	0	0
AssembleFlow	0.850	0	0	0	0	0
AlphaFold3	0.580	0.144	0.313	**0.191**	0.438	0
OXtal	**0.167**	**0.410**	**0.813**	0.056	**0.500**	**0.125**

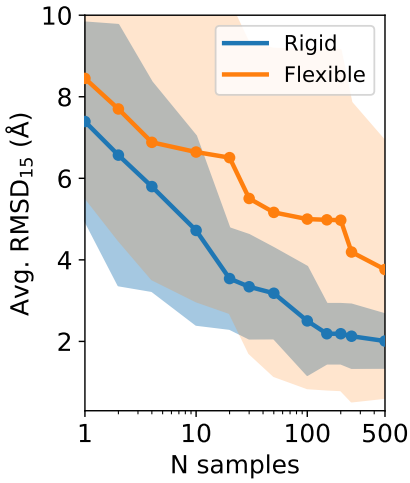


Figure 5.9: OXTAL sample efficiency for 10 rigid & flexible molecules.

Results. From these results, it is clear that protein folding models cannot simply be ported over for solve molecular CSP. Although AlphaFold3 does perform slightly better in recovering individual flexible conformers, it struggles significantly with recovering the periodicity that is essential to crystal packing. This makes sense, given that the model is much larger and is primarily trained on non-periodic protein data.

Sample Efficiency. For downstream screening and design, few-sample success is critical. OXTAL exhibits a log-linear improvement in RMSD_{15} among packing-similar predictions as n increases (Figure 5.9), with several rigid targets reaching Pac_C and $\widetilde{\text{Sol}}_C$ with $n < 10$. This suggests the sampler (i) often lands near the correct motif and (ii) refines global periodicity with additional draws. Flexible systems require more samples on average, but follow the same trend, indicating that OXTAL can eventually discover the correct combination of torsional states and packing contacts.

5.5.3 CCDC CSP Blind Tests

Every few years, CCDC holds a CSP blind test competition, which invites leading computational chemistry groups to solve a handful of hidden crystal structures that have been identified but not yet released [37–39]. We therefore evaluate OXTAL on structures from the three most recent (5th, 6th, and 7th) blind tests.

To contextualize our work, we provide a summary of the 5th, 6th, and 7th CCDC CSP blind tests. These community-wide challenges have benchmarked the state-of-the-art in predicting the crystal structures of organic molecules from their chemical graphs alone, primarily relying on density functional theory (DFT). Here, we provide

brief descriptions of these tests, and defer to the original reports for details [37–39]. Performance is assessed using the COMPACK algorithm, which quantifies structural similarity through root mean square deviation (RMSD) of molecular clusters.

Dataset Setup. In the first four CSP blind tests, the field evolved from force-field landscapes to the first widespread use of periodic dispersion-corrected DFT (DFT-D), which proved decisive in 4th blind test and set the stage for DFT to become the de facto standard for final lattice-energy evaluation.

The 5th blind test [37] marked a significant increase in the complexity of the target molecules. The six targets included rigid molecules, semi-flexible molecules, a 1:1 salt, a highly flexible pharmaceutical-like compound, and two co-crystals. This test highlighted what has become a standard workflow in CSP: broad structure generation (e.g., grid or quasi-random sampling, parallel tempering), followed by local minimization and hierarchical re-ranking. While DFT-D demonstrated its reliability for small and moderately flexible molecules, handling high flexibility and complex solid forms remained a major challenge. A total of 15 groups submitted structures for this blind test, although the report contains only 14 groups.

The 6th blind test [38] continued with five challenging targets: a small, nearly rigid molecule; a polymorphic former drug candidate; a chloride salt hydrate; a co-crystal; and a large, flexible molecule. This test solidifed the "search–optimize–rank" pipeline. On the search side, more than half of the methods allowed intramolecular flexibility during exploration, and many adopted hierarchical filtering starting with generating conformer and packing, followed by progressively tighter optimization and pruning. In optimization and ranking, periodic DFT-D became mainstream, with downstream forcefield potentials used to refine close competitors. The results demonstrated that while DFT-D provided reliable baseline energetics, accurate treatment of conformational flexibility remained the primary bottleneck. A total of 25 groups participated in this blind test.

The 7th blind test [39] introduced a two-phase structure to separate structure generation from ranking challenges. The seven targets featured a silicon-iodine containing molecule, a copper coordination complex, a near-rigid molecule, a co-crystal, a polymorphic small agrochemical, a highly flexible polymorphic drug candidate, and a polymorphic morpholine salt. The test also featured one of the most challenging systems in the history of the blind tests: a large, highly polymorphic pharmaceutical drug candidate. A key finding from the structure generation phase was that while different search methods often identified overlapping sets of low-energy structures, their success rates tended to decrease as the complexity of the target

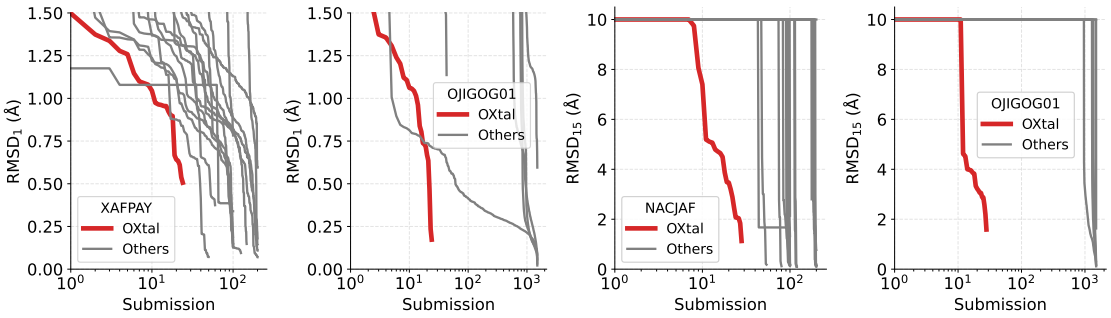


Figure 5.10: OXTAL sample efficiency plots for conformer recovery of XAFPAY (blind test 6, a flexible molecule) and OJIGOG01 (blind test 7) and for $RMSD_{15}$ of NACJAF (blind test 6) and OJIGOG01. OXTAL, for targets shown, achieves similar $RMSD_{1/15}$ with less (%) submissions compared to DFT.

molecule increased. For ranking, periodic DFT-D methods continued to produce results in agreement with experiments across most targets, whereas higher-level corrections, full free-energy treatments, and ML on DFT potentials were less reliable. A total of 22 groups participated in the structure generation phase of this blind test.

Experimental Setup. Metrics and experimental set up for OXTAL and other ML baselines (A-Transformer and AssembleFlow) follow the protocol outlined in Section 5.5.2. For DFT-based submissions, because each submission group can choose which crystals to attempt, we compute an aggregate metric that is averaged across all submitted methods. For sample-level metrics, we first compute each rate separately for every group, then take the mean of these per-group rates across all groups. n_S is the average number of samples per group per target. For crystal-level metrics, we average the rates over each group, and across *all* available targets for that competition (i.e. we consider the total number of crystal targets across all DFT methods to be $n_{\text{groups}} * n_{\text{targets}}$.). We count groups who did not submit any submissions for a crystal as a miss, since they did not show evidence of successfully solving it. Detailed results for each method and target are provided in Appendix C.3 for reference.

Note that in blind test 7 [39], Target XXX has two forms with different stoichiometries (MIVZEA and MIVZIE). For evaluation purposes, *ab initio* machine learning methods attempted 30 structures for each stoichiometry. As such, per-sample metrics for competition 7 are computed with these additional samples, and we consider 8 total targets for per-crystal metrics.

Table 5.3: Results for the 5th, 6th, and 7th CCDC CSP blind tests with the best model **bolded** and second best underlined. Classical chemistry methods aggregated as DFT_{avg}.

Model	n_S	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
	CSP Blind Test 5						
A-Transformer	30	0.833	0	0	0	0	0
AssembleFlow	30	0.717	0	0	0.150	0.500	0
DFT_{avg}	464	0.039	0.323	0.661	**0.784**	0.733	**0.544**
OXtal	30	**0.006**	**0.667**	**0.833**	0.572	**0.833**	0.167
	CSP Blind Test 6						
A-Transformer	30	0.967	0	0	0	0	0
AssembleFlow	30	0.800	0	0	0.073	0.200	0
DFT_{avg}	83	0.067	0.183	0.520	**0.655**	0.560	**0.496**
OXtal	30	**0.013**	**0.660**	**1.000**	0.160	**0.600**	0.200
	CSP Blind Test 7						
A-Transformer	30	0.950	0	0	0	0	0
AssembleFlow	30	0.808	0	0	0.063	0.286	0
DFT_{avg}	868	0.072	0.058	0.511	**0.337**	**0.449**	**0.421**
OXtal	30	**0.021**	**0.483**	**0.875**	0.129	0.375	0.125

Results. Table 5.3 shows that OXtal strongly outperforms *ab initio* ML baselines, and achieves the best or second best performance across all three tests. While DFT methods may sometimes attain higher conformer recovery or approximate solve rates, OXtal consistently scores the best in terms of packing similarity rate. From a per-sample basis, only $5 - 30\%$ of DFT samples match the lattice cluster, whereas OXtal generally recapitulates the packing structure $(48 - 67\%)$. This suggests that while DFT identifies many local energetic minima, OXtal can better capture the joint energy and kinetic features that determine which minima are more likely to be formed. Of the thousands of submitted predictions (and potentially more generated but unsubmitted structures), only a small percentage of DFT-identified minima are close to ground truth structures (Figure 5.10). Predicting the correct experimental structure in few shots is crucial for downstream discovery applications, and OXtal reliably approximates lattice packings *sans* any ranking methods.

These blind tests shows a clear trend that DFT-based ranking has become reliable, but limitations persist: computational expense, difficulty capturing kinetic effects, challenges with disorder, flexibility, high Z number, and the fundamental limitation that thermodynamic ranking often fails to predict which polymorphs

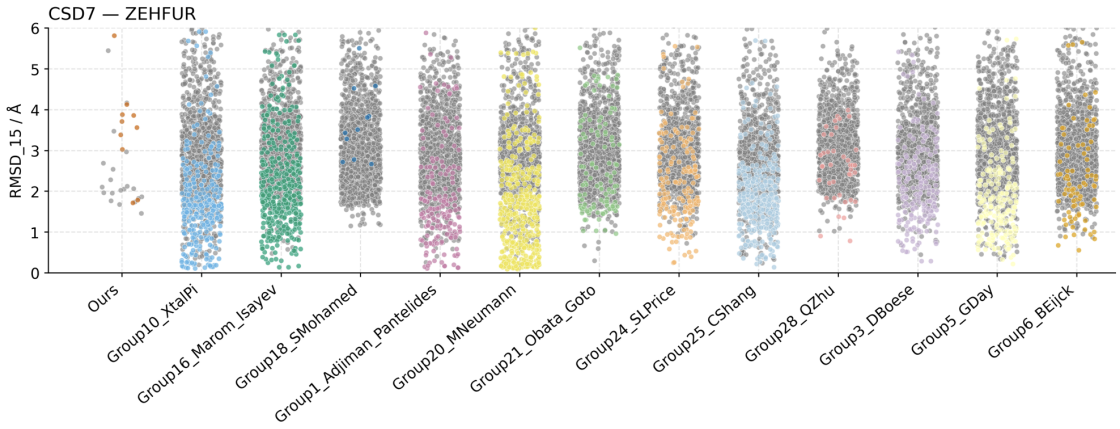


Figure 5.11: Beeswarm plot of submitted structures for Molecule XXXI (ZEHFUR) from CSD7. In 30 inference samples, OXtal obtains performance comparable to some DFT groups with thousands of submitted samples. Colored points signify a partial packing match and gray points signify packing mismatch (e.g. 3 out of 15 molecules match with low RMSD, but packing is significantly different).

are experimentally observable. Our approach aims to address these limitations by directly learning both thermodynamic and kinetic regularities from data, with the hope of eventually eliminating the need for extensive search, local optimization, and post-hoc ranking altogether. By generating a small number of high-quality structures directly, we bypass the traditional generate-optimize-rank pipeline that has dominated CSP. The beeswarm plot (Figure 5.11) of submissions for Target XXXI from blind test 7 (CSD ID: ZEHFUR) further highlights the extensive sampling currently required by DFT methods compared to OXtal.

Inference Cost. As mentioned previously, the major pitfall of DFT-based methods is the extremely high computational cost required to run the atomistic simulations. Recall that CCDC recently raised concerns over the 46 million CPU core hours that were reportedly utilized by submitted methods in CSP7 to solve only 8 crystal targets [39]. Unlike traditional DFT methods that require new simulations for each new molecule, OXtal's upfront training cost is amortized at inference time, allowing us to efficiently generate samples for new molecules. Using standardized on-demand cloud pricing (Section 5.5.3), *OXtal is over an order of magnitude cheaper at inference time*, while still achieving strong packing similarity rates (Figure 5.12). Given the CCDC's call for more efficient CSP algorithms, OXtal's cost profile enables broad screening and dense posterior sampling before any optional physics-based refinement.

We compute computational costs by multiplying the reported wall–clock compute time by an AWS on–demand unit price with hours taken directly from the three blind

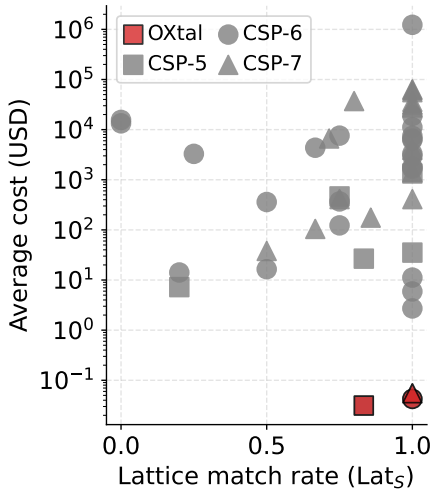
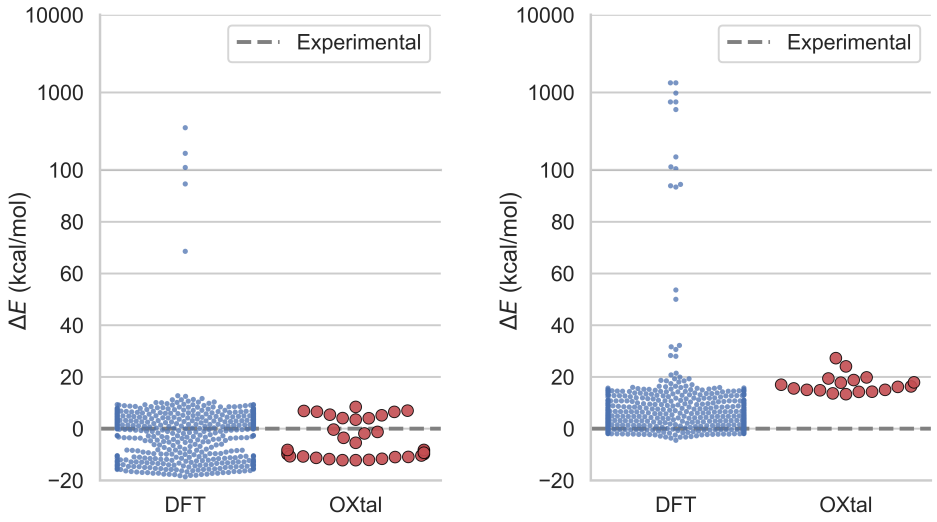


Figure 5.12: Packing similarity rate per crystal attempted relative to average inference cost (in $USD) for submitted CCDC competition methods. OXtal is denoted in red. Costs are normalized to a single on-demand AWS instance from Sept. 2025.

tests references [37–39]. Because the papers report heterogeneous processors and often normalize times to ∼3.0 GHz core–hours, we treat one normalized CPU hour as one AWS vCPU–hour; we then price CPU time at the c5.large on–demand rate in Sept. 2025, i.e., $0.085 per *2* vCPUs ⇒ $0.0425 per vCPU–hour, so CPU cost = hours × $0.0425. For L40S GPU runs, we map directly to `g6e.xlarge` instances (which contains one NVIDIA L40S) and are priced at $1.861 per instance–hour, so GPU cost = hours × $1.861. We apply the rates pro–rata without further rounding. All prices are on–demand EC2 instances as of September 2025. Exact inference times for all methods are reported in Appendix C.3.2 for reference.

Energy Analysis. To assess whether the generated samples contain (i) highly energetically unfavorable motifs or (ii) chemically plausible packing arrangements compared to physics-based methods, we utilized the semi-empirical GFN2-xTB method as a computational proxy. GFN2-xTB was selected for its balance of computational speed and accuracy in capturing non-covalent interactions. This analysis should be interpreted as a *consistency check on gross energetic behaviour* rather than as a quantitative ranking of polymorph stability.

We retrieved 1,500 physics-based structure predictions submitted to the CSP blind tests to serve as a baseline. Both the physics-based submissions and the experimental ground truth were expanded into 2×2×2 supercells to match the approximate atom count of the OXtal generations (30 molecules). For OXtal samples, hydrogens were added using the CCDC API.



(a) OBEQOD (XVII in CSP5) **(b)** NACJAF (XXII in CSP6)

Figure 5.13: GFN2-xTB single point energy analysis of ground truth crystal structures, 1,500 subsampled physics-based blind test submissions, and OXTAL samples.

We performed single-point energy calculations on all structures. The energy difference is relative to the experimental structure and standardized by number of molecules. It is important to note that the DFT baseline consists of structures that underwent varying degrees of geometry optimization, whereas OXTAL samples were evaluated as raw generative outputs with no geometry optimization, including the hydrogen atoms.

The results show that first, OXTAL samples do not contain energetically catastrophic motifs. Second, despite *lacking relaxation*, the generative samples exhibit a narrow energy distribution comparable to the stable basin of the DFT samples. This confirms that the generative model implicitly learns to avoid severe steric clashes and produces metastable packing configurations.

5.5.4 Ablation Studies

Now, we provide a series of different model ablation studies on OXTAL. Due to the high computational demand of training OXTAL, all results in this section are reported at 50k training steps. Following our predefined evaluation protocol, we evaluate $n_S = 30$ generated samples with 30 molecular copies for each crystal target.

Cropping Method Ablation. In order to isolate the performance gain from S^4, we directly compare it against other standard cropping methods in Table 5.4. Overall, the two variations of S^4 cropping perform the best across all metrics and

Table 5.4: OXTAL ablation results with different cropping methods. The best results are **bolded** and second best are underlined.

Crop Method	Dataset	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
kNN	Rigid	0.086	0.631	0.920	0.520	0.780	0.220
	Flexible	0.580	0.164	0.688	0.005	0.062	0
	CSP5	0.085	0.470	0.667	0.433	0.833	0
	CSP6	0.132	0.353	0.800	0.007	0.200	**0.200**
	CSP7	0.312	0.165	0.750	0.005	0.125	0
Centroid Radius	Rigid	0.040	0.669	**0.980**	0.591	**0.940**	**0.340**
	Flexible	0.425	0.301	0.688	0.048	0.375	0.125
	CSP5	0.054	0.470	**0.833**	0.429	0.833	0
	CSP6	0.151	0.324	**1.000**	0.173	0.400	0
	CSP7	0.104	0.225	0.750	0.104	0.250	0
S^4	Rigid	**0.023**	0.680	0.960	0.571	**0.940**	0.260
	Flexible	**0.271**	0.340	**0.812**	0.037	**0.500**	**0.250**
	CSP5	0.011	**0.561**	**0.833**	0.450	0.833	0
	CSP6	0.093	0.400	**1.000**	**0.207**	**0.800**	0
	CSP7	**0.029**	**0.287**	**0.875**	**0.125**	0.375	0
S^4/kNN (50/50)	Rigid	0.026	**0.688**	0.940	**0.629**	**0.940**	0.280
	Flexible	0.302	**0.346**	0.750	**0.050**	0.438	0.188
	CSP5	**0.000**	0.500	**0.833**	**0.478**	**1.000**	0
	CSP6	**0.047**	**0.440**	**1.000**	0.107	**0.800**	**0.200**
	CSP7	0.033	0.221	0.625	0.104	**0.500**	0

datasets. These results support the benefits of using S^4, since it is not subject to the same pitfalls of kNN and centroid radius cropping as illustrated in Figure 5.5.

Also, note that our implementation of centroid radius cropping is more generous than the standard one used in the literature [11]. Specifically, naive radius cropping selects any atom that lies within a pre-defined spatial radius. However, this often results in partial molecules being cropped, which naturally leads to lower performance. Instead, we consider all atoms in a given molecule if its *centroid* lies within 15 Å of a randomly sampled central molecule, and only consider complete molecules up to our token budget, which mitigates the issue of fragmented molecules during training. kNN is also implemented such that it only considers complete molecules up to the token budget, ordered by neighbor distance from a randomly sampled central molecule.

S^4 Radius Size Ablation. Next, we investigate the sensitivity of OXTAL to the choice of S^4 shell radius size in Table 5.5. We see that there are generally

Table 5.5: OXTAL ablation results highlighting the effect of S^4 shell radius size. The best results are **bolded** and second best are underlined.

Radius Size	Dataset	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
3.5 Å	Rigid	**0.015**	**0.694**	**0.980**	0.596	0.920	**0.320**
	Flexible	**0.275**	**0.394**	0.688	0.041	0.375	**0.250**
	CSP5	0.006	0.439	0.833	0.433	0.833	0
	CSP6	0.068	0.409	**1.000**	**0.174**	0.600	**0.400**
	CSP7	0.095	**0.238**	**0.750**	0.110	**0.500**	0
4.5 Å	Rigid	0.026	0.688	0.940	**0.629**	0.940	0.280
	Flexible	0.302	0.346	**0.750**	**0.050**	**0.438**	0.188
	CSP5	**0.000**	**0.500**	0.833	**0.478**	**1.000**	0
	CSP6	**0.047**	**0.440**	**1.000**	0.107	**0.800**	0.200
	CSP7	**0.033**	0.221	0.625	0.104	**0.500**	0
5.5 Å	Rigid	0.047	0.690	0.940	0.625	**0.960**	0.220
	Flexible	0.414	0.346	**0.750**	0.028	0.312	0.188
	CSP5	0.012	0.470	0.667	0.439	0.667	0
	CSP6	0.147	0.375	**1.000**	0.147	0.600	0.200
	CSP7	0.101	0.234	**0.750**	**0.142**	0.375	0

minimal changes in performance across three different choices of radius size. Recall that our main OXTAL model uses a shell radius size of 4.5, which was selected from analyzing the distribution of intermolecular distances presented in Figure 5.6. From this figure, both 4.5 and 5 Å are natural choices for the first shell, however due to the "layered" nature of crystallization, we consider a second shell at 9 rather than 10 Å to be more fitting.

Overall, our current implementation of S^4 may slightly favor smaller shell radius sizes due to the overall token budget constraint, which limits the number of full shells we are able to accommodate. Regardless, we see that all three radius settings for S^4 outperform existing cropping methods from Table 5.4, suggesting that OXTAL is somewhat robust to reasonable choices in S^4 radius size.

Model Size Ablation. To further investigate the effect of model size, we train a smaller version of OXTAL that contains roughly 50M total parameters. Recall that the full OXTAL model contains approximately 100M parameters, with the 70M parameter diffusion transformer accounting for the bulk of the size. The smaller model uses the same architecture as the full model, but with a reduced diffusion transformer that contains only 4 blocks instead of 12. All other model components and training hyperparametetrs remain unchanged.

Table 5.6: OXTAL ablation results highlighting the effect of OXTAL model size. The best results are in **bold** and second best are underlined.

Model	Dataset	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
A-Transformer	Rigid	0.731	0.015	0.060	0.033	0.120	0.060
	Flexible	0.874	0.002	0.063	0	0	0
	CSP5	0.833	0	0	0	0	0
	CSP6	0.967	0	0	0	0	0
	CSP7	0.950	0	0	0	0	0
AssembleFlow	Rigid	0.524	0.001	0.040	0.211	0.760	0
	Flexible	0.850	0	0	0	0	0
	CSP5	0.717	0	0	0.150	0.500	0
	CSP6	0.800	0	0	0.073	0.200	0
	CSP7	0.808	0	0	0.063	0.250	0
OXTAL (50M)	Rigid	0.066	**0.712**	**0.980**	0.621	**0.940**	**0.320**
	Flexible	0.423	0.253	0.562	0.018	0.188	0.062
	CSP5	0.170	0.491	**1.000**	0.400	0.833	0
	CSP6	0.261	0.370	**1.000**	0.094	0.400	**0.200**
	CSP7	0.159	0.173	**0.750**	**0.105**	0.250	**0.125**
OXTAL (100M)	Rigid	**0.026**	0.688	0.940	**0.629**	**0.940**	0.280
	Flexible	**0.302**	**0.346**	**0.750**	**0.050**	**0.438**	**0.188**
	CSP5	**0.000**	**0.500**	0.833	**0.478**	**1.000**	0
	CSP6	**0.047**	**0.440**	**1.000**	**0.107**	**0.800**	**0.200**
	CSP7	**0.033**	**0.221**	0.625	0.104	**0.500**	0

As shown in Table 5.6, halving the parameter budget of OXTAL reduces performance for several metrics. This is unsurprising, as larger models generally have greater capacity to learn the complex distribution of thermodynamic and kinetic factors that govern crystal formation. That said, the OXTAL (50M) model does outperform OXTAL (100M) on a few metrics since these results are reported at 50k training steps, and the smaller model may have converged faster at this point in training. We also see that the smaller 50M parameter OXTAL model significantly outperforms the other *ab-initio* ML methods, which we have provided again here for reference. A-Transformer contains roughly 42M parameters, which provides a fairly direct comparison to the 50M OXTAL model. On the other hand, Guo, Bengio and Liu [187] do not report exact parameter budgets for AssembleFlow, but we estimate that it contains roughly 8M parameters from the publicly released model checkpoints and codebae. The drastically smaller parameter budget is likely due to the fact that AssembleFlow uses an SE(3)-equivariant architecture, which makes it difficult to scale. This result reinforces the importance of scalability while also highlighting the benefits of OXTAL's lattice-free and non-equivariant architecture at a smaller scale.

Table 5.7: Performance of the explicitly equivariant EquiformerV2 model compared to OXtal, which does not strictly enforce symmetries.

Model	Dataset	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
EquiformerV2	Rigid	0.044	0.009	0.272	0.009	0.060	0
	Flexible	**0.084**	0	0	0	0	0
	CSP5	0.006	0	0	0	0	0
	CSP6	0.181	0	0	0	0	0
	CSP7	0.267	0	0	0	0	0
OXtal	Rigid	**0.026**	**0.688**	**0.940**	**0.629**	**0.940**	**0.280**
	Flexible	0.302	**0.346**	**0.750**	**0.050**	**0.438**	**0.188**
	CSP5	**0.000**	**0.500**	**0.833**	**0.478**	**1.000**	0
	CSP6	**0.047**	**0.440**	**1.000**	**0.107**	**0.800**	**0.200**
	CSP7	**0.033**	**0.221**	**0.625**	**0.104**	**0.500**	0

Note that although we did not perform a dedicated token budget ablation for OXtal, preliminary experiments with lower token sizes for the diffusion transformer yielded much higher training losses. Specifically, the average loss for OXtal trained with a maximum token size of 448, 512, and 640 (final) were around 4.22, 3.81, and 2.32, respectively. This suggests that OXtal benefits from a larger token budget, which allows it to capture more of the complex interactions in crystal packing.

Equivariance Ablation. To further highlight the importance of model scaling that a non-equivariant architecture provides, we perform an ablation study with EquiformerV2 [34]. In order to accommodate the larger molecular crystals in our training dataset, we are forced to significantly reduce the parameter budget for EquiformerV2 to fit it into memory. Specifically, we reduce the batch size to 4, and use a hidden dimension size of 32, 2 layers, and 4 attention heads. The rest of the training pipeline, including the use of S^4 cropping, remain the same as for OXtal. Overall, the EquiformerV2 version of the model has a total parameter size of 15M, as opposed to the 100M total parameter size of OXtal.

From Table 5.7, we see that EquiformerV2 performs quite well on the collision metric, indicating that the model is generally able to produce physically feasible structures. Furthermore, it is also able to recover a few crystal packings and molecular conformers from the Rigid molecule dataset. However, EquiformerV2 completely fails to recover any packings or conformers from the more challenging Flexible and CSP blind test datasets, which is unsurprising due to its drastically smaller parameter budget. Although these results are clearly influenced by the memory-constrained parameter budget, we were unable to evaluate a hybrid approach incorporating

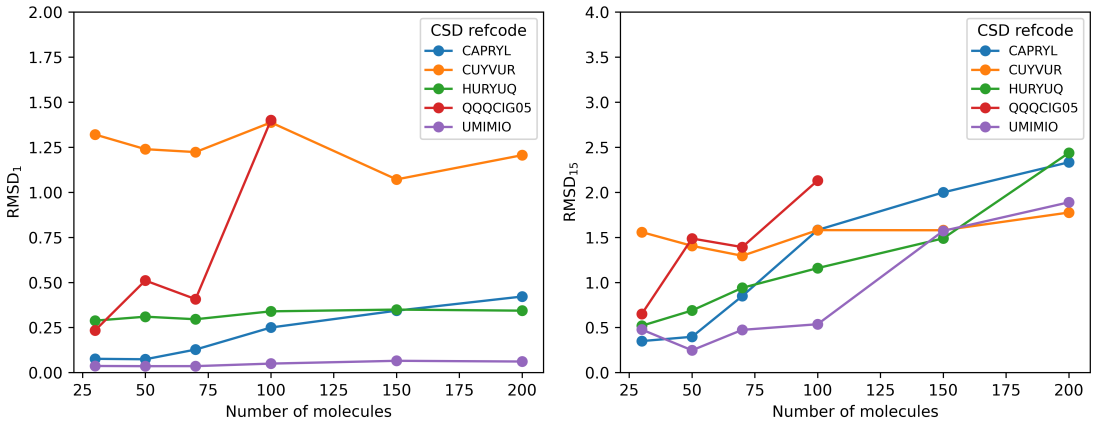


Figure 5.14: Performance of OXTAL on increasingly larger inference crystal blocks (denoted by the number of molecule copies generated). $RMSD_1$ (left) remains roughly constant, whereas $RMSD_{15}$ increases slightly for larger predicted packings. Note that QQQCIG05, which contains 42 heavy atoms (i.e. tokens) per molecules, goes out of memory past 100 molecules.

the equivariant components of AlphaFold2 into the AlphaFold3 framework, as no suitable implementation of the AlphaFold2 codebase was publicly available.

These results show the necessity of using a scalable, non-equivariant model: enforcing equivariance imposes a critical bottleneck on model size, which renders it insufficient for capturing larger molecular crystals in the complex crystallization landscape. This validates our design choice of scaling via a non-equivariant model and breaking free from lattice constraints while softly enforcing symmetry constraints.

5.5.5 Inference Analysis

In this section, we provide a deeper analysis into the inference and generation capabilities of OXTAL. Specifically, we investigate the performance of OXTAL on generating significantly larger crystal blocks, evaluate how the model performs when allowing for additional sampling, and provide a small qualitative analysis on the diversity of generated samples.

Inference on Larger Crystal Blocks. To evaluate the long-range periodicity of OXTAL, we perform inference on increasingly larger crystal blocks for a handful of different crystals (Figure 5.14). The respective number of atoms, and therefore tokens, for each molecule are provided in parentheses as follows: CAPRYL (10), CUYVUR (18), HURYUQ (14), QQQCIG05 (42), UMIMIO (12). Recall that OXTAL is trained with a maximum token budget of 640 (Appendix C.2.2), hence generating

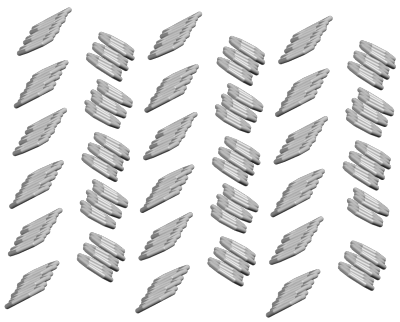


Figure 5.15: Example of ANTCEN with over 2,400 tokens with a herringbone backing structure. OXTAL learns small local neighborhoods from S^4 crops and generalizes to infer large and periodic structures.

200 copies of CAPRYL requires 2,000 tokens and 100 copies of QQQCIG05 requires 4,200 tokens, which are far beyond anything the training dataset would have seen.

In general, conformer recovery (denoted by RMSD_1) remains somewhat constant as more molecule copies are generated. This makes sense because if the model is already able to capture the correct crystal conformer, generating more copies of that should not change the conformer itself significantly. In terms of overall prediction accuracy (denoted by RMSD_{15}), OXTAL does struggle slightly with generating more molecules, since the packing arrangements cover distances that are further away. However, we note that although the RMSD_{15} does increase for these larger blocks, they still mostly remain under the 2.0 Å threshold to be considered "approximately solved."

This analysis supports the long-range generalizability of OXTAL to generate larger periodic packings, and we provide a visualization of the approximately solved larger packing for ANTCEN in Figure 5.15, which contains 2,400 tokens. Recall that during training, OXTAL only ever sees small S^4 crops containing a maximum of 640 atom tokens. This highlights the model's ability to learn local neighborhoods and generalize them to much larger periodic structures during inference.

Additional Sample Evaluation for CSP Blind Test 5. For all previously reported results, we evaluate OXTAL using 30 generated samples per crystal target. This is done in order to highlight the sample efficiency of OXTAL (cf. Figures 5.9 and 5.10), since few-sample success is critical for downstream screening and design. However, we note that this puts OXTAL at somewhat of a disadvantage when comparing against traditional DFT methods, which often generate hundreds or thousands of candidate packings. Here, we provide a more direct comparison of

Table 5.8: Evaluation of OXTAL on CSP blind test 5 with roughly the same number of generated samples per crystal target (n_S) as DFT_{avg}.

Model	n_S	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	$\text{Pac}_C \uparrow$	$\text{Rec}_S \uparrow$	$\text{Rec}_C \uparrow$	$\widetilde{\text{Sol}}_C \uparrow$
DFT_{avg}	464	0.039	0.323	0.611	**0.784**	0.733	**0.544**
OXTAL	30	**0.006**	**0.667**	<u>0.833</u>	<u>0.572</u>	<u>0.833</u>	0.167
OXTAL	500	<u>0.009</u>	<u>0.536</u>	**1.000**	0.548	**1.000**	<u>0.500</u>

OXTAL to DFT methods by evaluating OXTAL using the same number of generated samples on the CSP blind test 5 dataset in Table 5.8.

As expected, allowing OXTAL to generate more candidate packings increases its performance on our per-crystal evaluation metrics, while remaining roughly the same on per-sample metrics. Notably, we also see that increasing the number of generated samples bring the approximate solve rate of OXTAL equal to that of DFT_{avg}. This suggests that the model sampler is indeed able to generate accurate crystal packings and compete with traditional DFT methods.

5.5.6 Survey of Chemical Interpretability

Beyond benchmark metrics, we examine OXTAL's ability to reproduce chemically meaningful intramolecular and intermolecular features in practically relevant crystals. Figures 5.1, 5.16 and 5.17 highlight accurate packings ($\text{RMSD}_{15} < 1.5$Å) across diverse rigid and flexible chemotypes: drug-like molecules (HURYUQ), polymer precursors (CAPRYL), organometallics (ACACPD), π-conjugated materials (QQQCIG), and ROY (QAXMEH), the molecule which has the most known polymorphs.

Molecular interactions. For intramolecular interactions, OXTAL recovers *solid-state* geometries for highly flexible molecules (which are highly influenced by packing), including small-molecule drugs as well as biomolecular fragments in the Protein DataBank (e.g. $\text{RMSD}_1 = 1.3$ Å for a 6-mer peptide with 17 rotatable bonds (2OKZ) in Figure 5.16(a)). For intermolecular interactions, OXTAL accurately captures both strong and weak interactions, both in registry and lengths. Examples in Figure 5.16(b) include the complementary hydrogen bonds in a semiconducting crystal (XIJJOT), the weak Cl-H halogen bonds in an organometallic catalyst (OJIGOG), π-π stacking in π-functional molecule (TEPNIT), and multiple weak contacts in a cluster of the flexible drug aripiprazole (MELFIT) in Figure 5.17(b). Zooming out, OXTAL reproduces diverse packing motifs, including 1D columnar structures (e.g. BALNAD in Figure 5.16(d)), quasi-2D herringbone structures (e.g. ANTCEN in Figure 5.15), and 2D brickwork structures (e.g. UMIMIO in Figure 5.1).

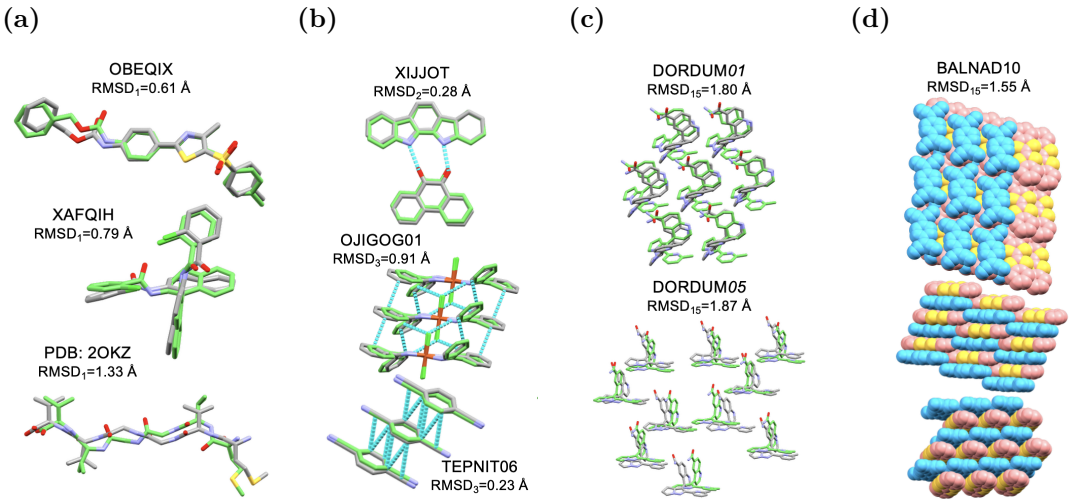


Figure 5.16: OXTAL (green) captures experimental (grey) (a) intramolecular and (b) intermolecular interactions in drug-like molecules, peptides, semiconductors, and catalysts. OXTAL can further infer (c) distinct experimental polymorphs as well as (d) co-crystals.

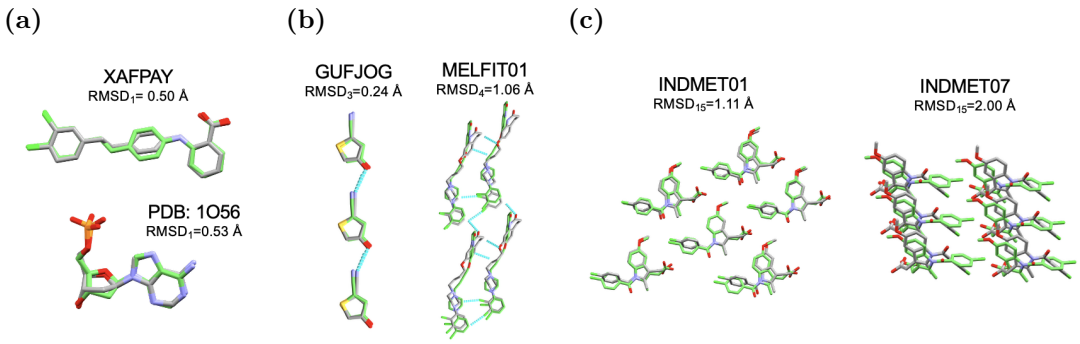


Figure 5.17: Examples of OXTAL generated structures (color) compared against experimental structures (gray) for (a) flexible conformers, (b) intermolecular interactions, and (c) polymorphs.

Polymorphs. For molecules with many known polymorphs, including highly rotatable scaffolds, independent OXTAL samples predict distinct experimental polymorphs (e.g., the drugs galunisertib DORDUM in Figure 5.16(c) and indomethacin INDMET in Figure 5.17(c)). This suggests the sampler can *explore multiple kinetic and thermodynamic basins* rather than collapsing to a single motif.

Multi-component Systems. Lastly, OXTAL is not restricted to single-component systems. OXTAL can correctly predict the interactions between electron donor and acceptors (Figures 5.16 and 5.18), which dictate the crystals' electronic properties. For the charge-transfer semiconducting co-crystals BALNAD and PERTCQ, OXTAL correctly reproduces the 1D π-stacked columns with alternating donors and acceptors, the precise inter-columnar brick-wall registry, as well as the π-π stacking distances.

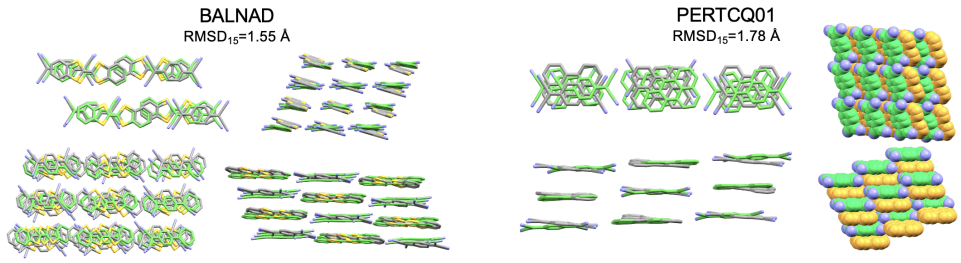


Figure 5.18: Examples of OXTAL generated co-crystal structures (color) compared against experimental structures (gray).

Diversity of Generated Samples. Finally, in Figure 5.19, we provide a preliminary qualitative visualization of some example packings generated by OXTAL. We see that although several samples lie in the same orientation, some generated samples also present a more complex herringbone packing. This is evident in the XATJOT co-crystal, which does not collapse the two different molecular components into the same orientation. These results suggest that OXTAL may prefer planar packings, which are more prevalent in the training dataset. However, this may be mitigated with additional tuning on noise scheduler parameters at inference time.

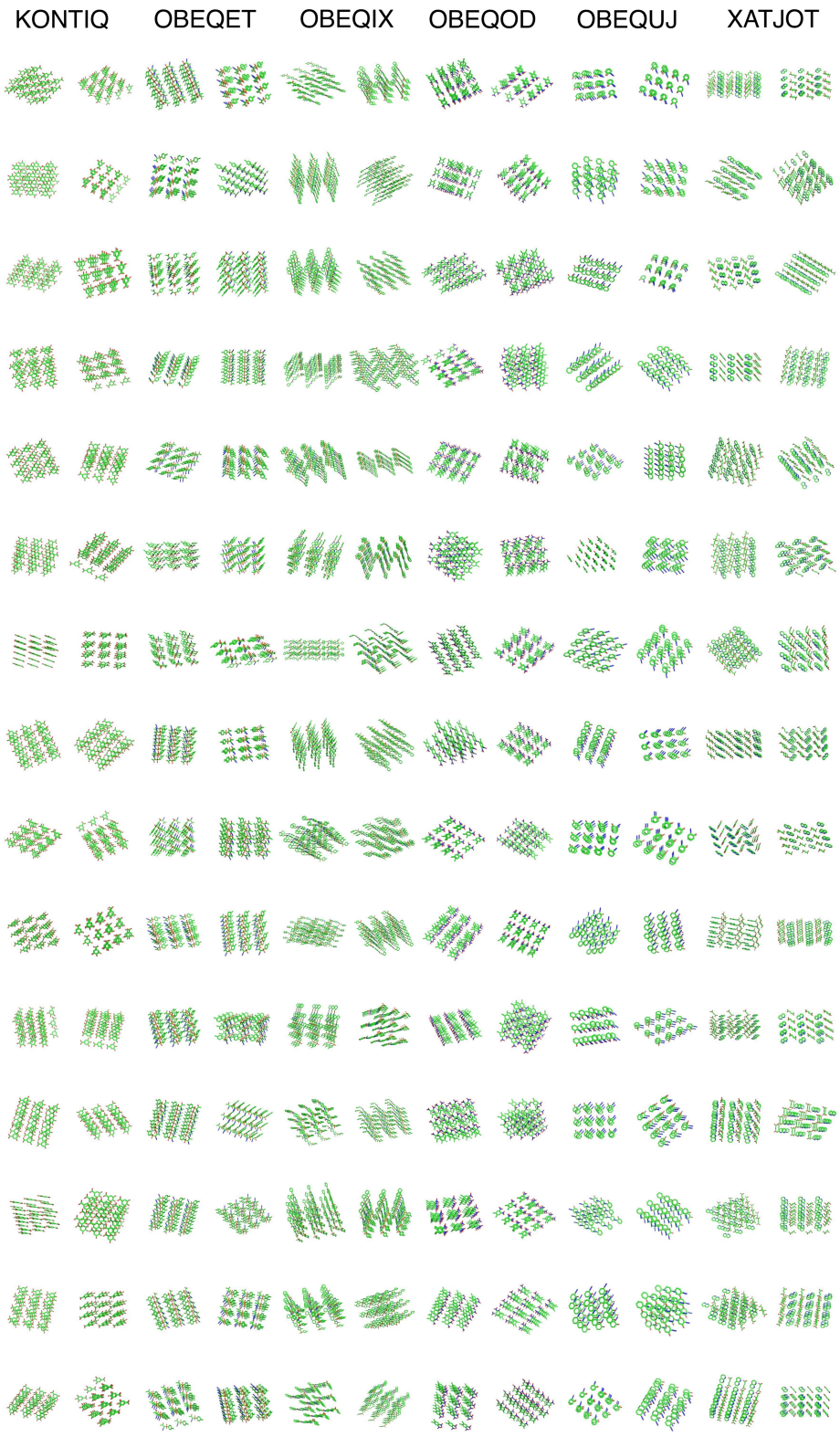


Figure 5.19: Example set of structures generated by OXtal for various crystals (labeled by CSD ID).

5.6 Summary and Outlook

In this chapter, we introduced OXTAL, a large-scale all-atom diffusion model for 3D molecular crystal structure prediction that learns the joint distribution of molecular conformations and periodic packing conditioned on 2D graphs (Section 5.4). Discarding explicit equivariance and unit-cell parametrization for symmetry-aware augmentation and S^4 cropping, OXTAL learns periodic motifs from locally consistent neighborhoods at scale, enabling efficient sampling at all-atom resolution (Section 5.4.1). Empirically, OXTAL achieves state-of-the-art results among *ab initio* ML methods, and attains competitive lattice recovery at orders-of-magnitude lower inference cost compared to DFT-based methods (Section 5.5).

Conceptually, our S^4 approach reframes the task of crystal structure prediction from "generate a valid global unit cell" to "generate a locally consistent, periodic atomic environment." Leveraging these powerful intermolecular local representations allows us to scale model size and accomodate larger molecular crystals. Furthermore, we also show that the error due to S^4 cropping decreases as the number of tokens increases (Proposition 5.4.1).

In our experimental results, we defined a broad range of different metrics (Section 5.5.1), and showed that OXTAL provides a step-change in progress over existing ML methods for CSP on both rigid and flexible molecules (Section 5.5.2). When evaluating OXTAL on the community CSP blind test datasets, we saw that OXTAL is able to keep up with the more expensive DFT-based methods, and is much more sample efficient (Section 5.5.3). An energy analysis showed that OXTAL does not generate energetically unfavorable structures, suggesting that it is sampling the correct energy wells. Importantly, OXTAL's ability to perform zero-shot structure prediction (unlike DFT which requires recomputing energies for each new molecule) translates into direct monetary impact at inference item and opens the door to high-throughput virtual screening of molecular crystals.

Beyond standard benchmark datasets, we also provided an in-depth ablation study to investigate the role of different OXTAL components (Section 5.5.4). These included analyses on the role of symmetry augmentation, the S^4 representation, and model size, which are all critical to achieving strong performance. This was complemented with an inference analysis (Section 5.5.5) that showed how OXTAL is able to generalize to much larger crystal blocks than seen during training while preserving the learned local consistency to recover global periodicity.

Finally, to emphasize the practical relevance of OXTAL, we provided a detailed chemical survey (Section 5.5.6) that highlighted OXTAL's ability to recover meaningful intramolecular and intermolecular interactions, sample distinct experimental

polymorphs, and generate multi-component co-crystal. This indicates that OXTAL is not merely memorizing training data, but is learning physically relevant motifs that govern molecular crystal packing and capturing the non-trivial kinetic and thermodynamicly governed crystallization landscape.

In terms of future directions for OXTAL, several opportunities arise. First, an integration of reliable ranking and local relaxation can lead to more consistent solves. Second, conditioning on crystallization context (solvent, co-formers, temperature, supersaturation, etc.) could refine the posterior toward experimentally observed forms. Third, improved sample efficiency would further reduce screening cost. On the inference side, many avenues remain to improve sample diversity and quality, including better noise scheduling, classifier guidance, and enhanced sampling techniques. On the training side, it would be interesting to explore richer molecular representations, such as those provided by MoSE, which may further improve the model's ability to capture complex chemical motifs.

While OXTAL substantially reduces the computational cost of crystal structure prediction, translating these predictions into experimentally synthesizable materials remains an important downstream challenge as well. In practice, OXTAL could form part of a broader materials discovery pipeline, where large numbers of candidate structures are first generated and filtered according to predicted stability, synthesizability, and target material properties, before undergoing higher-fidelity quantum chemical simulation and eventual experimental synthesis and characterization. In this setting, the efficiency of OXTAL is particularly valuable, as it enables high-throughput exploration of crystal landscapes that would be prohibitively expensive with traditional DFT-based workflows.

Overall, this chapter has highlighted the important interplay between representation, efficiency, and adaptability in the context of molecular crystal structure prediction. Although it seems like a natural approach to enforce equivariance and use a unit cell parametrization for periodic structures, we have shown that these choices can limit scalability and generalizability. Instead, by reworking this traditionally global problem into a local one (with meaningful local representations), and sacrificing hard-coded symmetry for scale, we achieve notable progress towards solving this long-standing challenge in computational chemistry.

6
Conclusion

The prevalence of graph-structured data has motivated the design of new machine learning methods that can capture increasingly complex structural relationships. GNNs and graph transformers have emerged as the dominant paradigm for learning on such data. However, their performance and interpretability are often dictated by how they represent and propagate local structural information, which remains an open question. Different applications, such as molecular and crystalline modelling, also require different representations and inductive biases. Consequently, this thesis has proposed new principles and methods for rethinking local structural awareness in graph representation learning, with a particular focus on molecular systems. We have examined two different formulations of local structure, one for capturing local structural motifs at the intramolecular scale, and another for capturing local interactions at the intermolecular scale. A summary of our contributions in this respect is provided below:

1. In Chapter 3, we investigated the theoretical expressivity of GNNs through the lens of graph motif parameters. We showed that this provides a unifying framework for previous works that have enriched GNN expressivity through subgraph isomorphism and adhoc homomorphism counts. In particular, we proved theoretical results demonstrating that using the homomorphism basis of a graph motif parameter offers a more principled and expressive way for incorporating local structural motifs into GNNs. From a practical perspective, we demonstrated how to most effectively incorporate these homomorphism basis features, which led to strong performance gains across a range of benchmarks—including molecular property prediction on ZINC and QM9, link prediction on COLLAB, and expressivity tests on the BREC dataset.

2. In Chapter 4, we introduced Motif Structural Encoding (MoSE) as a novel, flexible, and expressive structural encoding for graph transformers that leverages graph homomorphism counts. Theoretically, we proved that MoSE generalizes the popular random walk structural encodings (RWSE) and relative random walk probabilities (RRWP). We also established formal results relating MoSE and RWSE to the Weisfeiler-Lehman hierarchy. Empirically, we showed how MoSE consistently improved performance across molecular datasets such as ZINC, PCQM4Mv2, and Peptides, as well as non-molecular benchmarks like CIFAR10, MNIST, and a newly developed synthetic dataset for predicting fractional domination numbers, showing that MoSE generalizes beyond the chemical domain.

3. In Chapter 5, we proposed OXtal, the first large-scale all-atom diffusion model for predicting molecular crystal packings that leverages local representations to capture global periodic symmetries. Specifically, OXtal discards explicit unit-cell parameterization and rigid equivariant constraints, instead learning periodic motifs through a novel Stoichiometric Stochastic Shell Sampling (S^4) method. This design allows OXtal to learn periodic symmetries directly from atomic neighborhoods while remaining scalable to large systems. Empirically, OXtal achieves state-of-the-art performance among *ab initio* CSP approaches and remains competitive with traditional DFT-based methods, all while operating at orders-of-magnitude lower inference cost. Beyond quantitative benchmarks, we showed that the model demonstrates strong chemical interpretability and is able to accurately capture diverse intramolecular and intermolecular interactions, polymorphism, and packing motifs in both rigid and flexible molecules.

More broadly, this thesis contributes to the larger body of work that explores the "expressivity" of various graph learning models. Although we mainly focus on the definition of expressivity that pertains to distinguishing power, other notions of expressivity exist, which instead focus on universal approximation [188, 189], spectral representation capacity [125], logical definability [190], or the ability to efficiently encode domain-specific invariances and symmetries [14].

Our results suggest that expressivity should not be viewed purely as a theoretical property, but rather as a mechanism for injecting useful inductive biases that align model representations with the underlying scientific domain. This distinction is particularly important because recent work has shown that increased expressivity does not automatically imply improved downstream performance or generalization

[191]. Instead, margin-based analyses of GNNs suggest that richer structural representations that enable better *separation* of meaningful graph classes is the key to improved generalization. In particular, Franks et al. [191] provide evidence that subgraph-aware GNNs can improve classifier margins and yield stronger generalization guarantees. This further supports the motivation underlying the approaches proposed in Chapter 3 and Chapter 4, since subgraph count-based representations are contained within the class of graph motif parameters.

In general, incorporating domain knowledge into model design can be achieved in several ways, including through the construction of custom inductive biases, the design of architectural components, and the incorporation of domain-specific features, constraints, or symmetries. Understanding the underlying scientific domain and data distribution is particularly important in graph representation learning, where structural assumptions can strongly influence model performance. In molecular and crystalline systems, examples include local structural motifs, periodic symmetries, and chemically meaningful neighborhood representations. Incorporating such prior knowledge can improve not only predictive power, but also explainability, sample efficiency, and scientific reliability. In the broader context of biochemical research, this can lead to a variety of benefits, including improved drug candidates, better understanding of biological processes, faster screening, and more efficient exploration of chemical and materials design spaces.

Looking forward, several promising directions arise from this work. First, an integration of stronger structural encodings as molecular features could boost performance for organic crystal structure prediction models. There is also scope to explore the extension of these encodings for heterogeneous relational graphs, as well as further integration with databases and tabular reasoning models, allowing for more practically efficient computation. On the theoretical side, many open questions remain concerning the precise relationship between performance and generalization in different encoding methods and graph learning systems. Additional work could also provide a more rigorous mathematical framework for understanding the relationship of local to global symmetries in periodic systems.

Overall, this thesis has investigated how local structural information can be effectively and efficiently represented within graph-based deep learning frameworks, with a particular focus on molecular and crystalline systems. By exploring the interplay between expressivity, inductive bias, and scalability, this work advances both the theoretical understanding and practical design of graph representation learning models. Hopefully, this can help promote additional interdisciplinary research, allowing for the development of more powerful, scalable, and interpretable models at the intersection of machine learning and scientific discovery.

References

[1] Emily Jin et al. Homomorphism Counts for Graph Neural Networks: All About That Basis. In *ICML*. 2024.

[2] Linus Bao et al. Homomorphism Counts as Structural Encodings for Graph Learning. In *ICLR*. 2025.

[3] Emily Jin et al. OXtal: An All-Atom Diffusion Model for Organic Crystal Structure Prediction. In *ICLR*. 2026.

[4] Emily Jin, J Alison Noble and Mireille Gomes. A Review of Computer-Aided Diagnostic Algorithms for Cervical Neoplasia and an Assessment of Their Applicability to Female Genital Schistosomiasis. In *Mayo Clinic Proceedings: Digital Health* 1.3 (2023).

[5] Ilya Sutskever, Oriol Vinyals and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*. 2014.

[6] Ashish Vaswani et al. Attention Is All You Need. In *NeurIPS*. 2017.

[7] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*. 2012.

[8] Kaiming He et al. Deep Residual Learning for Image Recognition. In *CVPR*. 2016.

[9] Alexey Dosovitskiy et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. In *ICLR*. 2021.

[10] John Jumper et al. Highly Accurate Protein Structure Prediction with AlphaFold. In *Nature* 596.7873 (2021).

[11] Josh Abramson et al. Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3. In *Nature* 630.8016 (2024).

[12] Avishek Joey Bose et al. SE(3)-Stochastic Flow Matching for Protein Backbone Generation. In *ICLR*. 2024.

[13] William L Hamilton. *Graph Representation Learning*. Springer, 2020.

[14] Michael M Bronstein et al. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. In *arXiv* (2021).

[15] Shiwen Wu et al. Graph Neural Networks in Recommender Systems: A Survey. In *ACM Computing Surveys* 55.5 (2022).

[16] Michelle M Li, Kexin Huang and Marinka Zitnik. Graph Representation Learning in Biomedicine and Healthcare. In *Nature Biomedical Engineering* (2022).

[17] Jessica Vamathevan et al. Applications of Machine Learning in Drug Discovery and Development. In *Nature Reviews Drug Discovery* 18.6 (2019).

[18] Claudio Zeni et al. A Generative Model for Inorganic Materials Design. In *Nature* 639.8055 (2025).

[19] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. 2017.

[20] Will Hamilton, Zhitao Ying and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 2017.

[21] Petar Velickovic et al. Graph Attention Networks. In *ICLR*. 2018.

[22] Justin Gilmer et al. Neural Message Passing for Quantum Chemistry. In *ICML*. 2017.

[23] Keyulu Xu et al. How Powerful are Graph Neural Networks? In *ICLR*. 2019.

[24] Vıctor Garcia Satorras, Emiel Hoogeboom and Max Welling. E(n) Equivariant Graph Neural Networks. In *ICML*. PMLR. 2021.

[25] Christopher Morris et al. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI*. 2019.

[26] Giorgos Bouritsas et al. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2022).

[27] Pablo Barceló et al. Graph Neural Networks with Local Graph Parameters. In *NeurIPS*. 2021.

[28] Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs. In *arXiv* (2020).

[29] Ladislav Rampásek et al. Recipe for a General, Powerful, Scalable Graph Transformer. In *NeurIPS*. 2022.

[30] Liheng Ma et al. Graph Inductive Biases in Transformers without Message Passing. In *ICML*. 2023.

[31] Chaitanya K Joshi. Transformers are Graph Neural Networks. In *arXiv* (2025).

[32] Vijay Prakash Dwivedi et al. Graph Neural Networks with Learnable Structural and Positional Representations. In *ICLR*. 2022.

[33] Vijay Prakash Dwivedi et al. Benchmarking Graph Neural Networks. In *Journal of Machine Learning Research* 24.43 (2023).

[34] Yi-Lun Liao et al. Equiformerv2: Improved Equivariant Transformer for Scaling to Higher-Degree Representations. In *ICLR*. 2024.

[35] Frank Jensen. *Introduction to Computational Chemistry*. Wiley, 2017.

[36] Colin R Groom et al. The Cambridge Structural Database. In *Structural Science* 72.2 (2016).

[37] David A Bardwell et al. Towards Crystal Structure Prediction of Complex Organic Compounds–A Report on the Fifth Blind Test. In *Structural Science* 67.6 (2011).

[38] Anthony M Reilly et al. Report on the Sixth Blind Test of Organic Crystal Structure Prediction Methods. In *Structural Science* 72.4 (2016).

[39] Lily M Hunnisett et al. The Seventh Blind Test of Crystal Structure Prediction: Structure Generation Methods. In *Structural Science* 80.6 (2024).

[40] Zhenqin Wu et al. MoleculeNet: A Benchmark for Molecular Machine Learning. In *Chemical Science* 9.2 (2018).

[41] John J Irwin et al. ZINC: A Free Tool to Discover Chemistry for Biology. In *Journal of Chemical Information and Modeling* 52.7 (2012).

[42] Thomas Gaudelet et al. Utilizing Graph Machine Learning within Drug Discovery and Development. In *Briefings in Bioinformatics* 22.6 (2021).

[43] John Maddox. Crystals from First Principles. In *Nature* 335.6187 (1988).

[44] Thomas N Kipf and Max Welling. Variational Graph Auto-Encoders. In *arXiv* (2016).

[45] Tian Xie et al. Crystal Diffusion Variational Autoencoder for Periodic Material Generation. In *ICLR*. 2022.

[46] Yang Song et al. Score-based generative modeling through stochastic differential equations. In *ICLR*. 2021.

[47] Rui Jiao et al. Crystal Structure Prediction by Joint Equivariant Diffusion. In *NeurIPS*. 2023.

[48] Yaron Lipman et al. Flow Matching for Generative Modeling. In *ICLR*. 2023.

[49] Benjamin Kurt Miller et al. FlowMM: Generating Materials with Riemannian Flow Matching. In *ICML*. 2024.

[50] Fabian Fuchs et al. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *NeurIPS*. 2020.

[51] Theodore L Brown et al. *Chemistry: The Central Science*. Vol. 14. Pearson, 2017.

[52] Aleksandr Aleksandrovich Chernov. *Modern Crystallography III: Crystal Growth*. Vol. 36. Springer, 2012.

[53] Radu Curticapean, Holger Dell and Dániel Marx. Homomorphisms Are a Good Basis for Counting Small Subgraphs. In *STOC*. 2017.

[54] László Lovász. Operations with Structures. In *Acta Mathematica Hungarica* 18.3-4 (1967).

[55] László Lovász. *Large Networks and Graph Limits*. American Mathematical Society, 2012.

[56] Hoang Nguyen and Takanori Maehara. Graph Homomorphism Convolution. In *ICML*. 2020.

[57] Michael Freedman, László Lovász and Alexander Schrijver. Reflection Positivity, Rank Connectivity, and Homomorphism of Graphs. In *Journal of the American Mathematical Society* 20.1 (2007).

[58] Jin-Yi Cai and Artem Govorov. On a Theorem of Lovász that $\hom(\cdot, H)$ Determines the Isomorphism Type of H*. In *ACM Transactions on Computation Theory (TOCT)* 13.2 (2021).

[59] Bingxu Zhang et al. The Expressive Power of Graph Neural Networks: A Survey. In *IEEE Transactions on Knowledge and Data Engineering* (2025).

[60] Ningyuan Teresa Huang and Soledad Villar. A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants. In *ICASSP*. 2021.

[61] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. In *Nauchno-Technicheskaya Informatsia* 2.9 (1968).

[62] Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

[63] Martin Grohe and Martin Otto. Pebble Games and Linear Equations. In *The Journal of Symbolic Logic* 80.3 (2015).

[64] Jin-Yi Cai, Martin Fürer and Neil Immerman. An Optimal Lower Bound on the Number of Variables for Graph Identification. In *FOCS*. 1989.

[65] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. In *Neural Computation* 15.6 (2003).

[66] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. In *ICLR*. 2023.

[67] Xavier Bresson and Thomas Laurent. Residual Gated Graph ConvNets. In *arXiv* (2017).

[68] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 2003.

[69] Brian DO Anderson. Reverse-Time Diffusion Equation Models. In *Stochastic Processes and their Applications* 12.3 (1982).

[70] Peter Holderrieth et al. Generator Matching: Generative Modeling with Arbitrary Markov Processes. In *ICML*. 2025.

[71] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *ICCV*. 2023.

[72] Franco Scarselli et al. The Graph Neural Network Model. In *IEEE Transactions on Neural Networks* 20.1 (2008).

[73] Marco Gori, Gabriele Monfardini and Franco Scarselli. A New Model for Learning in Graph Domains. In *IJCNN*. 2005.

[74] Jonathan Shlomi, Peter Battaglia and Jean-Roch Vlimant. Graph Neural Networks in Particle Physics. In *Machine Learning: Science and Technology* (2020).

[75] David Duvenaud et al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *NeurIPS*. 2015.

[76] Marinka Zitnik, Monica Agrawal and Jure Leskovec. Modeling Polypharmacy Side Effects with Graph Convolutional Networks. In *Bioinformatics* 34.13 (2018).

[77] William L. Hamilton, Zhitao Ying and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 2017.

[78] Haggai Maron et al. Provably Powerful Graph Networks. In *NeurIPS*. 2019.

[79] Haggai Maron et al. On the Universality of Invariant Networks. In *ICML*. 2019.

[80] Nicolas Keriven and Gabriel Peyré. Universal Invariant and Equivariant Graph Neural Networks. In *NeurIPS*. 2019.

[81] Andreas Loukas. What Graph Neural Networks Cannot Learn: Depth vs Width. In *ICLR*. 2020.

[82] Ryoma Sato, Makoto Yamada and Hisashi Kashima. Random Features Strengthen Graph Neural Networks. In *SDM*. 2021.

[83] Ralph Abboud et al. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *IJCAI*. 2021.

[84] Bohang Zhang et al. Rethinking the Expressive Power of GNNs via Graph Biconnectivity. In *ICLR*. 2023.

[85] Radoslav Dimitrov et al. PlanE: Representation Learning over Planar Graphs. In *NeurIPS*. 2023.

[86] Zhengdao Chen et al. Can Graph Neural Networks Count Substructures? In *NeurIPS*. 2020.

[87] Cristian Bodnar et al. Weisfeiler and Lehman Go Cellular: CW Networks. In *NeurIPS*. 2021.

[88] Benjamin Paul Chamberlain et al. Graph neural networks for link prediction with subgraph sketching. In *arXiv preprint arXiv:2209.15486* (2022).

[89] Fabrizio Frasca et al. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *NeurIPS*. 2022.

[90] Beatrice Bevilacqua et al. Equivariant Subgraph Aggregation Networks. In *ICLR*. 2022.

[91] Bohang Zhang et al. A Complete Expressiveness Hierarchy for Subgraph GNNs via Subgraph Weisfeiler-Lehman Tests. In *ICML*. 2023.

[92] Pál András Papp and Roger Wattenhofer. A Theoretical Comparison of Graph Neural Network Extensions. In *ICML*. 2022.

[93] Pascal Welke et al. Expectation-Complete Graph Representations with Homomorphisms. In *ICML*. 2023.

[94] Bohang Zhang et al. Beyond Weisfeiler-Lehman: A Quantitative Framework for GNN Expressiveness. In *ICLR*. 2024.

[95] Hinrikus Wolf et al. Structural Node Embeddings with Homomorphism Counts. In *arXiv* (2023).

[96] Pablo Barceló et al. The Logical Expressiveness of Graph Neural Networks. In *ICLR*. 2020.

[97] Zdenek Dvorák. On Recognizing Graphs by Numbers of Homomorphisms. In *Journal of Graph Theory* 64.4 (2010).

[98] Daniel Neuen. Homomorphism-Distinguishing Closedness for Graphs of Bounded Tree-Width. In *STACS*. 2024.

[99] Pedro Ribeiro et al. A Survey on Subgraph Counting: Concepts, Algorithms, and Applications to Network Motifs and Graphlets. In *ACM Computing Surveys* 54.2 (2022).

[100] Josep Díaz, Maria Serna and Dimitrios M Thilikos. Counting H-colorings of partial k-trees. In *Theoretical Computer Science* 281.1-2 (2002).

[101] Dániel Marx. Can You Beat Treewidth? In *Theory of Computing* 6.1 (2010).

[102] Marco Bressan, Matthias Lanzinger and Marc Roth. The Complexity of Pattern Counting in Directed Graphs, Parameterised by the Outdegree. In *STOC*. 2023.

[103] Hubie Chen and Stefan Mengel. Counting Answers to Existential Positive Queries: A Complexity Classification. In *PODS*. 2016.

[104] Luigi P Cordella et al. A (Sub)graph Isomorphism Algorithm for Matching Large Graphs. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004).

[105] Floris Geerts and Juan L Reutter. Expressiveness and Approximation Properties of Graph Neural Networks. In *ICLR* (2022).

[106] Matthias Lanzinger and Pablo Barceló. On the Power of the Weisfeiler-Leman Test for Graph Motif Parameters. In *ICLR*. 2024.

[107] Marc Roth and Johannes Schmitt. Counting Induced Subgraphs: A Topological Approach to #W[1]-Hardness. In *Algorithmica* 82.8 (2020).

[108] Holger Dell, Marc Roth and Philip Wellnitz. Counting Answers to Existential Questions. In *ICALP*. 2019.

[109] Weihua Hu et al. Strategies for Pre-Training Graph Neural Networks. In *ICLR*. 2020.

[110] Gabriele Corso et al. Principal Neighbourhood Aggregation for Graph Nets. In *NeurIPS*. 2020.

[111] Michael Schlichtkrull et al. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 2018.

[112] Marc Brockschmidt. GNN-FiLM: Graph Neural Networks with Feature-wise Linear Modulation. In *ICML*. 2020.

[113] Ralph Abboud, Radoslav Dimitrov and İsmail İlkan Ceylan. Shortest Path Networks for Graph Property Prediction. In *LoG*. 2022.

[114] Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *ICLR*. 2021.

[115] Weihua Hu et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*. 2020.

[116] Yanbo Wang and Muhan Zhang. An Empirical Study of Realized GNN Expressiveness. In *ICML*. 2024.

[117] Matthias Lanzinger, Reinhard Pichler and Alexander Selzer. Avoiding Materialisation for Guarded Aggregate Queries. In *VLDB*. 2025.

[118] Luis Müller et al. Attending to Graph Transformers. In *Transactions on Machine Learning Research* (2024).

[119] Seongjun Yun et al. Graph Transformer Networks. In *NeurIPS*. 2019.

[120] Guy Bar-Shalom, Beatrice Bevilacqua and Haggai Maron. Subgraphormer: Unifying Subgraph GNNs and Graph Transformers via Graph Products. In *NeurIPS*. 2024.

[121] Eran Rosenbluth et al. Distinguished in Uniform: Self-Attention vs. Virtual Nodes. In *ICLR*. 2024.

[122] Hamed Shirzad et al. Exphormer: Sparse transformers for graphs. In *ICML*. 2023.

[123] Ahsan Shehzad et al. Graph Transformers: A Survey. In *IEEE Transactions on Neural Networks and Learning Systems* (2025).

[124] Devin Kreuzer et al. Rethinking Graph Transformers with Spectral Attention. In *NeurIPS*. 2021.

[125] Derek Lim et al. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. In *ICLR*. 2023.

[126] Derek Lim et al. Expressive Sign Equivariant Networks for Spectral Geometric Learning. In *NeurIPS*. 2023.

[127] Chengxuan Ying et al. Do Transformers Really Perform Badly for Graph Representation? In *NeurIPS*. 2021.

[128] Pan Li et al. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In *NeurIPS*. 2020.

[129] Yuankai Luo et al. Enhancing Graph Transformers with Hierarchical Distance Structural Encoding. In *NeurIPS*. 2024.

[130] Zhengdao Chen et al. On the Equivalence Between Graph Isomorphism Testing and Function Approximation with GNNs. In *NeurIPS*. 2019.

[131] Takanori Maehara and Hoang NT. Deep Homomorphism Networks. In *NeurIPS*. 2024.

[132] Moritz Lichter, Ilia Ponomarenko and Pascal Schweitzer. Walk Refinement, Walk Logic, and the Iteration Number of the Weisfeiler-Leman Algorithm. In *LICS*. 2019.

[133] Frank Fuhlbrück et al. The Weisfeiler-Leman Algorithm and Recognition of Graph Properties. In *Theoretical Computer Science* 895 (2021).

[134] Weihua Hu et al. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. In *NeurIPS*. 2021.

[135] Vijay Prakash Dwivedi et al. Long Range Graph Benchmark. In *NeurIPS*. 2022.

[136] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. 2009.

[137] Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research. In *IEEE Signal Processing Magazine* 29.6 (2012).

[138] Joshua Southern et al. Understanding Virtual Nodes: Oversquashing and Node Heterogeneity. In *ICLR*. 2025.

[139] Uma G. et al. Exploring Prism Graphs with Fractional Domination Parameters. In *Mathematics and Statistics* 12 (2024).

[140] Edward R Scheinerman and Daniel H Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley, 2013.

[141] Nate Schultheiss and Ann Newman. Pharmaceutical Cocrystals and Their Physicochemical Properties. In *Crystal Growth and Design* 9.6 (2009).

[142] Jie Chen et al. Pharmaceutical Crystallization. In *Crystal growth & design* 11.4 (2011).

[143] Xiaotao Zhang, Huanli Dong and Wenping Hu. Organic Semiconductor Single Crystals for Electronics and Photonics. In *Advanced Materials* 30.44 (2018).

[144] Yu Wang et al. Organic Crystalline Materials in Flexible Electronics. In *Chemical Society Reviews* 48.6 (2019).

[145] Eberhard Engel and Reiner M Dreizler. *Density Functional Theory*. Springer, 2011.

[146] Carl Ivar Branden and John Tooze. *Introduction to Protein Structure*. Garland Science, 2012.

[147] David H Case et al. Convergence Properties of Crystal Structure Prediction by Quasi-Random Sampling. In *Journal of Chemical Theory and Computation* 12.2 (2016).

[148] Chris J Pickard and RJ Needs. Ab Initio Random Structure Searching. In *Journal of Physics: Condensed Matter* 23.5 (2011).

[149] Rithwik Tom et al. Genarris 2.0: A Random Structure Generator for Molecular Crystals. In *Computer Physics Communications* 250 (2020).

[150] Bouke P van Eijck. Crystal structure predictions for disordered halobenzenes. In *Physical Chemistry Chemical Physics* 4.19 (2002).

[151] P Ganguly and Gautam R Desiraju. Long-Range Synthon Aufbau Modules (LSAM) in Crystal Structures: Systematic Changes in $C_6H_{6-n}F_n$ ($0 \leq n \leq 6$) Fluorobenzenes. In *CrystEngComm* 12.3 (2010).

[152] Luis Reinaudi, Ezequiel PM Leiva and Raúl E Carbonio. Simulated Annealing Prediction of the Crystal Structure of Ternary Inorganic Compounds Using Symmetry Restrictions. In *Dalton Transactions* 23 (2000).

[153] David J Earl and Michael W Deem. Parallel Tempering: Theory, Applications, and New Perspectives. In *Physical Chemistry Chemical Physics* 7.23 (2005).

[154] Farren Curtis et al. GAtor: A First-Principles Genetic Algorithm for Molecular Crystal Structure Prediction. In *Journal of Chemical Theory and Computation* 14.4 (2018).

[155] Andriy O Lyakhov et al. New Developments in Evolutionary Structure Prediction Algorithm USPEX. In *Computer Physics Communications* 184.4 (2013).

[156] Yanchao Wang et al. Crystal Structure Prediction via Particle-Swarm Optimization. In *Physical Review B: Condensed Matter and Materials Physics* 82.9 (2010).

[157] Atreyee Banerjee et al. Crystal Structure Prediction for Benzene using Basin-Hopping Global Optimization. In *The Journal of Physical Chemistry A* 125.17 (2021).

[158] Daniel S Levine et al. The Open Molecules 2025 (OMol25) Dataset, Evaluations, and Models. In *arXiv* (2025).

[159] Anuroop Sriram et al. The Open DAC 2023 Dataset and Challenges for Sorbent Discovery in Direct Air Capture. In *ACS Central Science* (2024).

[160] Luis Barroso-Luque et al. Open Materials 2024 (OMat24) Inorganic Materials Dataset and Models. In *arXiv* (2024).

[161] Justin S Smith et al. The ANI-1ccx and ANI-1x Data Sets, Coupled-Cluster and Density Functional Theory Properties for Molecules. In *Scientific Data* 7.1 (2020).

[162] Johannes Gasteiger, Florian Becker and Stephan Günnemann. Gemnet: Universal Directional Graph Neural Networks for Molecules. In *NeurIPS*. 2021.

[163] Johannes Gasteiger et al. GemNet-OC: Developing Graph Neural Networks for Large and Diverse Molecular Simulation Datasets. In *Transactions on Machine Learning Research* (2022).

[164] Ilyes Batatia et al. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. In *NeurIPS*. 2022.

[165] Ilyes Batatia et al. A Foundation Model for Atomistic Materials Chemistry. In *The Journal of Chemical Physics* 163.18 (2025).

[166] Brandon M Wood et al. UMA: A Family of Universal Models for Atoms. In *NeurIPS*. 2025.

[167] Amil Merchant et al. Scaling Deep Learning for Materials Discovery. In *Nature* 624.7990 (2023).

[168] Vahe Gharakhanyan et al. FastCSP: Accelerated Molecular Crystal Structure Prediction with Universal Model for Atoms. In *arXiv* (2025).

[169] Rui Jiao et al. Space Group Constrained Crystal Generation. In *ICLR*. 2024.

[170] Daniel Levy et al. SymmCD: Symmetry-Preserving Crystal Generation with Diffusion Models. In *ICLR*. 2025.

[171] Luis M Antunes, Keith T Butler and Ricardo Grau-Crespo. Crystal Structure Generation with Autoregressive Large Language Modeling. In *Nature Communications* 15.1 (2024).

[172] Qianggang Ding, Santiago Miret and Bang Liu. MatExpert: Decomposing Materials Discovery By Mimicking Human Experts. In *ICLR*. 2025.

[173] Zeming Lin et al. Evolutionary-Scale Prediction of Atomic-Level Protein Structure with a Language Model. In *Science* 379.6637 (2023).

[174] Joseph L Watson et al. De Novo Design of Protein Structure and Function with RFdiffusion. In *Nature* 620.7976 (2023).

[175] Jason Yim et al. SE(3) Diffusion Model with Application to Protein Backbone Generation. In *ICML*. 2023.

[176] Hanqing Zeng et al. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*. 2020.

[177] Philipp Pracht, Fabian Bohle and Stefan Grimme. Automated Exploration of the Low-Energy Chemical Space with Fast Quantum Chemical Methods. In *Physical Chemistry Chemical Physics* 22.14 (2020).

[178] Tero Karras et al. Elucidating the Design Space of Diffusion-Based Generative Models. In *NeurIPS*. 2022.

[179] ByteDance AML AI4Science Team et al. Protenix-Advancing Structure Prediction through a Comprehensive AlphaFold3 Reproduction. In *bioRxiv* (2025).

[180] Valerio Mariani et al. lDDT: A Local Superposition-Free Score for Comparing Protein Structures and Models using Distance Difference Tests. In *Bioinformatics* 29.21 (2013).

[181] Rajarshi Guha et al. The Blue Obelisk—interoperability in chemical informatics. In *Journal of Chemical Information and Modeling* 46.3 (2006).

[182] Aaron J Nessler et al. Progressive Alignment of Crystals: Reproducible and Efficient Assessment of Crystal Structure Similarity. In *Applied Crystallography* 55.6 (2022).

[183] DW Hoffmann et al. A test of crystal structure prediction of small organic molecules. In *Acta Crystallographica. Section B, Structural science Structural crystallography and crystal chemistry* (2000).

[184] WD Sam Motherwell et al. Crystal structure prediction of small organic molecules: a second blind test. In *Structural Science* 58.4 (2002).

[185] GM Day et al. A third blind test of crystal structure prediction. In *Structural Science* 61.5 (2005).

[186] Graeme M Day et al. Significant progress in predicting the crystal structures of small organic molecules–a report on the fourth blind test. In *Structural Science* 65.2 (2009).

[187] Hongyu Guo, Yoshua Bengio and Shengchao Liu. Assembleflow: Rigid Flow Matching with Inertial Frames for Molecular Assembly. In *ICLR*. 2025.

[188] George Cybenko. Approximation by superpositions of a sigmoidal function. In *Mathematics of Control, Signals and Systems* 2.4 (1989).

[189] Kurt Hornik, Maxwell Stinchcombe and Halbert White. Multilayer feedforward networks are universal approximators. In *Neural Networks* 2.5 (1989).

[190] Pablo Barceló et al. The Logical Expressiveness of Graph Neural Networks. In *ICLR*. 2020.

[191] Billy J Franks et al. Weisfeiler-leman at the margin: When more expressivity matters. In *arXiv* (2024).

[192] Weihua Hu et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*. 2020.

Appendices

A

Experimental Details for Chapter 3

Here, we provide the additional experimental details and hyperparameters for the results presented in Chapter 3.

Compute Resources. All homomorphism counting and experiments for ZINC, COLLAB, and BREC were run on a cluster with NVIDIA A10 GPUs (24 GB). Each node had 64 cores of Intel(R) Xeon(R) Gold 6326 CPU at 2.90GHz and ∼500GB of RAM. QM9 experiments were run on a cluster with NVIDIA A100 GPUs. All experiments used 1 GPU at a time.

A.1 Hyperparameter Protocol

A.1.1 Details for Encoding Experiments

All experiments are run with the GIN-E model from Hu et al. [109], that was implemented from scratch using the standard PyTorch Geometric library. All results are reported from running the model for 500 epochs, lr=0.001, a hidden dimension size equal to the size of the initial node features = 128, batch size = 128, d=10 (for normalization in PosHom and SubHom), and 4 layers.

A.1.2 Details for ZINC Experiments

For GAT, GCN, and GIN, the training is done using the following configuration: $optimizer = Adam$, $initial\_lr = 0.001$, $lr\_reduce\_factor = 0.5$, $minimum\_lr = 1e-5$, $patience = 10$. We implement all of these models from scratch using the standard PyTorch Geometric library, however we align with the formations from

Table A.1: Hyperparameters for ZINC experiments. Recall that d_h denotes the size of the hidden dimension.

Model	Layers	d_h.	Batch Size	Epochs	#Heads	Readout
GAT	4	18	128	1000	8	mean
	16	22	128	1000	8	mean
GCN	4	125	128	1000	N/A	mean
	16	172	128	1000	N/A	mean
GIN	4	110	128	1000	N/A	sum
	16	124	128	1000	N/A	sum
BasePlanE	3	128	256	500	N/A	sum

Dwivedi et al. [33] as much as possible. For BasePlanE, the training is done with the same configuration as listed for GAT, GCN, and GIN, but with a $patience = 30$ instead of 10. Note that fore BasePlanE, we build off of the implementation provided by Dimitrov et al. [85]. The remainder of the model hyperparameters are as listed in Table A.1. Results are reported as the average across 4 runs at different seeds.

A.1.3 Details for QM9 Experiments

For R-GCN, we build on the implementation provided by Abboud, Dimitrov and Ceylan [113], which uses the following hyperparameters from Brockschmidt [112] for the standard model: $batch\_size = 128$, $hidden\_dim = 128$, $layers = 8$, $lr = 0.001$, $aggregation = mean$, $epochs = 300$, and $optimizer = Adam$. We also use both batch and layer normalization and include residual connections at a frequency of every 2 layers.

For the R-GCN+FA experiments, we align with the R-GCN+FA hyperparameter setup from Brockschmidt [112], which uses: $batch\_size = 128$, $hidden\_dim = 128$, $layers = 8$, $lr = 0.000572$, $aggregation = sum$, $epochs = 400$, and $optimizer = RMSProps$. We use batch and layer normalization, as well as residual connections every 2 layers. All results for all 13 molecular properties are reported as the average across 5 runs at different seeds.

A.1.4 Details for COLLAB Experiments

For GCN and GraphSAGE, we use the exact same model and hyperparameters as the default example provided by Hu et al. [192]. For GAT, we construct a

Table A.2: Hyperparameters for COLLAB experiments, where d_h denotes the size of the hidden dimension, lr is learning rate, and h_{PE} denotes the size of the positional encoding dimension.

Model	Layers	d_h	Batch Size	lr	Epochs	d_{PE}	#Heads
GAT	3	256	65536	0.001	400	8	8
GCN	3	256	65536	0.001	400	8	N/A
GraphSAGE	3	256	65536	0.001	400	6	N/A

model that closely aligns with the overall framework of the GCN and GraphSAGE implementations using the standard PyTorch Geometric library. The specific model hyperparameter details are outlined in Table A.2. Results for all models are reported as the average across 4 runs at different seeds.

A.1.5 Details for BREC Experiments

For GIN+$\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq k})$, we use the exact same model construction and hyperparameters as was used to benchmark GSN on this dataset, and we build on the implementation provided by Wang and Zhang [116]. Specifically, we use: $layers = 4$, lr = 1e-4, $weight\_decay$ = 1e-5, $batch\_size$ = 16, $output\_dimension$ = 64, $epochs$ = 20. For PPGN+$\mathsf{Hom}(\Omega^{\mathsf{con}}_{\leq k})$, we slightly tweak the hyperparameters of the model used in the BREC benchmark by changing the dimensions of the blocks from 32 to 48 in order to accommodate our extra features. The rest of the hyperparameters remain the same, which are: $layers$ = 5, lr = 1e-4, $weight\_decay$ = 1e-4, $batch\_size$ = 32, $output\_dimension$ = 16, $epochs$ = 20.

A.2 Homomorphism Basis Graphs

Here, we provide the full set of graphs in the Spasm of C_6, C_7, and C_8, which we use as additional encodings. We also provide the set of graphs in the anchored Spasm of C_8 as an example of an anchored basis set.

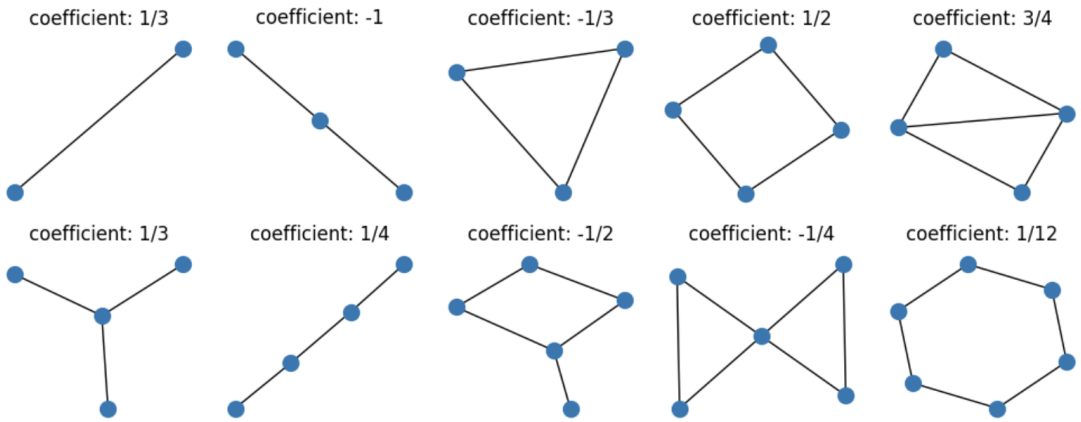


Figure A.1: Graphs and coefficients for $\mathsf{Spasm}(C_6)$.

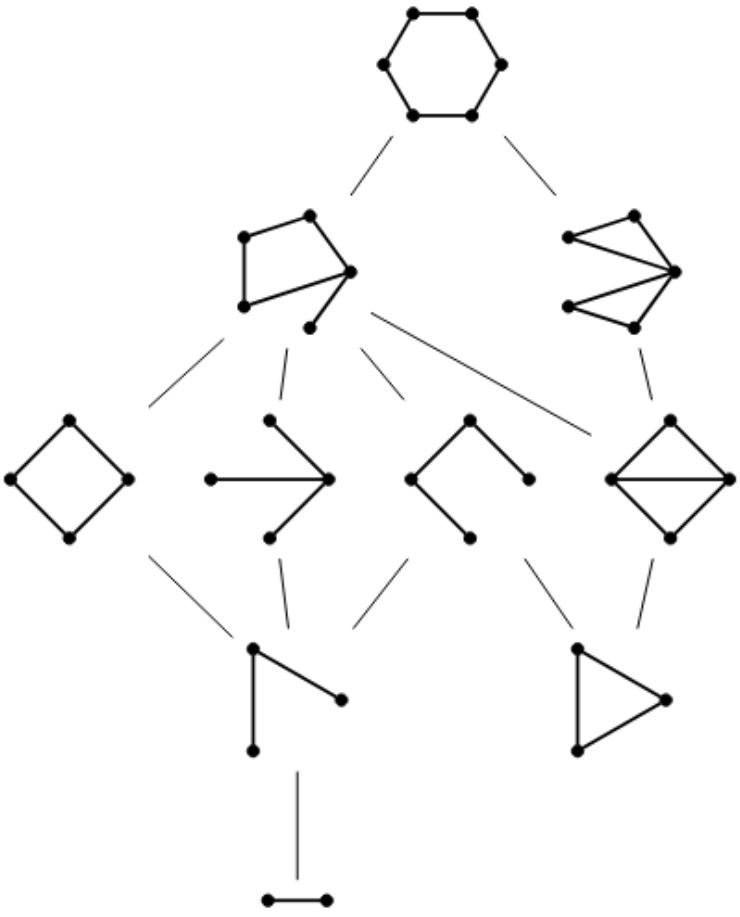


Figure A.2: Hasse diagram of the (loop-free) quotient relation for C_6.

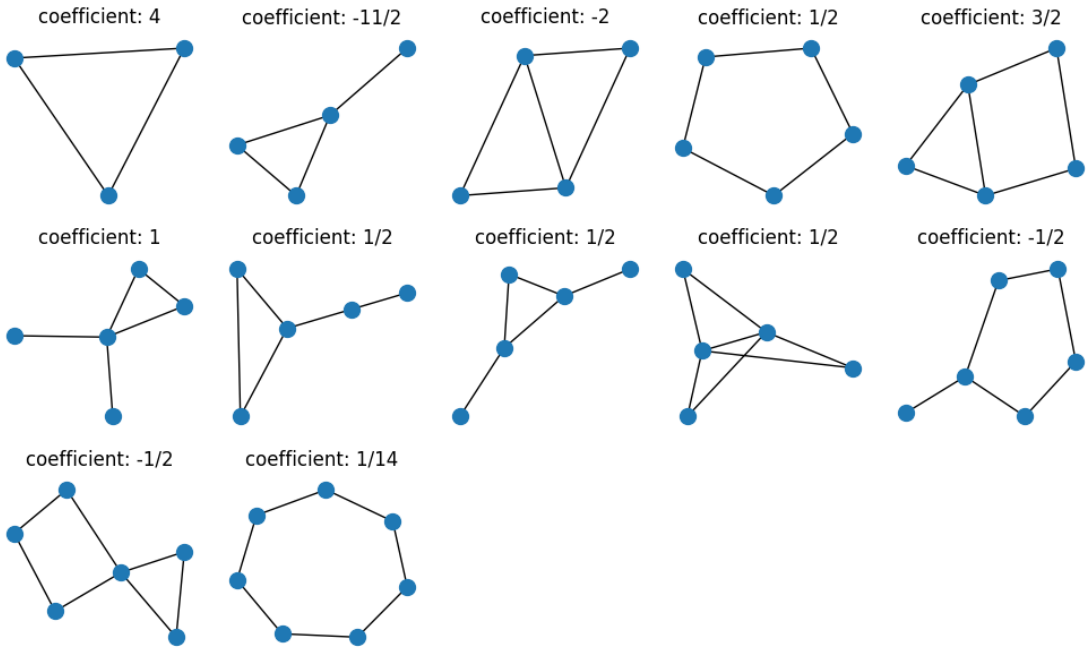


Figure A.3: Graphs and coefficients for $\mathsf{Spasm}(C_7)$.

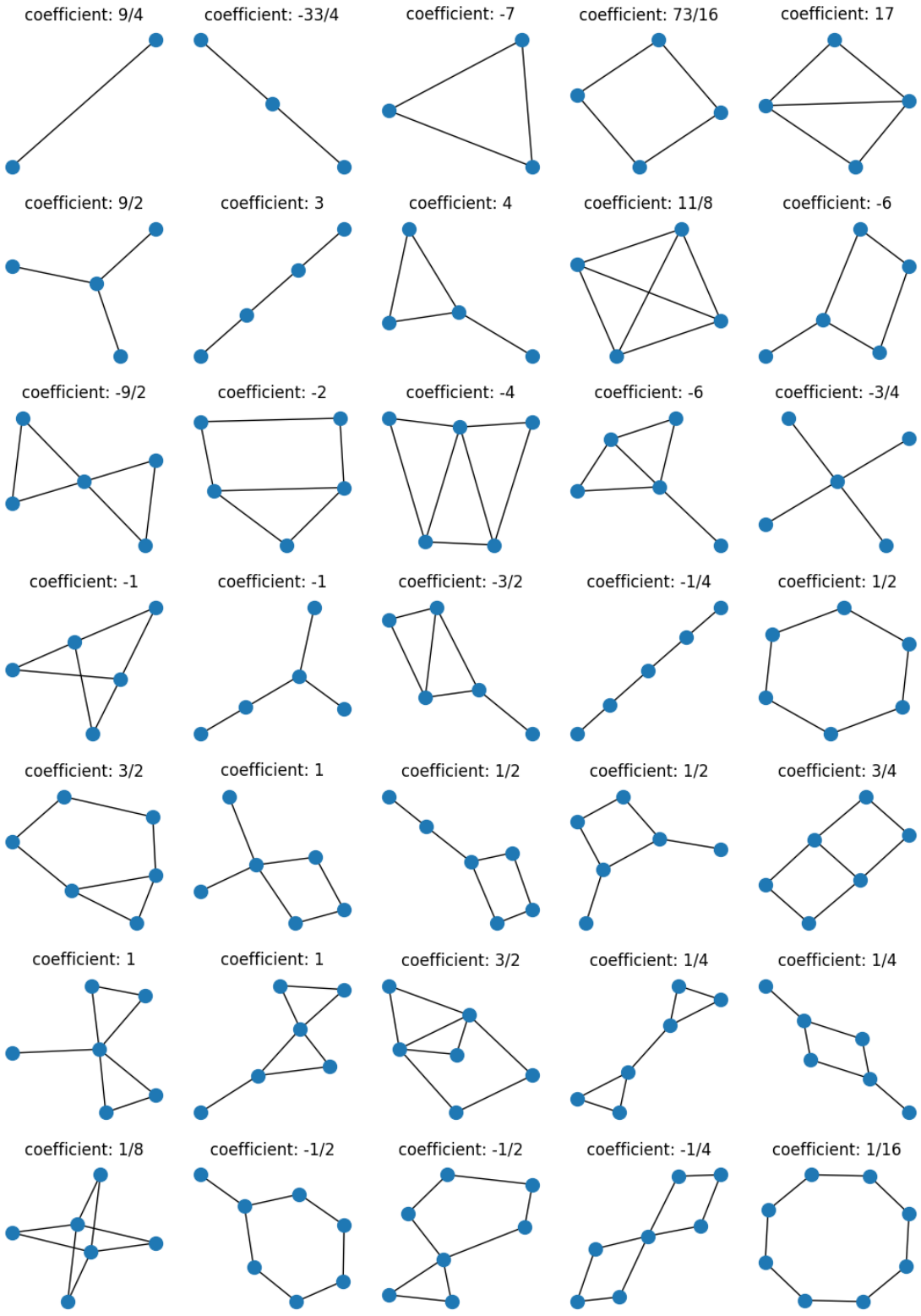


Figure A.4: Graphs and coefficients for $\mathsf{Spasm}(C_8)$.

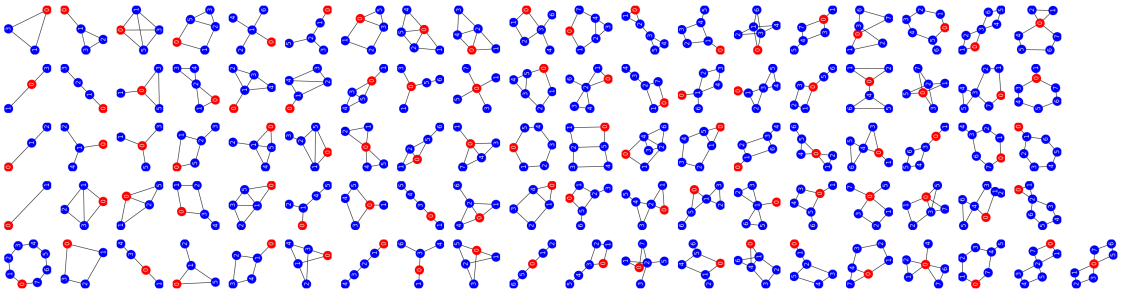


Figure A.5: Graphs in $\mathsf{Spasm}^{\circledast}(C_8)$.

(a) Untyped P_4 homomorphism basis (b) Typed P_4 homomorphism basis

Figure A.6: Different homomorphism bases for an untyped and typed path graph P_4. Note that in the typed version, there are two different version of the 2-hop path due to the different possible node type assignments. Additionally, the Spasm for the typed version does not contain a triangle graph, since the two end nodes are of different types, so they cannot be contracted. Analogous examples can be constructed for graphs with typed or directed edges.

B

Experimental Details for Chapter 4

Here, we provide the additional experimental details and hyperparameters for our results presented in Chapter 4.

Compute Resources. All experiments were conducted on a cluster with 12 NVIDIA A10 GPUs (24 GB) and 4 NVIDIA H100s. Each node had 64 cores of Intel(R) Xeon(R) Gold 6326 CPU at 2.90GHz and ∼500GB of RAM. All experiments used at most 1 GPU at a time.

B.1 Hyperparameter Protocol

In all models, a 2-layer MLP is used to embed the raw positional or structural encoding values. The dimension of this embedding scales according to the width of the main model being used. Unless otherwise specified, all models utilize global sum-pooling followed by a 3-layer MLP prediction head where each layer subsequently reduces the hidden dimension by a factor of 1/2 (other than the output layer, which has a dimension equal to the number of targets). Lastly, all models use the ReLU activation function.

B.1.1 Details for ZINC Experiments

Following Rampásek et al. [29], we constrain all models to a 500K parameter budget. We provide a rough summary of the final hyperparameters with different models using MoSE in Table B.1.

Table B.1: Hyperparameter summary for MoSE results on ZINC from Table 4.2. Here, d_h denotes hidden dimension size, lr denotes learning rate, and d_{MoSE} denotes the dimension of the MoSE encodings.

Model	Layers	d_h	Batch Size	lr	Epochs	d_{MoSE}	#Heads
MLP-E	8	64	32	0.001	1200	28	N/A
GIN-E	4	110	128	0.001	1000	42	N/A
GIN-E+VN	4	110	32	0.001	1000	42	N/A
GT-E	10	64	32	0.001	2000	28	4
GPS	8	64	32	0.001	2000	42	4

GRIT. For the GRIT results presented in Table 4.4, we simply remove the RRWP node and node-pair encodings from the configuration used in Ma et al. [30], and replace them with MoSE that is constructed from $\mathsf{Spasm}(C_7) \cup \mathsf{Spasm}(C_8)$. We summarize the hyperparameters for our GRIT+MoSE model as follows:

Layers: 10	*Width/Heads*: 64 / 8
Batch Size: 32	*Optimizer*: AdamW
Weight Decay: 1e-5	*Base LR*: 0.001
Max Epochs: 2000	*LR Scheduler*: Cosine
LR Warmup Epochs: 50	*Attention Dropout*: 0.2

B.1.2 Details for Peptides Experiments

We do not perform hyperparameter tuning on the Peptides dataset, using a 4-layer GT with width 36 for all encoding types (see Table B.2). Note that the original GT model without any encodings has a wider width of 96. Models for both Peptides-func and Peptides-struct tasks are trained for 200 epochs using the AdamW optimizer, a batch size of 128, and a base LR of 0.0003 (annealed with a cosine scheduler).

Table B.2: A summary of the GT hyperparameters used for the Peptides dataset.

Model	Layers	Width	Encoding dim
GT+none	4	96	N/A
GT+RWSE	4	36	16
GT+MoSE$_{\mathcal{S}}$ *(ours)*	4	36	24
GT+MoSE$_{\Omega}$ *(ours)*	4	36	16

B.1.3 Details for Fractional Domination Experiments

Similarly to the Peptides experiments, we do not perform hyperparameter tuning on the Fractional Domination synthetic dataset.

MLP. Our MLP experiments on the synthetic dataset (Table 4.11) use the following hyperparameters for all encodings:

Depth: 4	*Width*: 145	*Global Pooling*: Mean
Batch Size: 128	*Optimizer*: Adam	*Weight Decay*: 0.0
Base LR: 0.001	*Max Epochs*: 1000	*LR Scheduler*: Reduce on Plateau
LR Reduce Factor: 0.5	*Dropout*: 0.2	*LR Patience (min): 10 (1e-5)*

GT. Our GT experiments on the synthetic dataset (Table 4.11) use the following hyperparameters for all encodings:

Depth: 10	*Width*: 64	*Heads*: 4
Batch Size: 128	*Optimizer*: AdamW	*Weight Decay*: 0.001
Base LR: 0.0001	*Max Epochs*: 1200	*LR Scheduler*: Cosine
LR Warmup Epochs: 30	*Attention Dropout*: 0.5	

C

Experimental Details for Chapter 5

Here, we provide the additional experimental details and hyperparameters for our results presented in Chapter 5.

Compute Resources. Experiments were performed on heterogenous GPU clusters consisting of NVIDIA L40S and H100 GPUs. Models were primarily trained using multinode DDP training on L40S GPUs as cluster availability permitted. Inference is performed across all available hardware.

C.1 Rigid and Flexible Dataset Construction

The exact CSD identifiers for the rigid and flexible CSP datasets are provided below for reference.

Molecules in the Rigid Dataset: XULDUD, GUFJOG, QAMTAZ, BOQQUT, BOQWIN, UJIRIO, PAHYON, XATMIP, HAMTIZ, XATMOV, AXOSOW, SOXLEX, WIDBAO, HXACAN, ACSALA, PUBMUU, GLYCIN, TEPNIT, QQQCIG, ZZZMUC, FOYNEO, ANTCEN, URACIL, BTCOAC, CORONE, GUJTOX, TPHPOR, ACETAC, IMAZOL, CEBYUD, CILJIQ, CUMJOJ, DEZDUH, DOHFEM, GACGAU, GOLHIB, HURYUQ, IHEPUG, LECZOL, NICOAM, ROHBUL, UMIMIO, WEXREY, CAPRYL, ACACPD, BPYRUF, JIVNOV, MUBXAM, VUJBUB, NAVZAO.

Molecules in the Flexible Dataset: QAXMEH, DORDUM, BOTHUR, MOVZUW, YOPYEL, SUTHAZ, TPPRHC, FPAMCA, INDMET, MELFIT, YIGPIO, TEHZIP, DMANTL, YIGDUP, UWEQUL, COWZIA.

C.2 Hyperparameter Protocol

C.2.1 A-Transformer Hyperparameters

A-Transformer uses a model based on the PyTorch `nn.TransformerEncoder` module, with hidden size $d_h = 512$, $L = 13$ layers, and $H = 4$ heads.

C.2.2 OXtal Hyperparameters

Training. Our training implementation is based off of the open-source Protenix model [179]. Because we are not doing protein folding, we have disabled the previously built-in MSA and LM components. For training, we use an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a learning rate of 0.0018, and a weight decay of 1e-8. The learning rate is additionally set to a scheduler with 1,000 warmup steps and a decay factor of 0.95 for every 50,000 steps. We report results for our model trained at 70,000 and 110,000 steps.

The atom encoder has 3 blocks with 4 heads. The atom embedding size is 128, whereas the pair embedding is size is 16. In terms of the main pairformer trunk, we use 4 pairformer blocks, each with 16 attention heads and a dropout rate of 0.25. The hidden dimension for the pair representation is 128, and the hidden dimension for the single representations are 384. Furthermore, we do 2 rounds of recycling. For the diffusion block, we use a diffusion batch size of 32 and a chunk size of 1. The actual diffusion transformer has 12 transformer blocks, with 8 attention heads. Crop size (and therefore token size for the diffusion transformer) is set to 640, with a 50/50 ratio of S^4 and kNN cropping for training. All components are set to seed 42.

Inference. At inference time, we use the following hyperparameters for our diffusion sampling: $\gamma_0 = 0.8$, noise scale lambda = 1.003, steps = 200, and step scale eta = 1.5. When generating samples, we generate 1 sample from 30 different seeds for each target crystal structure. Seeds are set from 0 to 29.

C.3 Detailed CSP Blind Test Baselines

C.3.1 Performance Results

In the tables below, we provide the full baseline results for the 5th, 6th, and 7th CSP blind tests. Metrics are calculated based on the submitted structures provided in Bardwell et al. [37], Reilly et al. [38] and Hunnisett et al. [39], respectively.

Table C.1: Results per crystal of submitted methods for CSP blind test 5.

Model	n_S	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	Pac_C?	$\text{Rec}_S \uparrow$	Rec_C?	$\widetilde{\text{Sol}}_C$?
			Target XVI (OBEQUJ)				
Group Ammon	20	0.000	0.850	Y	1.000	Y	Y
Group Boerrigter	8456	0.000	0.113	Y	0.994	Y	Y
Group Day	549	0.000	0.222	Y	1.000	Y	Y
Group Desiraju	202	0.000	0.272	Y	1.000	Y	Y
Group Hofmann	103	0.000	0.000	N	1.000	Y	N
Group Jose	16	0.188	0.250	Y	0.438	Y	N
Group Kendrick	90	0.000	0.500	Y	1.000	Y	Y
Group Maleev	18	0.000	0.000	N	0.944	Y	N
Group Misquitta	103	0.000	0.214	Y	1.000	Y	Y
Group Nikylov	13	0.000	0.000	N	1.000	Y	N
Group Orendt	292	0.000	0.209	Y	1.000	Y	Y
Group Price	150	0.000	0.367	Y	1.000	Y	Y
Group Scheraga	61	0.000	0.393	Y	1.000	Y	Y
Group vanEijck	117	0.000	0.359	Y	1.000	Y	Y
Group Venuti	61	0.033	0.049	Y	0.475	Y	Y
			Target XVII (OBEQOD)				
Group Ammon	6	0.000	0.167	Y	1.000	Y	N
Group Boerrigter	4894	0.000	0.054	Y	0.999	Y	Y
Group Day	164	0.000	0.220	Y	1.000	Y	Y
Group Desiraju	202	0.000	0.302	Y	1.000	Y	Y
Group Hofmann	103	0.000	0.000	N	1.000	Y	N
Group Jose	15	0.533	0.133	Y	0.267	Y	N
Group Kendrick	150	0.000	0.200	Y	1.000	Y	Y
Group Maleev	24	0.000	0.083	Y	1.000	Y	N
Group Orendt	272	0.000	0.051	Y	0.971	Y	Y
Group Price	139	0.000	0.122	Y	1.000	Y	Y
Group Scheraga	74	0.000	0.189	Y	0.986	Y	Y
Group vanEijck	132	0.000	0.129	Y	0.992	Y	Y
Group Venuti	71	0.563	0.014	Y	0.211	Y	N
			Target XVIII (OBEQET)				
Group Ammon	5	0.000	0.000	N	0.600	Y	N
Group Boerrigter	5001	0.000	0.005	Y	0.448	Y	Y
Group Day	438	0.000	0.000	N	0.893	Y	N
Group Desiraju	156	0.000	0.038	Y	1.000	Y	Y
Group Hofmann	103	0.000	0.000	N	0.379	Y	N
Group Jose	11	0.364	0.000	N	0.000	N	N
Group Kendrick	79	0.000	0.127	Y	0.911	Y	Y
Group Maleev	10	0.000	0.000	N	0.800	Y	N
Group Orendt	165	0.000	0.012	Y	0.897	Y	Y
Group Price	259	0.000	0.008	Y	0.525	Y	Y
Group Scheraga	71	0.000	0.014	Y	0.986	Y	Y
Group vanEijck	113	0.000	0.000	N	0.000	N	N
Group Venuti	100	0.000	0.000	N	0.040	Y	N
			Target XIX (XATJOT)				
Group Boerrigter	383	0.000	0.355	Y	1.000	Y	Y
Group Day	103	0.000	0.252	Y	1.000	Y	Y
Group Desiraju	202	0.000	0.262	Y	1.000	Y	Y
Group Hofmann	103	0.000	0.155	Y	1.000	Y	Y
Group Kendrick	350	0.000	0.326	Y	1.000	Y	Y
Group Maleev	23	0.000	0.217	Y	1.000	Y	Y
Group Orendt	279	0.000	0.140	Y	1.000	Y	Y
Group Price	164	0.000	0.079	Y	1.000	Y	Y
Group Scheraga	34	0.000	0.147	Y	1.000	Y	Y
Group vanEijck	129	0.000	0.457	Y	1.000	Y	Y
Group Venuti	100	0.000	0.050	Y	1.000	Y	N
			Target XX (OBEQIX)				
Group Ammon	11	0.000	0.000	N	0.000	N	N
Group Boerrigter	5005	0.000	0.000	N	0.014	Y	N
Group Day	53	0.000	0.094	Y	0.358	Y	Y
Group Hofmann	103	0.971	0.000	N	0.000	N	N
Group Kendrick	117	0.000	0.205	Y	0.248	Y	Y
Group Maleev	8	0.000	0.000	N	0.000	N	N
Group Orendt	156	0.000	0.006	Y	0.006	Y	Y
Group Price	103	0.000	0.039	Y	0.243	Y	Y
Group vanEijck	121	0.000	0.000	N	0.017	Y	N
Group Venuti	100	0.020	0.000	N	0.000	N	N
			Target XXI (KONTIQ)				
Group Boerrigter	5002	0.000	0.902	Y	0.914	Y	Y
Group Day	2350	0.000	0.964	Y	0.966	Y	Y
Group Desiraju	244	0.000	0.943	Y	0.943	Y	Y
Group Hofmann	103	0.000	0.981	Y	0.981	Y	Y
Group Kendrick	1886	0.000	0.990	Y	0.991	Y	Y
Group Maleev	26	0.000	0.769	Y	0.769	Y	Y
Group Orendt	594	0.000	0.928	Y	0.941	Y	Y
Group Price	449	0.002	0.978	Y	0.980	Y	Y
Group vanEijck	132	0.000	0.985	Y	0.985	Y	Y
Group Venuti	294	0.000	0.568	Y	0.058	Y	Y

Table C.2: Results per crystal of submitted methods for CSP blind test 6.

Model	n_S	$\text{Col}_S \downarrow$	$\text{Pac}_S \uparrow$	Pac_C?	$\text{Rec}_S \uparrow$	Rec_C?	$\widetilde{\text{Sol}}_C$?
			Target XXII (NACJAF)				
Group01-Chadha-Singh	100	0.000	0.010	Y	1.000	Y	N
Group02-Cole	100	0.000	0.050	Y	1.000	Y	Y
Group03-Day	200	0.000	0.085	Y	1.000	Y	Y
Group04-Dzyabchenko	100	0.000	0.170	Y	1.000	Y	Y
Group05-vanEijck	100	0.000	0.070	Y	1.000	Y	Y
Group06-Elking-FustiMolnar	198	0.005	0.056	Y	0.939	Y	Y
Group07-vandenEnde-Cuppen	200	0.000	0.080	Y	1.000	Y	Y
Group08-Facelli	177	0.000	0.006	Y	0.977	Y	N
Group09-Goto-Obata	200	0.000	0.060	Y	1.000	Y	Y
Group10-Hofmann	100	0.000	0.000	N	1.000	Y	N
Group11-Lv-Wang-Ma	200	0.000	0.000	N	1.000	Y	N
Group12-Marom	600	0.028	0.032	Y	0.757	Y	Y
Group13-Mohamed	100	0.000	0.570	Y	1.000	Y	Y
Group14-Neumann-Kendrick-Leusen	100	0.000	0.070	Y	1.000	Y	Y
Group15-Adjiman-Pantelides	100	0.000	0.040	Y	1.000	Y	Y
Group16-Pickard-et-al	33	0.000	0.000	N	1.000	Y	N
Group17-Podeszwa	99	0.000	0.091	Y	1.000	Y	Y
Group18-Price	200	0.000	0.045	Y	1.000	Y	Y
Group19-Szalewicz-Price	100	0.000	0.030	Y	1.000	Y	Y
Group20-Tuckerman-Szalewicz	54	0.000	0.130	Y	1.000	Y	Y
Group21-Zhu	80	0.000	0.075	Y	0.988	Y	Y
Group22-Boese-Hofmann	31	0.000	0.000	N	1.000	Y	N
Group23-Brandenburg-Grimme	119	0.000	0.042	Y	0.992	Y	Y
Group25-Tkatchenko	120	0.000	0.067	Y	0.992	Y	Y
			Target XXIII (XAFPAY)				
Group01-Chadha-Singh	100	0.000	0.000	N	0.000	N	N
Group02-Cole	100	0.000	0.050	Y	0.090	Y	Y
Group03-Day	200	0.000	0.040	Y	0.035	Y	Y
Group05-vanEijck	100	0.000	0.040	Y	0.060	Y	Y
Group06-Elking-FustiMolnar	149	0.000	0.114	Y	0.094	Y	Y
Group07-vandenEnde-Cuppen	200	0.000	0.000	N	0.000	N	N
Group08-Facelli	100	0.000	0.000	N	0.000	N	N
Group09-Goto-Obata	200	0.000	0.120	Y	0.135	Y	Y
Group10-Hofmann	100	0.000	0.000	N	0.000	N	N
Group13-Mohamed	100	0.000	0.090	Y	0.350	Y	Y
Group14-Neumann-Kendrick-Leusen	200	0.000	0.170	Y	0.270	Y	Y
Group15-Adjiman-Pantelides	100	0.000	0.100	Y	0.120	Y	Y
Group18-Price	200	0.000	0.120	Y	0.165	Y	Y
Group21-Zhu	60	0.000	0.083	Y	0.133	Y	Y
Group22-Boese-Hofmann	18	0.000	0.000	N	0.000	N	N
Group23-Brandenburg-Grimme	125	0.000	0.400	Y	0.304	Y	Y
Group25-Tkatchenko	50	0.000	0.280	Y	0.220	Y	Y
			Target XXIV (XAFQON)				
Group03-Day	200	0.980	1.000	Y	0.020	Y	Y
Group05-vanEijck	100	0.000	0.980	Y	0.980	Y	Y
Group06-Elking-FustiMolnar	198	0.000	0.859	Y	0.005	Y	Y
Group08-Facelli	100	0.280	0.990	Y	0.720	Y	Y
Group10-Hofmann	100	0.000	0.990	Y	1.000	Y	Y
Group14-Neumann-Kendrick-Leusen	100	0.310	1.000	Y	0.690	Y	Y
Group18-Price	200	0.805	0.990	Y	0.195	Y	Y
Group21-Zhu	50	0.140	1.000	Y	0.860	Y	Y
Group22-Boese-Hofmann	15	0.200	1.000	Y	0.800	Y	Y
Group23-Brandenburg-Grimme	119	0.941	0.966	Y	0.059	Y	Y
Group24-Szalewicz	100	0.800	1.000	Y	0.200	Y	Y
Group25-Tkatchenko	50	0.600	1.000	Y	0.400	Y	Y
			Target XXV (XAFQAZ01)				
Group02-Cole	100	0.000	0.000	N	1.000	Y	N
Group03-Day	200	0.000	0.100	Y	1.000	Y	Y
Group04-Dzyabchenko	161	0.000	0.056	Y	1.000	Y	Y
Group05-vanEijck	100	0.000	0.060	Y	1.000	Y	Y
Group06-Elking-FustiMolnar	127	0.000	0.087	Y	0.984	Y	Y
Group07-vandenEnde-Cuppen	200	0.000	0.000	N	0.990	Y	N
Group08-Facelli	100	0.000	0.040	Y	1.000	Y	N
Group09-Goto-Obata	200	0.000	0.030	Y	1.000	Y	Y
Group10-Hofmann	100	0.000	0.000	N	1.000	Y	N
Group13-Mohamed	100	0.000	0.020	Y	1.000	Y	Y
Group14-Neumann-Kendrick-Leusen	100	0.000	0.100	Y	1.000	Y	Y
Group15-Adjiman-Pantelides	100	0.000	0.090	Y	1.000	Y	Y
Group18-Price	200	0.000	0.110	Y	1.000	Y	Y
Group21-Zhu	25	0.000	0.040	Y	1.000	Y	Y
Group22-Boese-Hofmann	10	0.000	0.000	N	1.000	Y	N
Group23-Brandenburg-Grimme	100	0.000	0.110	Y	1.000	Y	Y
Group25-Tkatchenko	20	0.000	0.100	Y	1.000	Y	Y
			Target XXVI (XAFQIH)				
Group01-Chadha-Singh	100	0.000	0.000	N	0.000	N	N
Group02-Cole	100	0.000	0.010	Y	0.010	Y	Y
Group03-Day	200	0.000	0.015	Y	0.000	N	Y
Group04-Dzyabchenko	71	0.000	0.014	Y	0.000	N	Y
Group05-vanEijck	100	0.000	0.000	N	0.000	N	N
Group06-Elking-FustiMolnar	138	0.000	0.312	Y	0.246	Y	Y
Group10-Hofmann	100	0.000	0.000	N	0.000	N	N
Group13-Mohamed	100	0.000	0.000	N	0.000	N	N
Group14-Neumann-Kendrick-Leusen	200	0.000	0.405	Y	0.490	Y	Y
Group15-Adjiman-Pantelides	100	0.000	0.020	Y	0.000	N	Y
Group18-Price	200	0.000	0.035	Y	0.040	Y	Y
Group21-Zhu	30	0.000	0.000	N	0.000	N	N
Group22-Boese-Hofmann	15	0.000	0.000	N	0.000	N	N
Group23-Brandenburg-Grimme	103	0.000	0.019	Y	0.058	Y	Y

Table C.3: Results per crystal of submitted methods for CSP blind test 7.

Model	n_S	$\mathrm{Col}_S \downarrow$	$\mathrm{Pac}_S \uparrow$	Pac_C?	$\mathrm{Rec}_S \uparrow$	Rec_C?	$\overline{\mathrm{Sol}}_C$?
Target XXVII (XIFZOF01)							
Group06-BEijck	1500	0.000	0.011	Y	0.006	Y	Y
Group08-DWMHofmann	1501	0.000	0.001	Y	0.000	N	N
Group10-XtalPi	1500	0.000	0.062	Y	0.038	Y	Y
Group16-Marom-Isayev	1500	0.000	0.027	Y	0.005	Y	Y
Group17-Matsui	1500	0.000	0.005	Y	0.005	Y	Y
Group19-OpenEye	1500	0.000	0.000	N	0.000	N	N
Group20-MNeumann	1500	0.000	0.015	Y	0.007	Y	Y
Group21-Obata-Goto	1500	0.000	0.011	Y	0.003	Y	Y
Group22-AOganov	1500	0.000	0.014	Y	0.000	N	Y
Group24-SLPrice	1500	0.000	0.005	Y	0.000	N	Y
Group25-CShang	1500	0.000	0.061	Y	0.007	Y	Y
Group27-KSzalewicz-MTuckerman	1500	0.000	0.027	Y	0.000	N	Y
Group28-QZhu	1500	0.000	0.000	N	0.000	N	N
Target XXVIII (OJIGOG01)							
Group06-BEijck	1500	0.076	0.002	Y	0.000	N	Y
Group08-DWMHofmann	31	1.000	0.516	Y	0.000	N	N
Group10-XtalPi	1500	1.000	0.141	Y	0.000	N	N
Group20-MNeumann	1500	0.947	0.127	Y	0.008	Y	N
Group22-AOganov	811	0.335	0.002	Y	0.000	N	N
Group24-SLPrice	1500	1.000	0.093	Y	0.000	N	N
Group25-CShang	1500	0.993	0.359	Y	0.005	Y	Y
Group26-KSzalewicz	1500	1.000	0.000	N	0.000	N	N
Target XXIX (FASMEV)							
Group01-Adjiman-Pantelides	1510	0.000	0.072	Y	0.961	Y	Y
Group03-DBoese	1510	0.000	0.060	Y	0.836	Y	Y
Group05-GDay	1510	0.000	0.143	Y	0.981	Y	Y
Group06-BEijck	1510	0.000	0.052	Y	0.668	Y	Y
Group10-XtalPi	1510	0.000	0.223	Y	0.997	Y	Y
Group11-Johnson-Otero-de-la-Roza	319	0.000	0.245	Y	0.997	Y	Y
Group12-JJose	2096	0.000	0.147	Y	0.000	N	N
Group13-DKhakimov	80	0.000	0.537	Y	1.000	Y	Y
Group16-Marom-Isayev	1510	0.000	0.476	Y	1.000	Y	Y
Group18-SMohamed	1510	0.000	0.054	Y	0.984	Y	Y
Group19-OpenEye	904	0.000	0.043	Y	0.893	Y	Y
Group20-MNeumann	1504	0.000	0.418	Y	0.999	Y	Y
Group21-Obata-Goto	1510	0.000	0.079	Y	0.738	Y	Y
Group22-AOganov	1510	0.000	0.054	Y	0.719	Y	Y
Group23-CJPickard	1510	0.001	0.154	Y	0.994	Y	Y
Group24-SLPrice	1265	0.000	0.043	Y	0.999	Y	Y
Group25-CShang	1500	0.000	0.159	Y	0.988	Y	Y
Group27-KSzalewicz-MTuckerman	1510	0.000	0.175	Y	0.972	Y	Y
Group28-QZhu	209	0.000	0.096	Y	0.947	Y	Y
Target XXX - 2:1 (MIVZEA)							
Group05-GDay	1600	0.000	0.007	Y	0.379	Y	Y
Group06-BEijck	1600	0.000	0.007	Y	0.239	Y	Y
Group10-XtalPi	1600	0.000	0.042	Y	0.186	Y	Y
Group12-JJose	403	0.000	0.000	N	0.432	Y	N
Group13-DKhakimov	281	0.000	0.000	N	1.000	Y	N
Group18-SMohamed	1600	0.000	0.002	Y	1.000	Y	Y
Group19-OpenEye	1600	0.000	0.001	Y	0.052	Y	Y
Group20-MNeumann	1600	0.000	0.042	Y	0.207	Y	Y
Group21-Obata-Goto	1600	0.000	0.002	Y	0.541	Y	Y
Group22-AOganov	1138	0.000	0.000	N	0.149	Y	N
Group24-SLPrice	1600	0.000	0.000	N	0.269	Y	N
Group27-KSzalewicz-MTuckerman	1600	0.000	0.001	Y	0.287	Y	Y
Group28-QZhu	1600	0.000	0.001	Y	0.070	Y	Y
Target XXX - 1:1 (MIVZIE)							
Group05-GDay	1600	0.000	0.016	Y	0.099	Y	Y
Group06-BEijck	1600	0.000	0.004	Y	0.041	Y	Y
Group10-XtalPi	1600	0.000	0.024	Y	0.107	Y	Y
Group12-JJose	403	0.000	0.000	N	0.000	N	N
Group13-DKhakimov	281	0.000	0.007	Y	0.000	N	N
Group18-SMohamed	1600	0.000	0.000	N	0.000	N	N
Group19-OpenEye	1600	0.000	0.001	Y	0.046	Y	N
Group20-MNeumann	1600	0.000	0.015	Y	0.111	Y	Y
Group21-Obata-Goto	1600	0.000	0.006	Y	0.001	Y	Y
Group22-AOganov	1138	0.000	0.001	Y	0.000	N	N
Group24-SLPrice	1600	0.000	0.001	Y	0.000	N	N
Group27-KSzalewicz-MTuckerman	1600	0.000	0.001	Y	0.016	Y	N
Group28-QZhu	1600	0.000	0.003	Y	0.083	Y	Y
Target XXXI (ZEHFUR)							
Group01-Adjiman-Pantelides	1500	0.000	0.029	Y	0.330	Y	Y
Group03-DBoese	1500	0.000	0.029	Y	0.266	Y	Y
Group05-GDay	1500	0.005	0.055	Y	0.351	Y	Y
Group06-BEijck	1500	0.000	0.008	Y	0.029	Y	Y
Group08-DWMHofmann	1500	0.000	0.004	Y	0.199	Y	Y
Group10-XtalPi	1500	0.000	0.055	Y	0.513	Y	Y
Group12-JJose	1500	0.001	0.025	Y	0.000	N	N
Group16-Marom-Isayev	1500	0.000	0.063	Y	0.405	Y	Y
Group18-SMohamed	1500	0.000	0.001	Y	0.000	N	N
Group19-OpenEye	1500	0.000	0.030	Y	0.090	Y	Y
Group20-MNeumann	1500	0.000	0.123	Y	0.636	Y	Y
Group21-Obata-Goto	1500	0.000	0.006	Y	0.137	Y	Y
Group22-AOganov	1500	0.000	0.003	Y	0.098	Y	Y
Group24-SLPrice	1500	0.000	0.017	Y	0.244	Y	Y
Group25-CShang	1500	0.000	0.069	Y	0.678	Y	Y
Group27-KSzalewicz-MTuckerman	1500	0.000	0.009	Y	0.073	Y	Y
Group28-QZhu	1500	0.000	0.001	Y	0.045	Y	N
Target XXXII (JEKVII)							
Group01-Adjiman-Pantelides	1499	0.006	0.001	Y	0.000	N	N
Group03-DBoese	1500	0.000	0.000	N	0.000	N	N
Group05-GDay	1500	0.000	0.000	N	0.000	N	N
Group06-BEijck	1500	0.000	0.000	N	0.000	N	N
Group10-XtalPi	1500	0.001	0.030	Y	0.020	Y	Y
Group18-SMohamed	203	0.000	0.000	N	0.000	N	N
Group19-OpenEye	1500	0.000	0.000	N	0.000	N	N
Group20-MNeumann	1500	0.000	0.134	Y	0.066	Y	Y
Group22-AOganov	1500	0.000	0.000	N	0.000	N	N
Group24-SLPrice	1500	0.000	0.000	N	0.000	N	N
Group25-CShang	1500	0.000	0.085	Y	0.014	Y	Y
Group27-KSzalewicz-MTuckerman	1500	0.000	0.000	N	0.000	N	N
Group28-QZhu	1495	0.000	0.000	N	0.000	N	N
Target XXXIII (ZEGWAN)							
Group01-Adjiman-Pantelides	1500	0.000	0.011	Y	0.429	Y	Y
Group05-GDay	1500	0.000	0.013	Y	0.623	Y	Y
Group06-BEijck	1500	0.000	0.015	Y	0.338	Y	Y
Group08-DWMHofmann	1500	0.000	0.004	Y	0.522	Y	Y
Group10-XtalPi	1500	0.000	0.029	Y	0.797	Y	Y
Group13-DKhakimov	56	0.000	0.000	N	0.000	N	N
Group19-OpenEye	1500	0.000	0.022	Y	0.758	Y	Y
Group20-MNeumann	1500	0.000	0.052	Y	0.729	Y	Y
Group21-Obata-Goto	1500	0.000	0.010	Y	0.553	Y	Y
Group22-AOganov	1500	0.000	0.000	N	0.008	Y	N
Group24-SLPrice	1500	0.000	0.013	Y	0.363	Y	Y
Group25-CShang	1500	0.000	0.009	Y	0.545	Y	Y
Group27-KSzalewicz-MTuckerman	1500	0.000	0.009	Y	0.231	Y	Y
Group28-QZhu	1453	0.000	0.012	Y	0.551	Y	Y

C.3.2 Inference Cost

In the tables below, we provide the reported inference time costs for models that competed in the the 5th, 6th, and 7th CSP blind tests. Note that these results are directly compiled from Bardwell et al. [37], Reilly et al. [38] and Hunnisett et al. [39]. We note that some models did not report their inference time, and thus are not included in these tables.

Table C.4: Summary of computational resources used by some of the participants as reported in the 5th CSP blind test. CPU hours are approximately normalized to 3.0 GHz. OXtal times are reported as elapsed wall time in hours on 1 L40S GPU and 6 CPUs.

Group	XVI	XVII	XVIII	XIX	XX	XXI	Total
Boerrigter	90	100	350	650	2,105	600	3,800
Day, Cruz-Cabeza	110	1,941	21,051	6,097	54,090	22,197	91,400
Desiraju, Thakur, Tiwari, Pal	114	2,303	324	114		1,431	4,600
Hofmann	2	7	12	694	670	187	1,600
Neumann, Leusen, Kendrick, van de Streek							115,000
Price et al.	200	5,000	14,000	3,000	120,000	52,800	195,000
Van Eijck	27						9,500
Della Valle, Venuti							3,200
Maleev, Zhitkov							7,500
Misquitta, Pickard & Needs							162,000
Scheraga, Arnautova	150	150	610	720			1,300
OXtal	0.024	0.011	0.012	0.013	0.029	0.011	0.100

Table C.5: Summary of the computational resources used by each submission in terms of raw CPU hours as reported in the 6th CSP blind test. OXtal times are reported as elapsed wall time in hours on 1 L40S GPU and 6 CPUs. Note that Group 22 re-ranks structures submitted by Group 10, and Groups 23, 24, and 25 re-rank structures submitted by Group 18.

Group	XXII	XXIII	XXIV	XXV	XXVI	Total
01-Chadha & Singh	350	450			600	1,400
02-Cole et al.	6	538		46	246	836
03-Day et al.	12,714	394,948	15,241	121,701	179,897	724,501
04-Dzyabchenko	144	3,648	3,360			7,152
05-van Eijck	130	2,810	1,400	8,060	7,630	20,030
06-Elking & Fusti-Molnar	418,540	242,000	235,400	135,000	190,000	1,220,940
07-van den Ende, Cuppen et al.	9,741	7,777		6,388	23,906	
08-Facelli et al.	268,012	38,500	11,500	39,000		357,012
09-Obata & Goto	19,200	346,000		325,000		690,200
10-Hofmann & Kuleshova	10	630	623	202	255	1,720
11-Lv, Wang, Ma	325,000					325,000
12-Marom et al.	30,000,000					30,000,000
13-Mohamed	26	106		81	61	274
14-Neumann, Kendrick, Leusen	32,160	146,120	103,700	84,680	356,844	723,504
15-Pantelides, Adjiman et al.	333	87,000		37,535	272,500	397,368
16-Pickard et al.	380,000					380,000
17-Podeszwa et al.	72,220					72,220
18-Price et al.	26,000	84,000	63,000	169,000	327,000	669,000
19-Szalewicz et al.	66,000					66,000
20-Tuckerman, Szalewicz et al.	81,000					81,000
21-Zhu, Oganov, Masunov	4,000	275,000	279,800	30,000	180,000	768,800
22-Boese	80,000	80,000	80,000	80,000	80,000	400,000
23-Brandenburg & Grimme	13,665	8,661	3,509	34,824	10,135	70,794
24-Szalewicz et al.			15,000			15,000
25-Tkatchenko et al.	100,000	2,100,000	500,000	500,000		3,200,000
OXtal	0.012	0.019	0.011	0.030	0.042	0.114

Table C.6: CPU core hours per target molecule for each prediction method as reported in the 7th CSP blind test. OXtal times are reported as elapsed wall time in hours on 1 L40S GPU and 6 CPUs.

Group	XXVII	XXVIII	XXIX	XXX	XXXI	XXXII	XXXIII	Total
01-Adjiman-Pantelides			652,495		840,000	1,597,000	412,000	3,501,495
03-DBoese			1,600,000		1,500,000	3,600,000		6,700,000
05-GDay	768,766		33,000	2,900,000	510,563	846,698	228,957	5,287,984
06-BEijck	8,120	1,350	1,310	9,800	1,470	2,900	4,980	29,930
08-DWMHofmann	3,200	10			4,000		1,840	9,050
10-XtalPi	772,500	1,242,500	1,146,588	644,927	381,672	644,927	612,500	5,445,614
11-Johnson-Otero-de-la-Roza			643,882					643,882
12-JJose			20,000	80,000	20,000			120,000
13-DKhakimov			350	1,500			500	2,350
16-Marom-Isayev	1,700,000		2,128,000		630,000			4,458,000
17-Matsui	95,819							95,819
18-SMohamed			1,050	36,864	632	1,561		40,107
19-OpenEye	30,000		40,000	1,250,000	140,000	400,000	60,000	1,920,000
20-MNeumann	1,022,976	283,538	755,712	1,769,472	1,028,064	3,935,232	728,064	9,523,058
21-Obata-Goto	333,586		92,890	580,436	1,889,649		477,210	3,373,771
22-AOganov	20,000	2,000	15,000	180,000	20,000	25,000	25,000	287,000
23-CJPickard			10,000					10,000
24-SLPrice	450,290	89,666	76,541	100,000	49,177	244,520	123,427	1,133,621
25-CShang	55,150	29,691	4,784		6,476	76,161	34,648	206,910
26-KSzalewicz					28,332			28,332
27-KSzalewicz-MTuckerman	1,280,566	60,457	242,424	213,722		1,663,940	150,650	3,611,759
28-QZhu	1,600		1,500	7,680	1,500	1,500	1,500	15,280
OXtal	0.060	0.026	0.011	0.056	0.015	0.050	0.017	0.235