Informatics

Graph Repräsentationen und Architekturen neuronaler Netzwerke für die Vorhersage von Aktivierungsenergien

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Jasper De Landsheere, MSc
Matrikelnummer 12228451

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing.(BA) Dr.rer.nat. Thomas Gärtner, MSc
Mitwirkung: Assistant Prof. Dr.in rer.nat. Esther Heid, MSc

Wien, 5. Mai 2025

[redacted] Jasper De Landsheere

[redacted] Thomas Gärtner

Informatics

Graph Representations and Neural Network Architectures for Reaction Barrier Height Prediction

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Jasper De Landsheere, MSc
Registration Number 12228451

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing.(BA) Dr.rer.nat. Thomas Gärtner, MSc
Assistance: Assistant Prof. Dr.in rer.nat. Esther Heid, MSc

Vienna, May 5, 2025

[redacted] [redacted]

Jasper De Landsheere   Thomas Gärtner

Erklärung zur Verfassung der Arbeit

Jasper De Landsheere, MSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 5. Mai 2025

[redacted]

Jasper De Landsheere

Acknowledgements

First and foremost, I am thankful to my thesis supervisors Thomas Gärtner and Esther Heid for their support, ideas, and dedicated time. I want to specifically thank Esther for letting me work on this amazing line of research and truly guiding me along the way. I am excited to continue this research journey as a PhD student in Esther's Machine Learning for Reactions group, and I look forward to contributing to its promising future.

Next, I want to show gratitude to all my friends and colleagues who have supported me throughout this journey. Specifically, I want to thank: Joan Salvà, Yannik Gaebel, Rebeka Angyal, Leonard Galustian, Johannes Karwounopoulos, Maximilian Kovar, Konstantin Mark, Leon Ganser, and the Madsen group.

Finally, I am deeply thankful to my family for giving me the freedom to pursue my dreams and ambitions.

Kurzfassung

Bei einer chemischen Reaktion handelt es sich um einen Vorgang, bei dem eine oder mehrere Verbindungen unter Freisetzung oder Zugabe von Energie umgewandelt werden. Die Energie, die dafür aufgebracht werden muss wird als Aktivierungsenergie (Barrier Height) bezeichnet und ist eine wichtige chemische Größe, die Aussagen darüber machen kann, ob eine Reaktion statt finden kann oder nicht. Deep-Learning-Ansätze zur Vorhersage von Aktivierungsenergien haben dabei vielversprechende Ergebnisse gezeigt, basieren jedoch typischerweise auf expliziten Atom-zu-Atom-Mapping-Informationen. Diese sind in realen Szenarien jedoch häufig nicht direkt verfügbar. Diese Arbeit untersucht systematisch Graph-Repräsentationen und Architekturen neuronaler Netzwerke zur Vorhersage von Aktivierungsenergien ohne die Abhängigkeit von explizitem Atom-Mapping. Wir zeigen, dass durch die Integration reaktionsspezifischer induktiver Biases in Architekturen neuronaler Netzwerke die Leistungslücke zwischen gemappten und nicht gemappten Repräsentationen erheblich verringert werden kann. Unsere beste gemappte Repräsentation, die Principal Neighbourhood Aggregation (PNA) unter Verwendung des Condensed Graph of Reaction (CGR), erreicht einen mittleren absoluten Fehler (Mean Absolute Error, MAE) von 4,32 ± 0,45 kcal/mol, während unser vorgeschlagener Reaction Graph Transformer (RGT), der ohne Atom-Mapping-Informationen auskommt, 6,18 ± 0,30 kcal/mol erreicht. Die Differenz verringert sich bei Datensätzen mit einem einzelnen Reaktionstyp noch weiter, was zeigt, dass reaktionsspezifische Architekturen effektiv fehlendes Mapping kompensieren können. In dieser Arbeit werden verschiedene Möglickeiten getestet, um Aktivierungsenergien ohne Atom-Mapping-Informationen effektiv vorherzugsagen. Dafür stellen wir eine flexible, modulare Codebasis für die Vorhersage von Reaktionseigenschaften vor, die Experimente mit verschiedenen Repräsentationen und Architekturen erleichtert.

Abstract

Chemical reactions are fundamental transformations that drive countless natural phenomena and technological applications, with their barrier heights, the minimum energy required for a reaction to potentially proceed, being crucial for understanding reactions. Deep learning approaches for predicting reaction barrier heights have shown promise but typically rely on explicit atom-to-atom mapping information, which is often unavailable in real-world scenarios. This thesis systematically explores graph representations and neural network architectures for predicting reaction barrier heights without relying on explicit atom mapping. We show that by incorporating reaction-specific inductive biases into neural network architectures, we can significantly reduce the performance gap between mapped and unmapped representations. Our best mapped-representation using Principal Neighbourhood Aggregation on the Condensed Graph of Reaction achieves a mean absolute error of 4.32 ± 0.45 kcal/mol, while our proposed Reaction Graph Transformer operating without atom mapping information reaches 6.18 ± 0.30 kcal/mol. The performance gap narrows even further on single reaction-type datasets, demonstrating that reaction-specific architectures can effectively compensate for missing mapping information. This work establishes a pathway toward more practical reaction property prediction tools that can operate effectively when atom mapping information is unavailable, enabling broader application in computational chemistry and drug discovery. Additionally, we contribute a flexible, modular code base for reaction property prediction that facilitates experimentation with different representations and architectures.

Contents

CHAPTER 1

Introduction

Chemical reactions transform substances into new materials, driving many natural phenomena and technological applications across different fields from pharmaceutical development to materials science, making their understanding important to scientific and industrial progress. At its core, a chemical reaction involves the breaking and forming of chemical bonds between atoms, resulting in the conversion of reactants into products. Figure 1.1 shows a simple type of chemical reaction, where a hydroxide ion (OH^-) attacks a carbon atom bonded to a bromine atom (Br). The bromine atom is then displaced, forming a new product where the hydroxide ion replaces the bromine atom, along with a bromide ion byproduct.

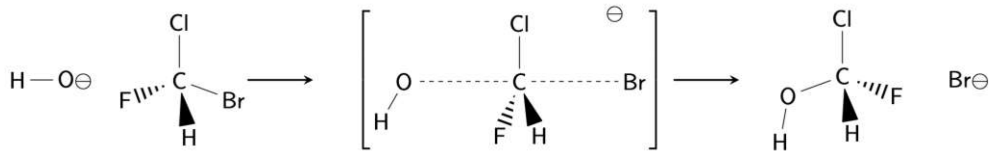


Figure 1.1: S_N2 reaction with a chiral reactant and product (courtesy of E. Heid).

While understanding the basic mechanism of a reaction is important, chemists are primarily concerned with its properties, such as the reaction yield[1], barrier[2], and rate[3]. These properties determine the efficiency and feasibility of a reaction, critical factors in both research and industrial applications. The reaction barrier height, the focus of this work, is a key property that represents the minimum energy required for a reaction to proceed. Predicting barrier heights accurately is important because it determines the feasibility of a reaction and guides process optimization for organic synthesis planning

[1]The amount of product formed relative to the amount of reactants used.
[2]The minimum energy required for a reaction to proceed.
[3]The speed at which a chemical reaction occurs.

[GPG20a]. However, due to the complex physical processes underlying these reactions, obtaining these properties often requires experimental or computational methods.

Deep learning, a subset of machine learning using multilayered neural networks, has recently emerged as a promising computational approach for predicting chemical reaction properties. The reproducibility of chemical experiments suggests a probabilistic relationship between a system's state and its reaction outcome, making deep learning an attractive tool for approximating this relationship without relying on expert knowledge. Recent advancements in predicting chemical reaction properties using deep learning techniques underscore this potential [HG21, SPG22b, vGBB+24]. However, the field continues to face significant challenges that prohibit important applications.

The first challenge originates in the complexity of reaction mechanisms, often prompting researchers to focus on a single reaction class, leading to high accuracy on specific datasets [GPG20a, HG21, HGC+23]. However, a truly effective model should perform well across different reaction types.

Next, there is no consensus on how to best represent a chemical reaction in a machine-readable format. Different representations can capture various aspects of a reaction. The choice of representation significantly impacts model performance and generalizability, necessitating further research to determine optimal approaches for different types of predictions.

Additionally, there is the problem of the accurate mapping of atom-to-atom correspondences. In Figure 1.1, it is easy to trace each atom in the products back to specific atoms in the reactants. However, as reactions become more complex, finding this mapping becomes increasingly difficult. Many recent approaches explicitly rely on atom-to-atom mapping by using the mapping to represent the reactions in a machine-readable format or by incorporating the information in the architecture [JZW+25, SPG22b, vGBB+24]. This is a significant limitation as, in reality, such information is typically unavailable when making predictions.

Finally, from a theoretical machine learning perspective, the domain of learning on related graph structures (reactants and products) remains significantly understudied. While molecular property prediction research offers valuable insights, this field itself has several open problems [CSJ+24]. Reaction representations are inherently different from molecular representations in their complexity and structure. In the molecular property prediction field, we leverage the molecular structure as an inductive bias which directly relates to the learning objective. For reactions, we can still leverage the molecular structures of the reactants and products, but we need to rely on additional information to truly capture the transformation that is inherent to reactions. Finding this reaction-specific inductive bias has not yet been seriously studied.

To address these challenges, we take a systematic approach focused on both representations and architectures. We are essentially interested in finding the right *inductive bias* for the chemical reaction property prediction field. This prompts us to look for domain-specific knowledge to solve our problem. For representations, we examine the Condensed Graph of Reaction (CGR), which leverages atom-to-atom mappings, alongside the unmapped Dual Molecular Graph (DMG) and several graph transformations designed to improve information flow. For neural network architectures, we benchmark Message Passing Neural Networks and construct new architectures specifically designed to overcome the limitations of unmapped representations. Our approach is built on the hypothesis that incorporating reaction-specific inductive biases can compensate for the absence of explicit atom mapping information.

By constraining our scope to single-step ground state (reactions induced by heat) gas-phase reactions, we exclude considerations of solvents, catalysts, or other molecules beyond those directly participating in the reaction.

Our objectives are twofold:

1. Explore and evaluate various graph representations of single-step, ground-state, gas-phase reactions, identifying the most effective ways to encode reaction information for deep learning models.
    - How do different graph representations compare in performance for predicting reaction barrier heights?
    - What is the impact of using unmapped graph representations versus those that rely on explicit atom-to-atom mappings?
2. Investigate, compare, and build novel neural network architectures for predicting reaction barrier heights.
    - How do different neural network architectures and components compare in performance for predicting reaction barrier heights?

This work is structured as follows: In Chapter 2, we provide relevant background information about chemical reactions, machine learning for reaction property prediction, and graph neural networks to establish the foundational concepts needed for this work. Next, in Chapter 3, we review related work across different approaches to reaction representations. In Chapter 4, we present different chemical reaction graph representations, including the Condensed Graph of Reaction (CGR), Dual Molecular Graph (DMG), and different graph transformations. Chapter 5 covers both existing graph neural network architectures, as well as new methods proposed for overcoming atom-to-atom mapping. We provide an overview of our flexible, modular codebase for reaction property prediction in Chapter 6. In Chapter 7, we empirically evaluate our representations and architectures. Finally, in Chapter 8, we summarize our findings and propose directions for future work in this field.

CHAPTER 2
Background

Since this work lies at the intersection of the machine learning and chemistry fields, we argue that a solution to the reaction property prediction problem is found by leveraging knowledge from both domains. This chapter briefly explains the background information needed to understand this work. Section 2.1 covers the appropriate concepts in the chemical reaction field, while Section 2.2 specifically looks at the machine learning for chemical reaction field. Section 2.3 introduces the machine learning paradigm used in this work, graph neural networks.

2.1 Chemical Reactions

A chemical reaction is defined by the breaking and forming of bonds in molecules. The molecules before this transformation are called the reactants, while the molecules after the transformation are called the products. Typically, reactions can be visualized by a chemical equation in which the reactants and products are shown in their structural formulas and separated by an arrow indicating the direction of the reaction. Figure 2.1 shows several reactions each belonging to a different reaction type. In this work, we specifically work with single-step gas-phase reactions. We further discuss these different reaction types in Section 2.1.1. Next, we discuss the tracking of atoms from reactants to products, called atom-to-atom mapping, in Section 2.1.2. Finally, Section 2.1.3 covers the reaction barrier height which we try to predict in this work, as well as a short explanation of how these barrier heights are obtained.

2.1.1 Reaction Types

Reactions can be categorized according to their reaction mechanisms and reactants. E.g., the reaction in Figure 2.1b belongs to a class named the S_N2 class, where a nucleophile[1]

[1]A reactant that forms bonds by donating a pair of electrons to form a covalent bond

bonds to an sp^3-hybridized carbon atom[2] and on the opposite side from this atom a group (in this case a chloride) leaves. The reaction in Figure 2.1a follows an E2 reaction type where both a carbon-hydrogen bond and a carbon-halogen[3] bond break and a double bond forms. In this work, we also look at single-step unimolecular reactions such as the one shown in Figure 2.1c. Reaction classes provide a framework to simplify complex chemical concepts and help researchers navigate reaction space more effectively [SPV+21]. Many chemical reaction datasets are often released by reaction type [vRHBvL20, SJC23]. This is a limitation in the chemical reaction property prediction field as many models are trained on and evaluated on one-type reaction datasets. This causes models to have poor generalization ability.

Chemical reactions follow Lavoisier's Law of Conservation of Mass, which states that matter cannot be created or destroyed in a chemical reaction. This means that the total mass of the reactants must be equal to the total mass of the products. A balanced reaction contains the same number of atoms of each element on both sides of the reaction equation. For example, in reaction B of Figure 2.1, both sides contain the same number of carbon, hydrogen, nitrogen, and bromine atoms. Unbalanced reactions violate this law because they have different numbers of atoms on the reactant and product sides. All reactions in our datasets are balanced.

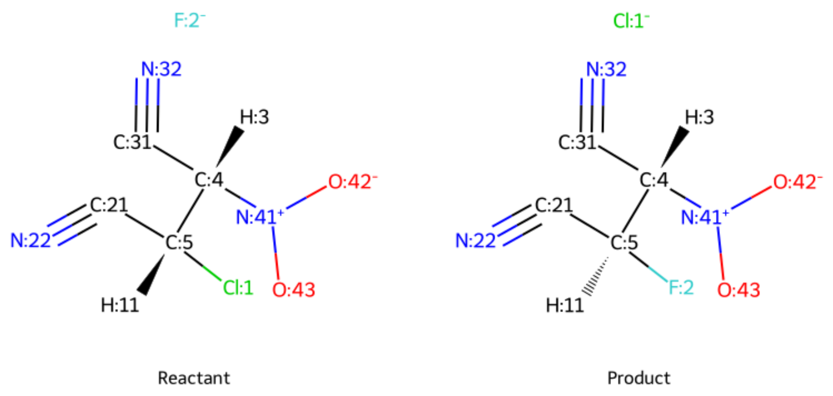
2.1.2 Atom-to-atom Mapping

In Figure 2.1, a chemist can easily trace each atom in the products back to specific atoms in the reactants (the numerical labels in Figure 2.1 denote the atom mappings). However, as reactions become more complex, finding this *atom-to-atom mapping* becomes increasingly more difficult. Many machine learning models for different chemical reaction problems rely on these mappings [HG21] [CJ22]. To obtain the mappings there exist rule-based systems [EPA25], several machine learning models [SHR+21] [CABJ24], or most accurately, the atom mappings can be manually obtained. Although successful attempts to predict reaction properties without relying on atom-mappings have been made [SVLR21, TSCB22, vGBB+24], the state-of-the-art models still rely on these mappings [HG21, SPG22b, vGBB+24].

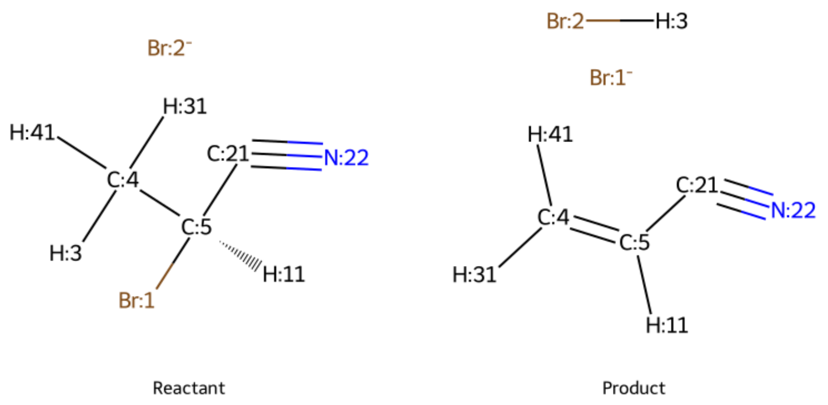
2.1.3 Reaction Barrier Height

A reaction can also be represented using an energy profile, which depicts an energetic pathway from the reactants to the products, as shown in Figure 2.2. The energy profile can be derived from the respective potential energy surface (PES), which denotes the relation between the energy of a system and its geometry. Using the energy profile we can thus get an idea of how the energy of our system changes as the structures of our reactants change as the reaction progresses. From the energy profile, we can see the reaction barrier height $\Delta G^{\ddagger}$ (sometimes referred to as the activation energy),

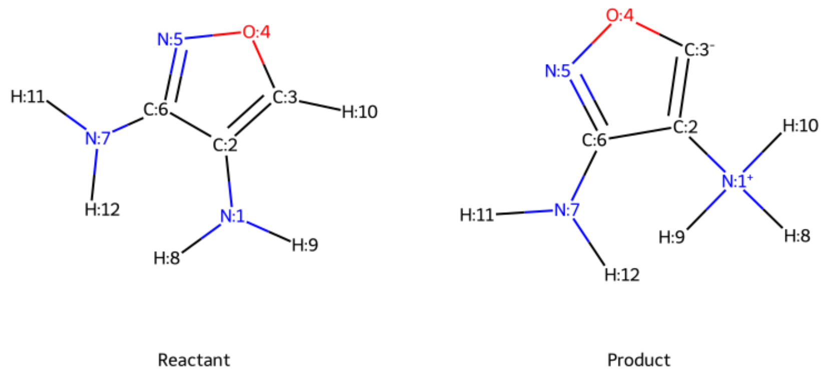
[2] A carbon with tetrahedral geometry due to sp^3 hybridization
[3] Bonds between carbon and the elements: F, Cl, Br, and I



(a) An S_N2 reaction from the S_N2 dataset [vRHBvL20].



(b) An E2 reaction from the E2 dataset [vRHBvL20].



(c) A unimolecular reaction from the RDB7 dataset [SPG22a].

Figure 2.1: Examples of different chemical reaction mechanisms, visualizations courtesy of M. Kovar.

which is the energy difference between reactants and the transition state (TS), under isothermic[4] and isobaric[5] conditions. The TS is the highest-energy configuration along our pathway where bonds are partially formed and broken. It represents a saddle point in the high-dimensional energy surface.

If enough energy is added to the reaction to overcome the reaction barrier height associated with the transition state, our reaction may proceed. This is an essential piece of information for chemists as it allows them to calculate reaction rates and understand reaction pathways [vGBA+24].

The reaction barrier height's quantity is energy, most commonly given in kilocalories per mole (kcal/mol), and can be either experimentally measured or calculated using quantum mechanics (QM) methods. Currently, QM methods provide the most accurate predictions of reaction barrier heights [BM07]. The majority of reaction barrier heights found in data sets are calculated using density functional theory (DFT) [Par89]. The accuracy of a QM method depends on its *level of theory.* For DFT specifically, accuracy and computational load depend on the functional and set of basis functions used, but even with simple functionals and small basis sets, these calculations are too computationally expensive to explore large numbers of reactions. In search of better accuracy and faster inference, researchers in the chemical reaction field are exploring machine learning techniques for reaction property prediction.

2.2 Machine Learning for Chemical Reactions

Due to the complexity of the underlying mechanisms of chemical reactions, there exist many different aspects of chemical reactions, each with its subfield and respective machine learning tasks. In this section, we want to give the reader the necessary background to understand some of the basic concepts in the field. We start by briefly explaining the features used in the field in Section 2.2.1. Next, we introduce reaction-specific data splits, more specifically, scaffold splits, in Section 2.2.2. Finally, Section 2.2.3 gives a brief overview of some of the machine learning tasks regarding chemical reactions.

2.2.1 Atom and Bond Features

Historically, machine learning methods for chemical reactions have primarily relied on features designed by domain experts [AEL+18]. For atoms, these features typically include elemental properties (such as atomic number and mass), local connectivity (degrees and hybridization), electronic characteristics (formal charge), and structural information (aromaticity). Bond features similarly encode chemical bond types, electronic properties like conjugation[6], and structural information.

[4]Temperature remains constant
[5]Pressure remains constant
[6]When three or more adjacent resonant atoms overlap with one another

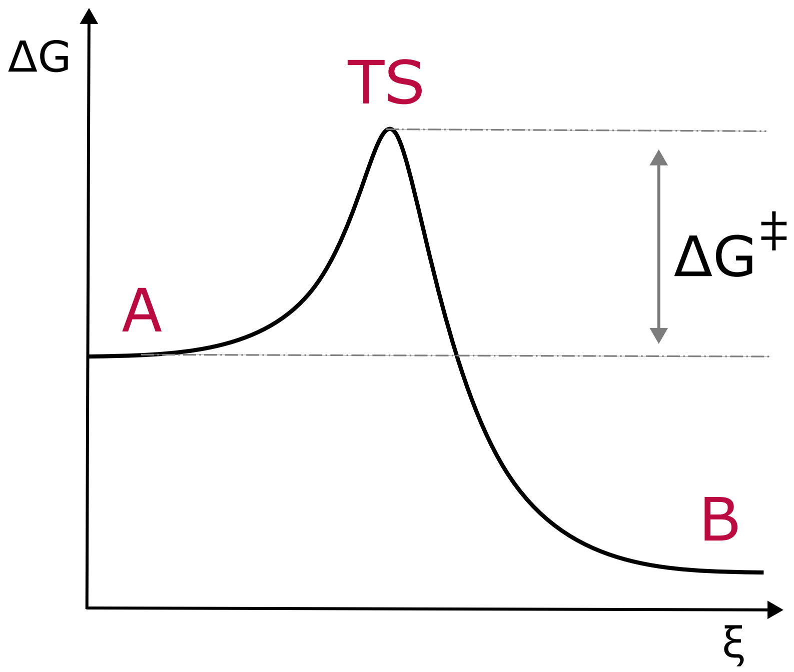


Figure 2.2: Reaction Barrier Height. The energy profile shows the change in Gibbs free energy (ΔG) along the reaction coordinate (ξ), with reactants (A) transforming to products (B) via a transition state (TS). The reaction barrier height ($\Delta G^{\ddagger}$) represents the energy difference between reactants and the transition state, indicating the minimum energy required for the reaction to proceed.

2.2.2 Scaffold Splits

When evaluating machine learning models, we usually randomly split the data in a train, validation, and test split. However, models trained on molecular data usually learn specific representations by memorizing the molecular *scaffolds* present in the training data, resulting in a poor ability to generalize to unseen structures [YSJ^{+}19]. A scaffold refers to the main structural parts of a molecule that define its structure. To combat this we use something called a scaffold split, which is more challenging than a random split as it introduces structural diversity between training and testing sets. By separating molecules based on their core scaffolds, we ensure that validation and test molecules contain structural patterns not encountered during training. This creates a more realistic evaluation scenario that measures a model's ability to extrapolate to new reaction types rather than just interpolate between similar training reactions.

2.2.3 Machine learning Tasks for Chemical Reactions

Machine learning has been increasingly applied to several chemical reaction tasks (listed in Table 2.1) [Sch21]. These tasks tackle different prediction problems, from reaction yield and rate prediction to more complex tasks like reaction outcome prediction and retrosynthesis. Reaction yield prediction aims to quantify product formation, while reaction rate prediction predicts the speed of chemical transformations. Reaction classification categorizes reactions into their respective reaction types, while atom mapping refers to finding the right atom-to-atom mapping between reactant and product atoms. Retrosynthesis works backward from target molecules to identify viable precursors and reaction routes.

Table 2.1: Common machine learning tasks for chemical reactions and their descriptions, adapted from [Sch21].

Prediction Task	Description
Reaction Yield	Percentage of reactants converted to desired products
Reaction Rate	Speed at which chemical reactions proceed
Reaction Outcome	Predicted products of a given set of reactants and conditions
Reaction Classification	Classifies reactions into reaction types and mechanisms
Reaction Barrier Height	The energy required for a reaction to proceed
Retrosynthesis	Identifies possible reactants and pathways that lead to a target molecule
Atom Mapping	The correspondence between atoms in reactants and products

2.3 Graph Neural Networks

In this work, we represent chemical reactions as graphs. Subsequently, we leverage graph neural networks to predict reaction barrier heights. In this section, we start by explaining what directed and undirected graphs are in Section 2.3.1. Next, Section 2.3.2 covers the most popular GNN framework, message passing neural networks. To understand the GNN architectures in Chapter 5, we cover important GNN properties influencing architectural design choices in Section 2.3.3. Finally, we explain how to encode positional and structural information of graphs in Section 2.3.4.

2.3.1 Graphs

A directed graph is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (also called nodes) and $\mathcal{E} \subseteq \{(u, v) | (u, v) \in \mathcal{V}^2 \wedge u \neq v\}$ is the set of edges connecting the vertices. The neighborhood of $v \in \mathcal{V}$ is $\mathcal{N}_v = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\}$. node and edge attributes can be incorporated into the graph such that $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$, where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, for $d > 0$, is the node attribute matrix and $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times e}$, for $e > 0$, is the edge attribute matrix. Graphs are often sparse, i.e., $|\mathcal{E}| \ll |\mathcal{V}|^2$. We denote the adjacency matrix of $\mathcal{G}$ as $A \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ such that $A_{uv} = 1 \iff (u,v) \in \mathcal{E}$.

An undirected graph is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (also called nodes) and $\mathcal{E} \subseteq \{\{u,v\}|u,v \in \mathcal{V} \wedge u \neq v\}$ is the set of edges connecting the vertices. The neighborhood of $v \in \mathcal{V}$ is $\mathcal{N}_v = \{u \in \mathcal{V}|\{v,u\} \in \mathcal{E}\}$. Node and edge attributes can be incorporated into the graph such that $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$, where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$, for $d > 0$, is the node attribute matrix and $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times e}$, for $e > 0$, is the edge attribute matrix. Graphs are often sparse, i.e., $|\mathcal{E}| \ll |\mathcal{V}|^2$. We denote the adjacency matrix of $\mathcal{G}$ as $A \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ such that $A_{uv} = 1 \iff \{u,v\} \in \mathcal{E}$. Note that for undirected graphs, the adjacency matrix is symmetric, i.e., $A_{uv} = A_{vu}$ for all $u, v \in \mathcal{V}$. In some machine learning frameworks such as PyTorch Geometric [PGM+19], undirected graphs are implemented as directed graphs where each node is connected by two bidirectional edges.

2.3.2 Message Passing Neural Networks

The most popular framework for building graph neural networks is message passing [GSR+17]. Here, a node u is encoded by a vector representation $\mathbf{h}_u^{(l)}$ at layer l, where $\mathbf{h}_u^{(0)} = \mathbf{x}_u$. This vector representation is then iteratively updated by a message passing layer that (1) computes "messages" between u and each of the nodes in the neighborhood $\mathcal{N}_u$ (Equation 2.1), (2) aggregates all messages coming from $\mathcal{N}_u$ (Equation 2.2), and (3) updates the representation of node u (Equation 2.3):

$$\begin{aligned}
\mathbf{m}_{uv}^{(l)} &= \texttt{MESSAGE}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}), & \forall (u,v) \in \mathcal{E} \quad (2.1)\\
\mathbf{a}_u^{(l)} &= \texttt{AGGREGATION}(\{\mathbf{m}_{uv}^{(l)}|v \in \mathcal{N}_u\}), & \forall u \in \mathcal{V}, \quad (2.2)\\
\mathbf{h}_u^{(l)} &= \texttt{UPDATE}(\mathbf{h}_u^{(l-1)}, \mathbf{a}_u^{(l)}), & \forall u \in \mathcal{V}. \quad (2.3)
\end{aligned}$$

The operations `MESSAGE`, `AGGREGATION`, and `UPDATE` can be implemented as neural networks. In practice, `AGGREGATION` is usually a fixed permutation-invariant aggregation operator (e.g., a sum or mean), while `MESSAGE` and `UPDATE` are usually simple feed-forward neural networks. Finally, we can take the updated node representations $\{\mathbf{h}_v^{(L)}|v \in \mathcal{V}\}$ as input to a `READOUT` operation to obtain a graph representation:

$$z = \texttt{READOUT}(\{\mathbf{h}_v^{(L)}|v \in \mathcal{V}\}), \quad (2.4)$$

where L is the total number of message passing layers and the `READOUT` operation is usually another permutation-invariant operator. This layer, known as a pooling layer, is often followed by a simple feed-forward neural network to obtain predictions.

2.3.3 Properties of Graph Neural Networks

Graph Neural Networks have unique properties that distinguish them from traditional neural networks and significantly influence their design, expressivity, and performance.

In this section, we give a brief overview of some of these properties as they have a major influence on how graph neural networks are constructed and evaluated.

Inductive Bias. The big difference between standard feedforward neural networks and GNNs is that we restrict our GNN to update nodes based on their neighboring nodes. Such a restriction, together with all other architectural design choices and assumptions on the data falls under a model's *inductive bias*. Since we are interested in different graph representations and neural networks for reaction barrier height prediction, we are essentially interested in finding the right inductive bias for our problem.

Expressivity. Expressivity refers to the range of functions a model can learn. Researchers in the GNN field often analyze the expressivity of their GNN architectures as this provides insight into how to optimize model architecture and featurization. Overly increasing expressivity can lead to overfitting and subsequently worse generalization [CSJ+24]. One way to look at expressivity is through the concept of graph isomorphism where we look at the ability of a model to distinguish two identical graphs. This way we can tell how expressive a GNN is based on which classes of graphs the model can distinguish.

The Weisfeiler-Leman (WL) isomorphism test is often mentioned when talking about GNN expressivity [XHLJ18] [MRF+19]. This algorithm works by iteratively labeling each node based on its label and the multiset of its neighbors' labels, then comparing these color distributions between graphs to determine potential isomorphism. The standard version of this test is called the 1-WL test, while the k-WL refers to the k-dimensional WL test, which works on k-tuples of nodes. Higher k-WL tests can distinguish more classes of graphs than lower tests. Since a GNN works as a WL test, GNN architectures' expressiveness can be compared to the k-WL test. Typical ways to increase expressivity are using information from long-range interactions or modeling high-order relationships such as triangles, and adding features such as positional encodings as discussed in Section 2.3.4.

Over-Smoothing and Over-Squashing As MPNNs process information across multiple layers, they can encounter a problem known as over-smoothing. This issue arises when node representations become increasingly more similar as the number of layers grows [RBM23]. The problem of over-squashing describes how a graph's structure can hinder the flow of information between distant nodes [AY20]. This occurs when the graph's layout creates bottlenecks where nodes' features are compressed into one node's representation.

2.3.4 Positional Encodings

Positioning encoding (PE) was first introduced in the Natural Language Processing (NLP) field to incorporate sequence ordering into transformers [VSP+17]. Since transformers employ global attention where every token attends to every other token, the inherent sequential nature of text would be lost without positional information. This calls for an encoding that gives the model positional context. For example, [VSP+17] implemented positional encoding using sinusoidal functions of different frequencies. The motivation behind this representation is that positions are easily learnable while allowing the model to extrapolate to sequence lengths unseen during training.

When adapting these concepts to graph neural networks, the challenge becomes more complex. Unlike the clear sequential ordering in text, graphs lack canonical positional information for nodes. Many different positional encodings have been proposed [DJL+23, DLL+21, YCL+21, RGD+22] and are usually added as a soft bias by adding them to the input features.

CHAPTER 3

Related Work

Historically, chemical reaction property prediction problems were modeled using quantitative structure-activity relationships (QSAR) which correlate reaction descriptors to regression or classification values. The recent surge in deep learning (DL) methods has amassed numerous DL models for problems surrounding chemical reactions [MC19, DAMR19]. Since a reaction is a complex physical phenomenon, accurately representing a reaction in a machine-readable format is an open problem [DQC+24]. Hence, researchers have used different reaction representations over time for their respective reaction problems. In this work, we follow the most popular categorization of these representations [DQC+24]: strings, fingerprints, descriptors, and graphs. In this chapter, we give for each representation a short overview of the contributions made to the chemical reaction field using the respective representation. We discuss string-based, fingerprint-based, descriptor-based, and graph-based methods in Section 3.1, 3.2, 3.3, 3.4, respectively.

3.1 String-based Methods

Given the recent advancements in Natural Language Processing (NLP) [VSP+17], string-based methods have benefited from the surge in language models to process chemical reactions. The Simplified Molecular-Input Line-Entry System (SMILES) [Wei88] is the most common string-based representation for molecules. This representation can further be extended to reactions by using the following Reaction SMILES format [LBG+99]:

$$\{\text{Reactant 1}\}.\{\text{Reactant 2}\}...\{\text{Reactant } n\} > \{\text{Product 1}\}.\{\text{Product 2}\}...\{\text{Product } n\},$$

where "." separates individual molecules within the same group and ">" separates the reactants and products. {} holds the respective SMILES for each molecule. The chemical reaction datasets used in this work contain Reaction SMILES to encode the reactions.

[SPV+21] leveraged a transformer-based model [VSP+17] that classifies reactions directly from reaction unmapped SMILES representations, achieving an accuracy of 98.2%. The authors also showed that the learned reaction embeddings can be used as reaction fingerprints (see the next section). In further work, [SVLR21] used a similar model that predicts chemical reaction yields. The recent work of [CZZL25] leverages the hierarchical and structural information of reactions as inductive bias. It uses transformer-based models to learn features at different hierarchies of reactions (reaction, molecule, and local atomic environment level) and uses those for different prediction tasks.

SMILES representations of chemical reactions are limited by their one-dimensional text format, which cannot capture 3D conformations and only approximates topological information, making it difficult to represent the spatial relationships between atoms [DQC+24].

3.2 Fingerprint-based Methods

Fingerprint-based representations encode reactions as bit or count fixed-length vectors and are derived using rule-based systems. For example, one could check if certain structural features are present within a molecule, encode each feature as a binary vector, and concatenate these into a fingerprint. These types of fingerprints are called MACCS keys [DQC+24]. The most popular fingerprinting technique is called Extended Connectivity Fingerprints (ECFP) [RH10]. Here, the fingerprints represent reactions by iteratively aggregating information from each atom's surrounding environment. [SLSL15] generates a reaction fingerprint by taking the difference of the atom-pair (AP) fingerprints of products and reactants. Atom-pair fingerprints encode molecules by capturing pairs of atoms and the shortest path between them, representing these relationships as hashed bit strings in a fixed-length vector [CSV85].

3.3 Descriptor-based Methods

Descriptor-based methods also encode reactions as fixed-length vectors but instead of using a rule-based system, they are obtained by computing chemical properties of molecules and reactions. Due to these computations, descriptors are generally more time-consuming [DQC+24]. For example, [AEL+18] computed different vibrational, atomic, and molecular properties using density functional theory and predicted reaction yields by training a random forest on the computed chemical descriptors.

3.4 Graph-based Methods

Due to the rise in popularity of graph neural networks [CSJ+24] and their success in the molecular property prediction field [GYG21, SBC+22, LGH+24], graphs are increasingly seen as the preferred method for modeling reactions [GPG20a, HG21, SPG22b, vGBB+24]. In this work, we focus on 2D graphs, meaning that our graphs do not contain information on the 3D coordinates of our reactants and products. However, molecules and reactions inherently exist in 3D space. Subsequently, many methods incorporate 3D information into their models [MDK$^+$22, vGBB$^+$24, JZW$^+$25]. We exclude 3D information since: (1) not all datasets contain the coordinates of reactants and products, (2) 3D information is hard to incorporate effectively into an architecture, and (3) to see how good a reaction property prediction model can get purely using 2D information.

[GPG20a] directly encoded the reactants and products to graphs and used a direct message passing neural network (D-MPNN) [YSJ$^+$19] to obtain embeddings for these molecules. Then, based on the atom-to-atom mapping, for each atom the difference of the reactant and product features was taken, to get a reaction embedding. [LSZ$^+$23] also encoded the reactants and products to graphs, but concatenated the molecular embeddings after the message passing into a reaction embedding, and thus their method does not rely on atom mapping. [TSCB22] created a connected hypergraph structure by adding virtual nodes at molecule and reaction levels. These virtual nodes enable message passing between reactants and product graphs molecular components without requiring atom-to-atom mapping.

[HG21, HGC$^+$23] were the first to use the so-called condensed graph of reaction (CGR) for reaction property prediction, which they combined with a directed message passing neural network to obtain the reaction embeddings. The CGR is formed by taking the overlap of the reactants and products graphs of the reaction, which requires atom mapping. Then, [SPG22b] directly built upon this work by incorporating additional atom and bond features. Additionally, they concatenated additional features before the dense layer readout to improve prediction performance.

[vGBB$^+$24] introduced a geometric deep learning model that encodes both 3D structures of reactants and products and (optional) atom-mapping information. The model processes molecules through separate symmetry-adapted channels (either invariant or equivariant to rotations) and then combines them to form a reaction representation. Similarly to the work of [HG21], [JZW$^+$25] do not encode reactants and products separately. Instead, they used atom-mapping to create artificial edges between corresponding atoms. They incorporated 3D information through edge lengths and bond angle information.

CHAPTER 4

Chemical Reaction Representations

Finding the right chemical reaction representation is not a straightforward task. The machine learning for molecular reaction property prediction field can intuitively encode a molecule to a graph and subsequently train a model on this representation. The representation provides the model an *inductive bias* that is directly influenced by the structure of a molecule. A reaction however is fundamentally different than a molecule, since we have a set of molecules that undergo a chemical transformation and we end up with a new set of molecules. In this chapter, we are interested in finding a representation that gives the right inductive bias for reaction property prediction models. Section 4.1 covers our point of reference and current SOTA representation, the Condensed Graph of Reaction (CGR). Next, we naturally translate reactants and products to graphs that combined form the Dual Molecular Graph (DMG), discussed in 4.2. Finally, we also explore different graph transformations that potentially could provide a better inductive bias in Section 4.3.

4.1 Condensed Graph of Reaction

The condensed graph of reaction (CGR) was first introduced by [Fuj86] and [HG21] were the first to use in a deep learning model for reaction property prediction. It is formed by taking the overlap of the reactant and product graphs, as shown in Figure 4.1c.

Let $\mathcal{G}^{\text{reac}} = (\mathcal{V}^{\text{reac}}, \mathcal{E}^{\text{reac}}, \mathbf{X}^{\text{reac}}, \mathbf{E}^{\text{reac}})$ and $\mathcal{G}^{\text{prod}} = (\mathcal{V}^{\text{prod}}, \mathcal{E}^{\text{prod}}, \mathbf{X}^{\text{prod}}, \mathbf{E}^{\text{prod}})$ represent the reactant and product molecular graphs respectively, where $\mathcal{V}^{\text{reac}}$ and $\mathcal{V}^{\text{prod}}$ share a bijective atom mapping $\phi : \mathcal{V}^{\text{reac}} \rightarrow \mathcal{V}^{\text{prod}}$ derived from the atom indices in the reaction SMILES notation. The CGR is defined as:

$$\mathcal{G}^{\text{CGR}} = (\mathcal{V}^{\text{CGR}}, \mathcal{E}^{\text{CGR}}, \mathbf{X}^{\text{CGR}}, \mathbf{E}^{\text{CGR}}).$$

For balanced reactions, the node set is defined as $\mathcal{V}^{\text{CGR}} = \mathcal{V}^{\text{reac}} = \mathcal{V}^{\text{prod}}$. The edge set $\mathcal{E}^{\text{CGR}}$ is the union of reactant edges and mapped product bonds, defined as $\mathcal{E}^{\text{CGR}} = \mathcal{E}^{\text{reac}} \cup \{(u, v) \mid (\phi(u), \phi(v)) \in \mathcal{E}^{\text{prod}}\}$. For node features, we combine the reactant features with the difference between product and reactant features:

$$\mathbf{x}_u^{\text{CGR}} = [\mathbf{x}_u^{\text{reac}} \| \, \mathbf{x}_{\phi(u)}^{\text{prod}} - \mathbf{x}_u^{\text{reac}}],$$

where $[\cdot\|\cdot]$ denotes a concatenation. [HG21] also experimented with different concatenation schemes but found that concatenating the reactant features with the difference between reactant and product features works the best. For edge features, we follow the same pattern:

$$\mathbf{e}_{uv}^{\text{CGR}} = \begin{cases} [\mathbf{e}_{uv}^{\text{reac}} \| \mathbf{e}_{\phi(u)\phi(v)}^{\text{prod}} - \mathbf{e}_{uv}^{\text{reac}}] & \text{if } (u, v) \in \mathcal{E}^{\text{reac}} \text{ and } (\phi(u), \phi(v)) \in \mathcal{E}^{\text{prod}}, \\ [\mathbf{e}_{uv}^{\text{reac}} \| \vec{0}] & \text{if } (u, v) \in \mathcal{E}^{\text{reac}} \text{ and } (\phi(u), \phi(v)) \notin \mathcal{E}^{\text{prod}}, \\ [\vec{0} \| \mathbf{e}_{\phi(u)\phi(v)}^{\text{prod}}] & \text{if } (u, v) \notin \mathcal{E}^{\text{reac}} \text{ and } (\phi(u), \phi(v)) \in \mathcal{E}^{\text{prod}}, \end{cases}$$

where zero vectors are used in cases where a bond does not exist in either reactants or products.

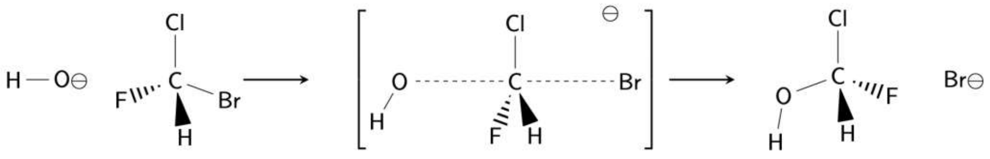
Generally, domain experts can impose strong inductive biases according to the information at hand in the data. We argue that the CGR is a very strong inductive bias for reaction models since it directly uses atom-to-atom mapping. This way reactant and product features are combined and the graph represents a sort of intermediate of the reaction, similar to the transition state. If we take away this piece of information, the atom-to-atom map, then we lose the ability to impose this inductive bias since we cannot form the CGR. Since finding a representation that does not rely on atom mapping is desirable, we must instead focus on other information at hand, which sets us up for the next section.

4.2 Dual Molecular Graph

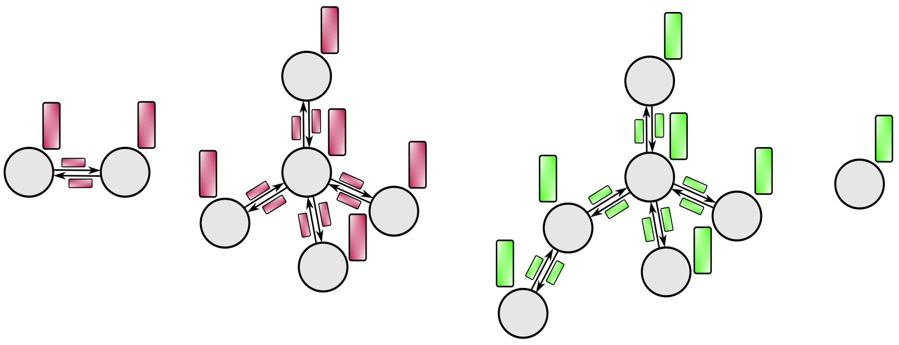
The Dual Molecular Graph (DMG) provides a straightforward representation of chemical reactions by combining the reactant and product graphs into a single disconnected graph (see Figure 4.1b). The reactants and products are different connected components within the disconnected graph. Unlike the CGR, the DMG does not require explicit atom-to-atom mapping information, making it more practical for real-world applications where such mappings are unavailable.

Formally, given reactant graph $\mathcal{G}^{\text{reac}} = (\mathcal{V}^{\text{reac}}, \mathcal{E}^{\text{reac}}, \mathbf{X}^{\text{reac}}, \mathbf{E}^{\text{reac}})$ and product graph $\mathcal{G}^{\text{prod}} = (\mathcal{V}^{\text{prod}}, \mathcal{E}^{\text{prod}}, \mathbf{X}^{\text{prod}}, \mathbf{E}^{\text{prod}})$, the DMG is defined as:

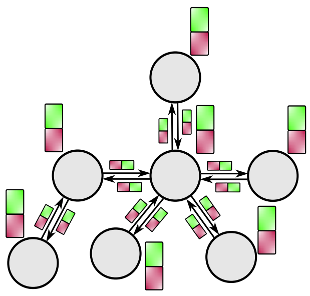
$$\mathcal{G}^{\text{DMG}} = (\mathcal{V}^{\text{DMG}}, \mathcal{E}^{\text{DMG}}, \mathbf{X}^{\text{DMG}}, \mathbf{E}^{\text{DMG}}),$$



(a) An S_N2 reaction (courtesy of E. Heid).



(b) The DMG graph representation of the reaction.



(c) The CGR graph representation of the reaction.

Figure 4.1: An S_N2 reaction and its DMG and CGR representations.

where the node set $\mathcal{V}^{\text{DMG}}$ is the disjoint union of nodes $\mathcal{V}^{\text{reac}} \uplus \mathcal{V}^{\text{prod}}$, and $\uplus$ indicates that reactant and product atoms are treated as distinct nodes in the disconnected graph, and the edge set $\mathcal{E}^{\text{DMG}}$ is the disjoint union of edges $\mathcal{E}^{\text{reac}} \uplus \mathcal{E}^{\text{prod}}$.

We argue that the CGR is so powerful since it uses important information at hand: which reactant node corresponds to which product node. The DMG does not have access to atom mappings, so we argue that (1) information on a node belonging to either reactants or products and (2) information on which molecule (e.g., reactant 1, reactant 2, product 1, or product 2) an atom belongs to, is crucial information for imposing inductive biases in the model. These properties can be explicitly encoded in the node and edge features or implicitly utilized through specialized neural network architectures as discussed in Chapter 5.

4.3 Transformed Representations

A simple way to potentially improve our representations is to *transform* our graph representations. The potential lies in finding a graph structure better suited for reaction barrier height prediction. Theoretical work also has shown that graph transformations can simulate higher-order GNNs [JTG23], meaning that we might improve the expressivity of our models by using transformed graphs. Many graph transformations are possible, here we choose three that are of interest to our problem. In Section 4.3.1, we add a virtual node to our DMG representation as it enables information exchange between reactants and products without the need for atom mapping. Next, to possibly leverage the available edge information we explore line graph transformations in Section 4.3.2. Finally, in Section 4.3.3, we directly connect nodes in the DMG based on their atom-to-atom mapping to better understand information exchange along atom mappings.

4.3.1 Virtual Node

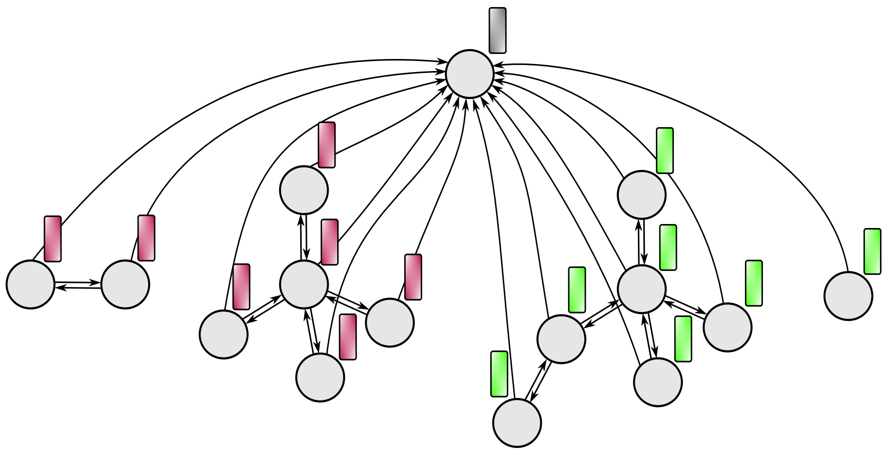
The Dual Molecular Graph representation has a fundamental limitation: it consists of disconnected components representing separate reactant and product molecules, preventing direct information exchange between them. This hinders the model's ability to learn the relationships between reactants and products, which might be important for predicting reaction barrier heights. Virtual nodes offer a potential solution to this problem by enabling information exchange between the disconnected components of the DMG without relying on atom-to-atom map information.

Virtual Node. A virtual node is an extra node v_{vn} added to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$ to capture global information. Formally, the augmented graph becomes $\mathcal{G}' = (\mathcal{V} \cup \{v_{\text{vn}}\}, \mathcal{E} \cup \{(v_{\text{vn}}, u), (u, v_{\text{vn}}) \mid u \in \mathcal{V}\}, \mathbf{X}', \mathbf{E}')$, where $\mathbf{X}' \in \mathbb{R}^{(|\mathcal{V}|+1)\times d}$ extends $\mathbf{X}$ with feature attributes for v_{vn}, and $\mathbf{E}'$ incorporates both original edges and new edges connecting v_{vn} to all $u \in \mathcal{V}$. The virtual node is fully connected to all original nodes, allowing for direct interaction with every node via its neighborhood $\mathcal{N}_{v_{\text{vn}}} = \mathcal{V}$. This allows for better long-range information exchange, which has been shown to be a limitation of MPNNs [CSJ+24].

We propose two virtual node configurations as shown in Figure 4.2: a global virtual node connected to all atoms, and a dual approach with separate virtual nodes for reactants and products that can also be connected (similarly as in the work of [TSCB22]). For both configurations, we test different connection schemes (bidirectional or unidirectional edges) and pooling strategies (pooling all nodes or only virtual nodes for prediction).

4.3.2 Line Graphs

Chemical reactions are fundamentally defined by the transfer of electrons [DQC+24]. This redistribution of electrons defines which bonds remain unchanged, are broken, or formed. In both the CGR and the DMG, this information lies in the edges, and not in the nodes. Most GNNs are specifically node-focused as we update node embeddings


(a) The DMG representation transformed by adding a global node with one directed edge from the DMG to the virtual node.

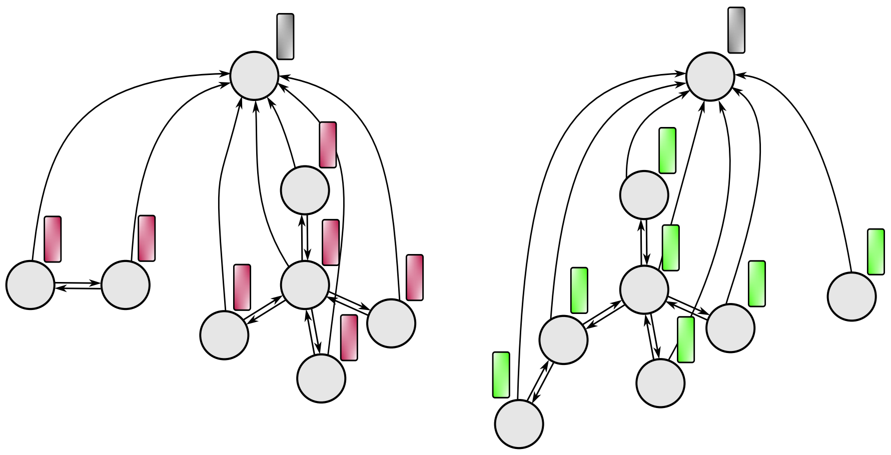

(b) The DMG representation transformed by adding two virtual nodes with directed edges from the DMG to the virtual nodes. Both reactants and products are connected to a respective virtual node.

Figure 4.2: Different virtual node transformations on the DMG representation.

based on the neighboring nodes' embeddings. Given this, we propose to transform our representations into their respective *line graph*.

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where edges are unordered pairs $\{u, v\}$, its line graph $L(\mathcal{G}) = (\mathcal{V}^L, \mathcal{E}^L)$ is constructed by representing each edge of $\mathcal{G}$ as a node in $L(\mathcal{G})$. Thus, $\mathcal{V}^L = \mathcal{E}$. Two nodes $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$ in $\mathcal{V}^L$ are adjacent in $L(\mathcal{G})$ if and only if the original edges e_1 and e_2 share a common node in $\mathcal{G}$. Formally, $\{e_1, e_2\} \in \mathcal{E}^L$ if $\{u_1, v_1\} \cap \{u_2, v_2\} \neq \emptyset$. This construction transforms shared nodes between edges in $\mathcal{G}$ into explicit adjacencies in $L(\mathcal{G})$. The adjacency matrix $A^L \in \{0, 1\}^{|\mathcal{E}| \times |\mathcal{E}|}$ encodes this relationship, where $A^L[e_1, e_2] = 1$ if and only if e_1 and e_2 share a node. Importantly, nodes in $L(\mathcal{G})$ correspond to edges in $\mathcal{G}$, while edges in $L(\mathcal{G})$ represent shared nodes between original edges.

The idea is that instead of adapting existing GNNs to incorporate edge feature processing we simply transform our graph representations. This opens the opportunity to use more models and potentially find a better-performing one. It has been shown that the current SOTA backbone model for reaction barrier height prediction, the Directed Message Passing Neural Network (D-MPNN) [DDS16] [YSJ+19] essentially does normal message passing on line graphs [GYG21].

For the Dual Molecular Graph (DMG), we implement two approaches to construct the line graph, directed and undirected. In the directed approach, each bond $(u, v) \in \mathcal{E}$ in the original DMG becomes a node in the line graph $L(\mathcal{G})$. The node features in $L(\mathcal{G})$ are created by concatenating the source atom features with the bond features:

$$\mathbf{x}^L_{(u,v)} = [\mathbf{x}_u \| \mathbf{e}_{uv}],$$

where $\mathbf{x}_u$ represents the features of atom u and $\mathbf{e}_{uv}$ represents the features of the bond between atoms u and v. Two nodes in the line graph are connected if the target of one bond is the source of another:

$$((u, v), (j, k)) \in \mathcal{E}^L \text{ iff } v = j.$$

For any atom $u \in \mathcal{V}$ in the original graph that does not participate in any bond (i.e., an isolated atom), we add a self-loop edge (u, u) to $\mathcal{E}$. The corresponding node in the line graph is then defined with features:

$$\mathbf{x}^L_{(u,u)} = [\mathbf{x}_u \| \mathbf{0}],$$

where $\mathbf{0}$ is a zero-padded vector corresponding to the missing bond features.

In the undirected approach (shown in Figure 4.3), each bond $\{u, v\} \in \mathcal{E}$ becomes a single node in $L(\mathcal{G})$ regardless of direction. To remove the sense of direction, we do not concatenate the atom features but instead use permutation-invariant operations, including summation:

$$\mathbf{x}^L_{(u,v)} = [\mathbf{x}_u + \mathbf{x}_v \| \mathbf{e}_{uv}],$$

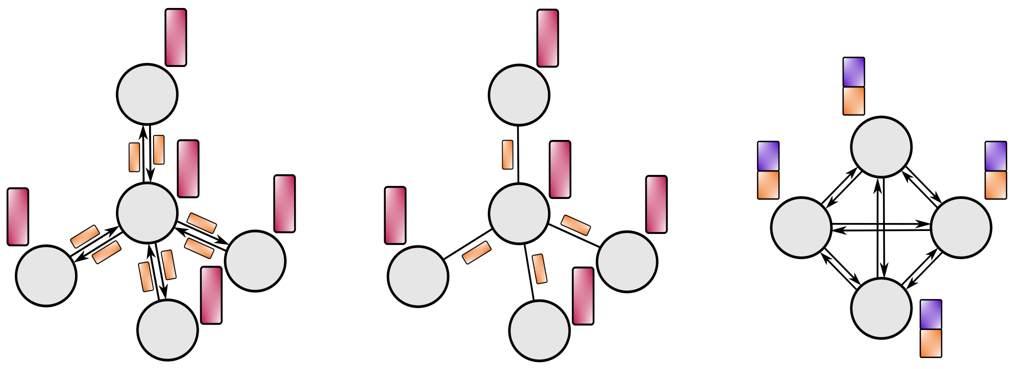


Figure 4.3: From left to right: a graph with bidirectional edges, the undirected version of the graph, and its line graph.

and averaging:

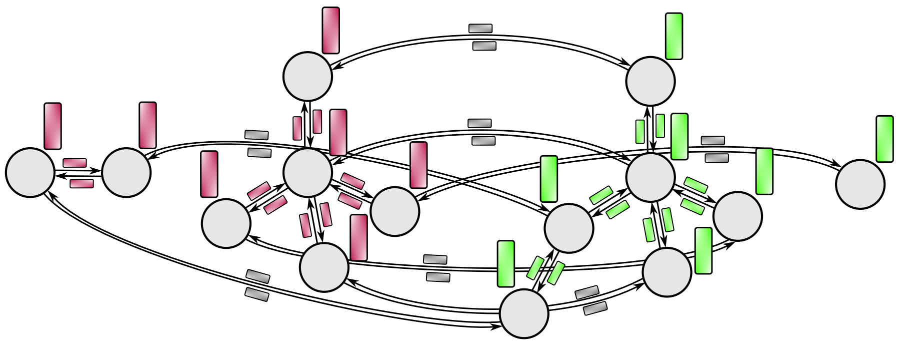
$$\mathbf{x}^L_{(u,v)} = [\frac{\mathbf{x}_u + \mathbf{x}_v}{2} \| \mathbf{e}_{uv}].$$

The edge features in the line graph encode the edge origin types (reactant or product) using one-hot encoding.

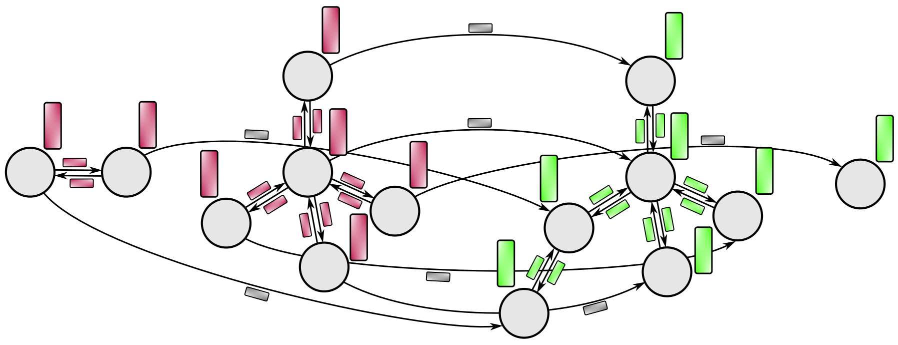
4.3.3 Connected Dual Molecular Graph

We hypothesize that the CGR representation is so effective [HG21] due to its node features being made out of a concatenation of reactant and product features. This forces the model to explicitly learn over each node's reactant and product features. Models learning on the DMG representation, on the other hand, take in for each node only the respective product or reactant features. Assuming that knowing the before and after (reactant and product, respectively) features for each node is meaningful for reaction barrier height prediction, a model trained on a DMG is in this case at a disadvantage. The CGR explicitly gives this information, while the DMG does not. Models training on a DMG can only possibly learn this information. Chapter 5 deals with finding models that could learn these relationships more effectively using just the DMG representation.

Since the CGR is an example of a representation that gives this information explicitly, we also propose a new transformed representation that gives similar explicit information, the connected DMG. As shown in Figure 4.4, the DMG is transformed by adding connections between corresponding atoms in reactants and products (so using the atom-to-atom map). This is similar to the work of [JZW+25]. We propose three different connection schemes: bidirectional connections, connections from reactant nodes to product nodes, and connections from product to reactant nodes. In the case of unidirectional connections, we pooled only the target group of the connection edges (e.g., for connections from products to reactants, we only pool reactants). This experiment also serves as a proof of concept that information exchange between product and reactant nodes is beneficial, which sets up the reasoning for further model development in the next chapter.



(a) The DMG representation transformed by adding bidirectional edges between corresponding atoms based on the atom-to-atom mapping.



(b) The DMG representation transformed by adding directional edges from reactant nodes to corresponding product nodes based on the atom-to-atom mapping.

Figure 4.4: Different connection transformations on the DMG representation.

CHAPTER 5

Graph Neural Network Architectures

In this chapter, we are focused on testing existing graph neural networks as well as designing new architectures for our chemical reaction property prediction problem. Many different graph neural networks could potentially be interesting for our problem. When it comes to GNNs, there is no one-fits-all solution. Due to this reason, we benchmark the most promising GNNs on our representations. Like representations, neural networks also provide inductive biases that could be interesting for chemical reaction property prediction. So, in addition to benchmarking, we also explore new architectures that could provide better inductive biases.

Our approach is two-fold. First, we perform benchmarking of existing graph neural networks across our different reaction representations. We explore message passing neural networks (MPNNs), as well as attention-based architectures, that have shown promise in molecular property prediction tasks. Second, we design novel architectures for reaction property prediction. Specifically, we are interested in models that perform well on the Dual Molecular Graph (DMG) representation without relying on explicit atom-to-atom mapping information. This direction is especially significant, as atom mapping information is often unavailable in real-world scenarios. If we can achieve comparable performance with unmapped representations, it would represent a significant improvement for the field.

5.1 Benchmarking Message Passing Neural Networks

Message Passing Neural Networks have historically dominated the graph-based machine learning field with only recently getting competition from hybrid models (MPNN and attention-based) and graph transformers [YCL^{+}21] [RGD^{+}22] [BJOL24]. Besides its track record, MPNNs are generally more efficient (standard GTs have quadratic complexity) and recent studies have shown that MPNNs can generally improve their performance by incorporating established techniques [TRRG23] [LSW24] [LSW25]. Given these, MPNNs are a promising choice for our reaction barrier height prediction problem. Additionally, current state-of-the-art models for reaction barrier height prediction are based on MPNNs [SPG22b] [vGBB+24].

We are generally not interested in finding the optimal set of hyperparameters for each MPNN. Rather, we are interested in comparing models within a budget constraint of parameters. This idea comes from the work of [DJL+23] that establishes a benchmarking framework containing benchmarking datasets and benchmarking guidelines. We constrain the amount of our models' parameters to 500k. This follows the most common budget constraint of the ZINC dataset [DJL+23] which is similar to our benchmarking dataset in terms of the number of graphs, average number of nodes, and task (more details in Chapter 7).

5.1.1 Message Passing Neural Networks

We benchmark the following MPNNs on our different representations: Graph Convolutional Networks (GCN) [KW16], GINE [HLG+19][XHLJ18], Gated Graph Convolutional Networks (GatedGCN) [BL17], Graph Attention Networks (GAT) [VCC+17], GATv2 [BAY21], Directed Message Passing Neural Networks (D-MPNNs) [YSJ+19] [HG21] and Principal Neighbourhood Aggregation (PNA) [CCB+20]. For the specific message passing schemes, the reader is referred to Appendix A. We conduct a grid search for GCN, GINE, GatedGCN, GAT, DMPNN, and PNA. For GATv2 we adopt the final hyperparameters of GAT. We choose GCN, GINE, GatedGCN, GAT, and PNA for their widespread use and known performance, GATv2 since it is a known improvement over GAT [BAY21] and DMPNN for being the SOTA backbone model for reaction barrier height prediction.

5.1.2 Encoders, Pooling Functions, and Prediction Head

In our reaction property prediction framework, we use several important components besides the message passing layers themselves: encoders to transform input features, pooling operations to aggregate node-level information into graph-level representations, and prediction heads to generate final outputs.

Encoders. Encoders' objectives are twofold: (1) they project different (node, edge, or PE) input features into a unified higher-dimensional embedding space where we can perform message passing, and (2) they can learn relevant feature relationships. We use simple linear encoders for nodes and edges:

$$\texttt{NodeEncoder}(\mathbf{X}) = \mathbf{X}\mathbf{W}_{\text{node}},$$

$$\texttt{EdgeEncoder}(\mathbf{E}) = \mathbf{E}\mathbf{W}_{\text{edge}},$$

where $\mathbf{W}_{\text{node}} \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{node}}}$, $\mathbf{W}_{\text{edge}} \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{edge}}}$ are trainable weight matrices (bias terms omitted for clarity).

Pooling Functions. These functions aggregate node-level representations into a single graph-level representation after the final message passing layer. We experiment with two types of pooling functions, both we refer to as `Pool`. The first one is sum pooling $\texttt{Pool}(\{\mathbf{X}_v^{(L)} | v \in \mathcal{V}\}) = \sum_{v \in \mathcal{V}} \mathbf{X}_v^{(L)}$. As a second pooling function, we propose Pooling by Multi-head Attention (PMA) [BJK+23], which learns to weight node contributions through attention mechanisms. PMA works by introducing a set of learnable seed vectors that attend to all node embeddings through a multi-head attention mechanism (see Section 5.4). The seed vectors are normalized, processed by multi-head attention with the node embeddings, and then passed through an MLP to produce the final graph representation.

Prediction Head. To obtain the final prediction, we process the graph-level representation through a prediction head. For our reaction barrier height regression task, we use a multi-layer perceptron (MLP) with ReLU activation:

$$\texttt{Head}(\mathbf{h}_{\mathcal{G}}) = \mathbf{W}_2(\text{ReLU}(\mathbf{W}_1 \mathbf{h}_{\mathcal{G}}),$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{hidden}}}$, $\mathbf{W}_2 \in \mathbb{R}^{1 \times d_{\text{hidden}}}$, and $\mathbf{b}_1, \mathbf{b}_2$ are trainable weight matrices (bias terms omitted for clarity). This MLP maps the graph representation to a single scalar value representing the predicted reaction barrier height.

5.1.3 Elevating MPNN Performance

[LSW25] found that while the literature often states that GTs outperform MPNNs, MPNN performance is generally understated due to the lack of several general techniques that are known to potentially improve performance. Specifically, [LSW25] enhance message passing layers with: edge feature integration, normalization, dropout, residual connections, feed-forward networks, and positional encoding. We exclude edge feature integration and positional encoding from our benchmarking experiments due to having separate experiments for these topics. We include non-linearity in the benchmarking experiments. Below, we briefly discuss each technique (see Figure 5.1):

Normalization. Normalization is essential for stabilizing the training of neural networks by reducing the effects of covariate shift, where the distribution of node embeddings changes across layers during training [LSW25]. There are two well-known techniques to combat this: (1) Batch Normalization (BN) [IS15] and (2) Layer Normalization (LN) [BKH16]. In this work, we always use BN, unless stated otherwise.

Activation. Activation functions introduce non-linearity into neural networks, allowing them to learn complex relationships. In this work, we use the ReLU activation function, $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.

Dropout. Dropout [SHK+14] is a regularization technique that randomly sets a fraction of input units to zero at each update during training, which helps prevent overfitting. For graph neural networks, dropout is particularly useful as it reduces co-adaptation of node features, making the model more robust. Following [LSW25], we apply dropout after the activation function.

Residual Connection. Residual connections [HZRS16] add the input of a layer to its output. This technique helps overcome the over-smoothing problem (see Section 2.3.3) and allows for deeper training (i.e., having more layers).

Feed-forward Neural Network. We add a feed-forward neural network (FFN) after each message passing layer (similarly as in the transformer architecture [VSP+17]). We implement a two-layer FFN with ReLU activation:

$$\mathrm{FFN}(\mathbf{h}) = \mathrm{Norm}(\mathrm{ReLU}(\mathbf{h}\mathbf{W}_{\mathrm{FFN1}})\mathbf{W}_{\mathrm{FFN2}} + \mathbf{h}),$$

where $\mathbf{W}_{\mathrm{FFN1}}$ and $\mathbf{W}_{\mathrm{FFN2}}$ are trainable weight matrices. The complete augmented message passing layer combines all these techniques in Algorithm 5.1.

We perform hyperparameter searches on one fold of RDB7. We chose this dataset due to its diverse set of reaction types and its reasonable size. For the MPNN hyperparameter search, we follow the work of [DJL+23] and constrain our parameter budget to 500,000. We chose this constraint due to RDB7 being similar in size to the subset of ZINC ($\sim$12,000) used in [DJL+23] and also being a chemical dataset containing molecular graphs. The hyperparameter searches for the MPNNs as well as the final parameters are found in Appendix B.

5.2 Positional Encodings

Positional encoding (PE) for graphs is usually mentioned when talking about Graph Transformers. Since in graph transformers, we let every node attend to every other node, we lose the notion of a graph. To make up for this, we add positional and structural encodings that encode the graph structure [DB20, LWWL20, CLLG+23]. These encodings can also be used for MPNNs to increase expressivity [RGD+22].

We propose to use Random Walk PE (RWPE) from the work of [DLL+21], which is an effective approach that encodes node positions (and structure) based on the random walk diffusion process. Inspired by [LWWL20], RWPE represents each node position using the probability of returning to itself after k steps in a random walk:

Algorithm 5.1: Augmented Message Passing Neural Network

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, node features
       $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{node}}$, edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_{edge}}$, and positional encodings
       $\mathbf{P}_{\text{RWPE}} \in \mathbb{R}^{|\mathcal{V}| \times K}$; Number of layers $L$
Output: Predicted reaction barrier height $y \in \mathbb{R}$
1  $\mathbf{X}^{(0)} \leftarrow \texttt{NodeEncoder}(\mathbf{X})$;
2  $\mathbf{P}^{(0)}_{\text{RWPE}} \leftarrow \texttt{PEEncoder}(\mathbf{P}_{\text{RWPE}})$;
3  $\mathbf{E}^{(0)} \leftarrow \texttt{EdgeEncoder}(\mathbf{E})$;
4  $\mathbf{X}^{(0)} \leftarrow [\mathbf{X}^{(0)} \| \mathbf{P}^{(0)}_{\text{RWPE}}]$;
5  for $l \leftarrow 0$ to $L - 1$ do
6      $\hat{\mathbf{X}}^{(l+1)}, \mathbf{E}^{(l+1)} \leftarrow \texttt{MPNN}^{(l)}(\mathbf{X}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A})$;
7      $\hat{\mathbf{X}}^{(l+1)} \leftarrow \texttt{Norm}(\hat{\mathbf{X}}^{(l+1)})$;
8      $\hat{\mathbf{X}}^{(l+1)} \leftarrow \texttt{Activation}(\hat{\mathbf{X}}^{(l+1)})$;
9      $\hat{\mathbf{X}}^{(l+1)} \leftarrow \texttt{Dropout}(\hat{\mathbf{X}}^{(l+1)})$;
10     $\hat{\mathbf{X}}^{(l+1)} \leftarrow \hat{\mathbf{X}}^{(l+1)} + \mathbf{X}^{(l)}$;
11     $\mathbf{X}^{(l+1)} \leftarrow \texttt{FFN}(\hat{\mathbf{X}}^{(l+1)})$;
12 end
13 $\mathbf{h}_{\mathcal{G}} \leftarrow \texttt{Pool}(\{\mathbf{X}_v^{(L)} | v \in \mathcal{V}\})$;
14 $y \leftarrow \texttt{Head}(\mathbf{h}_{\mathcal{G}})$;
15 return $y$
```

$$p_i^{\text{RWPE}} = [\text{RW}_{ii}, \text{RW}^2_{ii}, \dots, \text{RW}^k_{ii}] \in \mathbb{R}^k,$$

where RW $= AD^{-1}$ is the random walk operator, A the adjacency matrix, and D the degree matrix. [LWWL20] uses the full pairwise random walk matrix, while [DLL+21] uses only the landing probability of a node to itself (RW_{ii}). Due to the significant decrease in computational complexity, we also adopt this version of RWPE. We pass the positional encodings through a simple encoder before concatenating them to the node features.

5.3 A Simple Architectural Inductive Bias

We hypothesize that explicitly modeling the distinction between reactants and products gives an effective inductive bias for reaction barrier height prediction. We try this by adapting our MPNN models from the previous section by separately pooling reactant and product nodes, which lets the model learn separate representations for reactants and products. Similarly to the work of [LSZ+23], we subsequently concatenate the reactant and product features to obtain a reaction embedding which is passed through the prediction head. The full algorithm is shown in Algorithm 5.2.

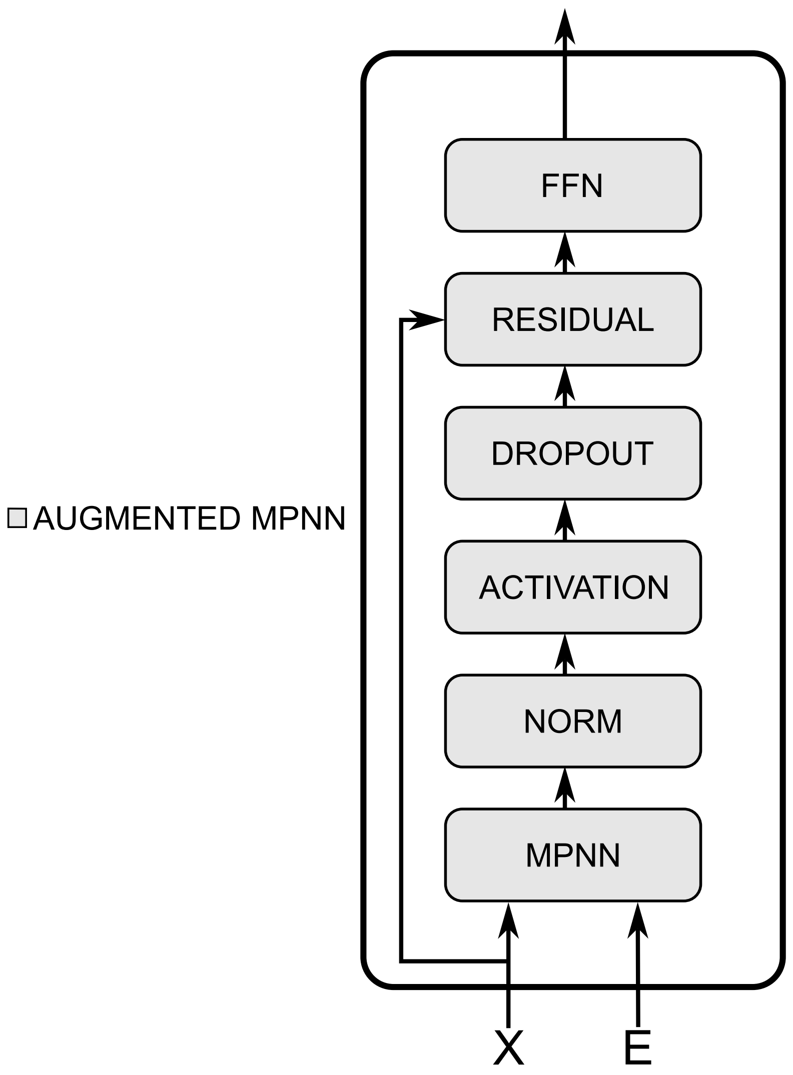


Figure 5.1: The augmented message passing layer that takes in node features **X** and edge features **E**. The features are first updated by the original message passing scheme, followed by a normalization, an update by an activation function, a dropout layer, a residual connection, and finally updated by a feed-forward neural network.

5.4 Learning on Disconnected Graphs Using Attention

Assuming information exchange between product and reactant nodes is beneficial for better reaction barrier height predictions, we are left with the problem that our DMG representation is a disconnected graph, and thus information exchange between some nodes is impossible. To overcome this, we propose using elements from the transformer architecture [VSP+17]. The transformer architecture has made significant progress in the natural language processing [VSP+17] and computer vision [DBK+20] field. Following the

Algorithm 5.2: A Simple Architectural Inductive Bias

```
   Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, node features
          $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{node}}$, edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_{edge}}$, and positional encodings
          $\mathbf{P}_{\text{RWPE}} \in \mathbb{R}^{|\mathcal{V}| \times K}$; Number of layers $L$
   Output: Predicted reaction barrier height $y \in \mathbb{R}$
 1 $\mathbf{X}^{(0)} \leftarrow$ NodeEncoder($\mathbf{X}$);
 2 $\mathbf{P}^{(0)}_{\text{RWPE}} \leftarrow$ PEEncoder($\mathbf{P}_{\text{RWPE}}$);
 3 $\mathbf{E}^{(0)} \leftarrow$ EdgeEncoder($\mathbf{E}$);
 4 $\mathbf{X}^{(0)} \leftarrow [\mathbf{X}^{(0)} \| \mathbf{P}^{(0)}_{\text{RWPE}}]$;
 5 for $l \leftarrow 0$ to $L - 1$ do
 6     $\mathbf{X}^{(l+1)} \leftarrow$ AugmentedMPNN($\mathbf{X}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}$);
 7 end
 8 $\mathbf{h}_{\text{reactants}} \leftarrow$ Pool($\{\mathbf{x}_v^{(L)} | v \in \mathcal{V}_{\text{reactants}}\}$);
 9 $\mathbf{h}_{\text{products}} \leftarrow$ Pool($\{\mathbf{x}_u^{(L)} | u \in \mathcal{V}_{\text{products}}\}$);
10 $\mathbf{h}_{\mathcal{G}} \leftarrow [\mathbf{h}_{\text{reactants}} \| \mathbf{h}_{\text{products}}]$;
11 $y \leftarrow$ Head($\mathbf{h}_{\mathcal{G}}$);
12 return $y$
```

success, researchers in the graph neural networks field have tried adapting the transformer architecture for graph data [DB20, YCL+21, RGD+22, MLL+23, BJOL24], resulting in so-called graph transformers (GT). These GTs could alleviate some of the problems that come with MPNNs, discussed in Section 2.3.3 [MLL+23].

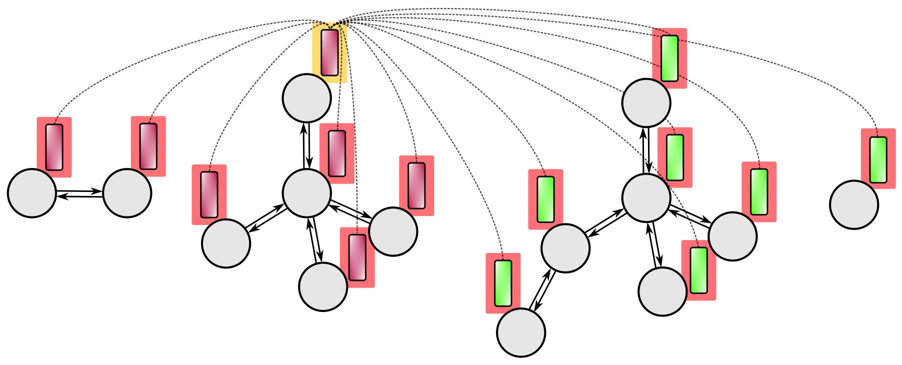
Graph Transformer. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \{(u, v) | (u, v) \in \mathcal{V}^2 \wedge u \neq v\}$ is the set of edges connecting the vertices, and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, for $d > 0$, is the node attribute matrix. At its core, a transformer is built by stacking l layers of transformer blocks which consist of a multi-head attention mechanism and a fully-connected feed-forward network. A single-head attention mechanism is given by:

$$\texttt{Attention}(\mathbf{X}^{(l)}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right)\mathbf{V}, \tag{5.1}$$

where $\mathbf{X}^{(l)}$ is the node feature matrix at layer l, d_K the feature dimension of the matrices $\mathbf{Q}$ and $\mathbf{K}$. $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are obtained by projecting $\mathbf{X}^{(l)}$ by three matrices $\mathbf{W}_Q \in \mathbb{R}^{d \times d_K}$, $\mathbf{W}_K \in \mathbb{R}^{d \times d_K}$, and $\mathbf{W}_V \in \mathbb{R}^{d \times d_V}$:

$$\mathbf{Q} = \mathbf{X}^{(l)}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}^{(l)}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}^{(l)}\mathbf{W}_V, \tag{5.2}$$

with bias terms omitted for simplicity. Finally, multi-head attention (`MultiHeadAttention`) is obtained by concatenating multiple single attention heads and subsequently projecting


(a) Using global attention, each node's features in the DMG are updated using all other nodes' features, allowing for information exchange between disconnected components.

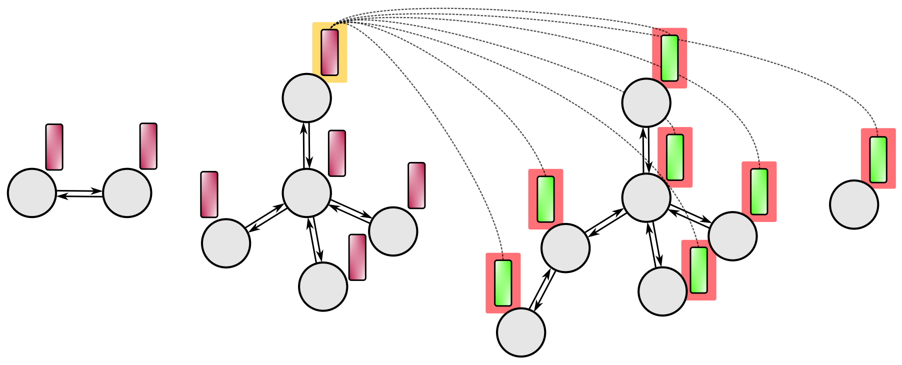

(b) Using reaction attention, each node's features in the DMG are updated using only respective product or reactant nodes' features.

Figure 5.2: Different attention mechanisms visualized on the DMG representation.

to the feature space of $\mathbf{X}^{(l)}$.

Attention-based models could learn atom-at-atom maps internally. Since in attention, we let every node attend to every node (illustrated in Figure 5.2a), there is a possibility that the model learns to exchange information along the atom-to-atom map or possibly more meaningful patterns.

While standard global attention allows information exchange between all nodes in the disconnected DMG, we hypothesize that a more structured attention mechanism, explicitly designed to model the interaction between the reactant and product sets, provides a stronger and more relevant inductive bias. By forcing reactant nodes to attend specifically to product nodes (and vice-versa), we guide the model to focus on learning the correlations and changes across reactants and products. This targeted approach could potentially be more effective at capturing the essence of the transformation than undirected global attention or locally-operating MPNNs, even without explicit atom mapping.

Reaction Attention. Here, we propose an adapted form of the global attention mechanism by restricting the input to the query, key, and value matrices (see Figure 5.3). Let $\mathbf{X}^{(l)}_{\text{reac}}$ and $\mathbf{X}^{(l)}_{\text{prod}}$ denote the reactant and product node features, respectively, at layer l, where $\mathbf{X}^{(l)}_{\text{reac}} \in \mathbb{R}^{|\mathcal{V}_{\text{reac}}|\times d}$ and $\mathbf{X}^{(l)}_{\text{prod}} \in \mathbb{R}^{|\mathcal{V}_{\text{prod}}|\times d}$. The Reaction Attention mechanism consists of two separate attention operations:

$$\texttt{ReacAttention}(\mathbf{X}^{(l)}_{\text{reac}}, \mathbf{X}^{(l)}_{\text{prod}}) = \text{softmax}\left(\frac{\mathbf{Q}_{\text{reac}}\mathbf{K}^T_{\text{prod}}}{\sqrt{d_K}}\right)\mathbf{V}_{\text{prod}}, \tag{5.3}$$

$$\texttt{ProdAttention}(\mathbf{X}^{(l)}_{\text{prod}}, \mathbf{X}^{(l)}_{\text{reac}}) = \text{softmax}\left(\frac{\mathbf{Q}_{\text{prod}}\mathbf{K}^T_{\text{reac}}}{\sqrt{d_K}}\right)\mathbf{V}_{\text{reac}}, \tag{5.4}$$

$$\mathbf{X}^{(l+1)} = \texttt{ReactionAttention}(\mathbf{X}^{(l)}) = \begin{bmatrix}\texttt{ReacAttention}(\mathbf{X}^{(l)}_{\text{reac}}, \mathbf{X}^{(l)}_{\text{prod}}) \\ \texttt{ProdAttention}(\mathbf{X}^{(l)}_{\text{prod}}, \mathbf{X}^{(l)}_{\text{reac}})\end{bmatrix} \tag{5.5}$$

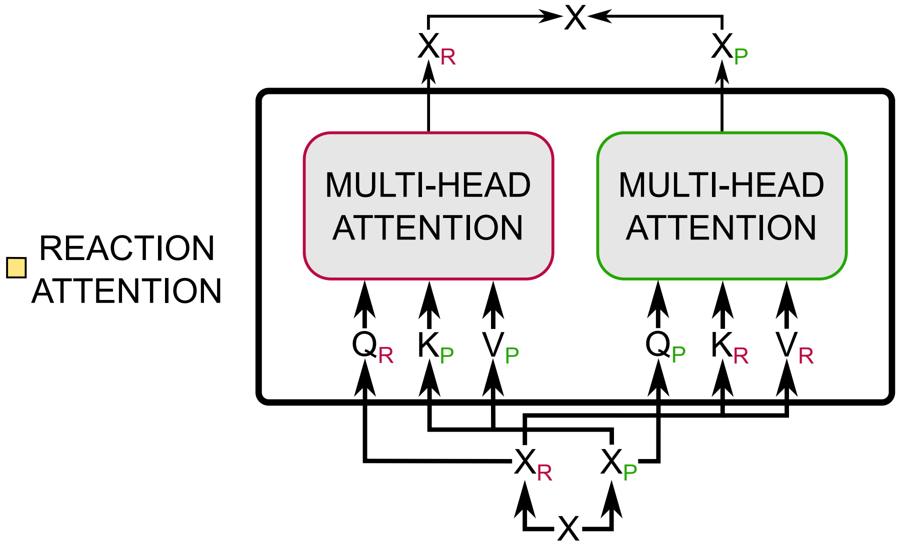


Figure 5.3: Reaction Attention. Node features are split into reactant and product node features and subsequently processed by two separate multi-head attention mechanisms.

5.5 Message Passing and Attention

Since MPNNs are proven to work well on the CGR [HG21], we are interested in using them together with attention layers on the DMG representation to leverage both local, global, and targeted information exchange. We propose a simple approach to this by adding one or more attention layers after the message passing layers, similarly as in the work of [WJW+21]. Algorithm 5.3 shows the full model (normalization, residual connections, and FFNs are omitted for clarity). We do not use positional encoding due to the MPNN layers already capturing local information.

Algorithm 5.3: Stacking Attention Layers on Top of MPNN-Layers

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, node features
       $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d_{node}}$, edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times d_{edge}}$; Number of AugmentedMPNN
       layers $L_M$; Number of Attention layers $L_A$
Output: Predicted reaction barrier height $y \in \mathbb{R}$
1  $\mathbf{X}^{(0)} \leftarrow$ NodeEncoder($\mathbf{X}$);
2  $\mathbf{E}^{(0)} \leftarrow$ EdgeEncoder($\mathbf{E}$);
3  for $l \leftarrow 0$ to $L_M - 1$ do
4      $\mathbf{X}^{(l+1)} \leftarrow$ AugmentedMPNN($\mathbf{X}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A}$);
5  end
6  for $l \leftarrow 0$ to $L_A - 1$ do
7      $\mathbf{X}^{(L_M+l+1)} \leftarrow$ Attention($\mathbf{X}^{(L_M+l)}$);
8  end
9  $\mathbf{h}_\mathcal{G} \leftarrow$ Pool($\{\mathbf{x}_v^{(L_M+L_A)} | v \in \mathcal{V}\}$);
10 $y \leftarrow$ Head($\mathbf{h}_\mathcal{G}$);
11 return $y$
```

This method however might be subject to the MPNN-specific problems of over-squashing, over-smoothing, and low expressivity [RGD+22]. Hence, we also propose to use the GPS architecture [RGD+22] that leverages a hybrid MPNN and transformer layer, rather than the two-stage approach. Algorithm 5.4 shows the used GPS model, in which we use RWPE positional encoding. We omitted normalization, residual connections, and FFNs for clarity.

For both approaches, we can leverage normal attention as well as the proposed reaction attention. For both architectures, we experiment with both standard global attention and our proposed Reaction Attention mechanism. Additionally, we explore directional variants of the Reaction Attention mechanism, such as a product-focused approach where we only update product features by allowing product nodes to attend to reactant nodes, followed by pooling only the product nodes for the final prediction.

5.6 Reaction Graph Transformer

In search of a stronger inductive bias, we propose a graph transformer architecture specifically designed for chemical reaction graphs, the Reaction Graph Transformer (RGT). Unlike hybrid approaches that integrate message passing with attention mechanisms, RGT is a pure graph transformer architecture that relies entirely on attention mechanisms for learning representations of chemical reactions. The RGT architecture consists of multiple transformer layers, each containing a multi-head self-attention mechanism with

Algorithm 5.4: GraphGPS-style network with RWPE

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, node features
       $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{node}}$, edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_{edge}}$, and positional encodings
       $\mathbf{P}_{\text{RWPE}} \in \mathbb{R}^{|\mathcal{V}| \times K}$; Number of layers $L$; Local message passing model
       instance MPNN; Attention model instance Attention
Output: Predicted reaction barrier height $y \in \mathbb{R}$
1  $\mathbf{X}^{(0)} \leftarrow \texttt{NodeEncoder}(\mathbf{X})$;
2  $\mathbf{P}^{(0)}_{\text{RWPE}} \leftarrow \texttt{PEEncoder}(\mathbf{P}_{\text{RWPE}})$;
3  $\mathbf{E}^{(0)} \leftarrow \texttt{EdgeEncoder}(\mathbf{E})$;
4  $\mathbf{X}^{(0)} \leftarrow [\mathbf{X}^{(0)} \| \mathbf{P}^{(0)}_{\text{RWPE}}]$;
5  for $l \leftarrow 0$ to $L - 1$ do
6      $\mathbf{X}^{(l+1)}_M, \mathbf{E}^{(l+1)} \leftarrow \texttt{MPNN}(\mathbf{X}^{(l)}, \mathbf{E}^{(l)}, \mathbf{A})$;
7      $\mathbf{X}^{(l+1)}_T \leftarrow \texttt{Attention}(\mathbf{X}^{(l)})$;
8      $\mathbf{X}^{(l+1)} \leftarrow \texttt{MLP}(\mathbf{X}^{(l+1)}_M + \mathbf{X}^{(l+1)}_T)$;
9  end
10 $\mathbf{h}_\mathcal{G} \leftarrow \texttt{Pool}(\{\mathbf{X}^{(L)}_v | v \in \mathcal{V}\})$;
11 $y \leftarrow \texttt{Head}(\mathbf{h}_\mathcal{G})$;
12 return $y$
```

a specialized masking strategy, followed by a feed-forward neural network (see Algorithm 5.5).

The key idea of our approach is using chemical knowledge to create inductive biases through attention masking. While previous attention methods typically only differentiate between reactants and products, our architecture leverages richer chemical information, such as molecular groupings and connectivity patterns, to create more informative information flow pathways (as shown in Figure 5.4).

The main building block of the RGT architecture is the masked multi-head attention mechanism. Formally, the masked attention operation is defined as:

$$\texttt{MaskedAttention}(\mathbf{X}^{(l)}, \mathbf{A}_{\text{mask}}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}} + \mathbf{A}_{\text{mask}}\right)\mathbf{V}, \tag{5.6}$$

where attention is computed only for pairs of atoms allowed by the mask $\mathbf{A}_{\text{mask}}$. The matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are obtained by projecting the node features $\mathbf{X}^{(l)}$ using learned weight matrices.

As shown in Table 5.1, we support different masking strategies that encode different chemical relationships:

- Local (L) Mask: Restricts attention to only directly bonded neighbors. This

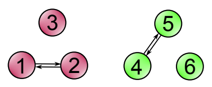
Algorithm 5.5: Masked Graph Transformer with Layer-Specific Attention Masks

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, node features
       $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{node}}$, edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_{edge}}$, and positional encodings
       $\mathbf{P}_{\text{RWPE}} \in \mathbb{R}^{|\mathcal{V}| \times K}$; Atom origin types $\mathbf{O} \in \{0,1\}^{|\mathcal{V}|}$; Molecule indices
       $\mathbf{C} \in \mathbb{N}^{|\mathcal{V}|}$; Sequence of attention masks $\mathbf{M} = [M_1, M_2, \dots, M_L]$ where
       $M_i \in \{\text{local}, \text{inter}, \text{intra}, \text{molecule}, \dots\}$
Output: Predicted reaction barrier height $y \in \mathbb{R}$
1  $\mathbf{X}^{(0)} \leftarrow$ NodeEncoder($\mathbf{X}$);
2  $\mathbf{P}^{(0)}_{\text{RWPE}} \leftarrow$ PEEncoder($\mathbf{P}_{\text{RWPE}}$);
3  $\mathbf{X}^{(0)} \leftarrow [\mathbf{X}^{(0)} \| \mathbf{P}^{(0)}_{\text{RWPE}}]$;
4  for $l \leftarrow 0$ to $L-1$ do
5      $\mathbf{A}^{(l)}_{\text{mask}} \leftarrow$ GenerateMask($\mathbf{X}^{(l)}, \mathbf{mask}, \mathbf{O}, \mathbf{C}, M_l, \mathbf{A}$);
6      $\hat{\mathbf{X}}^{(l+1)} \leftarrow$ MultiHeadAttention($\mathbf{X}^{(l)}, \mathbf{X}^{(l)}, \mathbf{X}^{(l)}, \mathbf{A}^{(l)}_{\text{mask}}$);
7      $\tilde{\mathbf{X}}^{(l+1)} \leftarrow$ Norm($\hat{\mathbf{X}}^{(l+1)} + \mathbf{X}^{(l)}$);
8      $\mathbf{X}^{(l+1)} \leftarrow$ FFN($\tilde{\mathbf{X}}^{(l+1)}$) $+ \tilde{\mathbf{X}}^{(l+1)}$;
9  end
10 $\mathbf{h}_{\text{reactants}} \leftarrow$ Pool($\{\mathbf{x}^{(L)}_v | v \in \mathcal{V}_{\text{reactants}}\}$);
11 $\mathbf{h}_{\text{products}} \leftarrow$ Pool($\{\mathbf{x}^{(L)}_u | u \in \mathcal{V}_{\text{products}}\}$);
12 $\mathbf{h}_{\mathcal{G}} \leftarrow [\mathbf{h}_{\text{reactants}} \| \mathbf{h}_{\text{products}}]$;
13 $y \leftarrow$ Head($\mathbf{h}_{\mathcal{G}}$);
14 return $y$
```

is similar to MPNNs, forcing the model to learn representations based on the neighboring atoms within the disconnected reactant and product components.

- Inter (I) Mask: Allows attention only between reactant atoms and product atoms. This mask is similar to the proposed Reaction Attention in Section 5.4.
- Intra (In) Mask: Allows attention only within the reactant set or within the product set, but not between them. This allows the model to update representations based on the properties of all reactants (or products) together, without being restricted by the disconnected components of reactants (or products).
- Same Molecule (Sm) / Other Molecules (Om) Masks: These provide more restrictions within the Intra category, allowing attention only within the atoms of a single molecule (Sm) or between atoms of different molecules of the same type (e.g., Reactant 1 to Reactant 2) (Om). This helps in learning molecule-specific features or inter-reactant/inter-product interactions separately.

By stacking layers with different attention masks, the RGT architecture resembles a



$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & -\infty & -\infty & -\infty & 0 & 0 & 0 \\ 2 & -\infty & -\infty & -\infty & 0 & 0 & 0 \\ 3 & -\infty & -\infty & -\infty & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & -\infty & -\infty & -\infty \\ 5 & 0 & 0 & 0 & -\infty & -\infty & -\infty \\ 6 & 0 & 0 & 0 & -\infty & -\infty & -\infty \end{array}$$

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & -\infty & -\infty & -\infty & -\infty \\ 2 & 0 & 0 & -\infty & -\infty & -\infty & -\infty \\ 3 & -\infty & -\infty & 0 & -\infty & -\infty & -\infty \\ 4 & -\infty & -\infty & -\infty & 0 & -\infty & -\infty \\ 5 & -\infty & -\infty & -\infty & -\infty & 0 & 0 \\ 6 & -\infty & -\infty & -\infty & -\infty & 0 & 0 \end{array}$$

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & -\infty & 0 & -\infty & -\infty & -\infty \\ 2 & -\infty & 0 & 0 & -\infty & -\infty & -\infty \\ 3 & 0 & 0 & 0 & -\infty & -\infty & -\infty \\ 4 & -\infty & -\infty & -\infty & 0 & 0 & 0 \\ 5 & -\infty & -\infty & -\infty & 0 & 0 & -\infty \\ 6 & -\infty & -\infty & -\infty & 0 & -\infty & 0 \end{array}$$

Figure 5.4: A DMG and different masking schemes. From left to right and top to bottom: (1) a DMG representation of simple reaction, (2) an inter (I) mask that only allows attention between reactant and product nodes, (3) a local (L) mask where only neighboring nodes are attended to, and (4) an intra (In) mask where attention is only allowed inside respective reactants or products.

chemically-restricted GT. Different masking strategies could mimic previous architectures: pure local masks mimic MPNNs and local masks followed by inter masks or alternating local and inter masks mimic the models from Section 5.4.

Table 5.1: Attention mask types for the Reaction Graph Transformer.

Mask Type	**Abbrev.**	**Description**
Self	S	Global attention where all nodes attend to all other nodes.
Local	L	Attention between neighboring nodes.
Inter	I	Attention only between reactant and product atoms (cross-reaction attention).
Intra	In	Attention only within the same category (reactant-to-reactant or product-to-product).
Same Molecule	Sm	Atoms can only attend to other atoms within the same molecule.
Other Molecules Same Category	Om	Atoms can attend to atoms in different molecules but of the same category (reactant or product).

CHAPTER 6

Codebase

A well-prepared code base can have a significant impact on future work. For example, the GraphGPS [RGD+22] code base provides a modular approach to building graph transformers where one can use different positional encodings, message passing mechanisms, and global attention mechanisms. Subsequently, many works used the GraphGPS code base for their research [MLL+23, SVV+23, LSW25]. In this chapter, we present our code base that addresses a critical need in the chemical reaction prediction field: the lack of a flexible, modular framework that allows researchers to experiment with different graph representations of reactions and neural network architectures without extensive code rewriting. This enables faster iteration of model designs and more systematic evaluation of different approaches to reaction barrier height prediction. Section 6.1 covers the design philosophy behind our codebase and Section 6.2 gives an overview of the codebase and its modules.

6.1 Design Philosophy

Due to the interdisciplinary nature of this work, we require a code base that can accommodate both graph-based machine learning architectures and the specific requirements of chemical reaction data. Our design philosophy is based on creating a modular, extensible, and user-friendly framework that facilitates experimentation and reproducibility with different reaction graph representations and neural network architectures.

Similar to the approach taken by [RGD+22] with GraphGPS, we adopt a modular design that allows researchers to easily swap components and experiment with different configurations. However, while GraphGPS was built on GraphGym [YYL20] for configuration management and module initialization, we chose to use Hydra [Yad19] for this purpose. Hydra offers several advantages: a more flexible configuration system with hierarchical configuration files, easy command-line overrides, and multi-run functionality for parallel experimentation [YYL20].

The key principles behind our design include:

Modularity. Each component of the code base, such as data processing, graph representations, neural network architectures, and training settings, is implemented as an independent, replaceable module. This allows researchers to experiment with new components while reusing established ones, significantly accelerating the research cycle. For example, new graph representations can be tested with existing neural network architectures, or vice versa.

Reproducibility. By providing a general framework, we facilitate and unify the training of models for reaction property prediction. This means researchers can easily load the same data, process it in the same way, and use the same splits for training. By leveraging a proper configuration schema, researchers can keep track of their experiments and easily share training, model, and data settings.

Unlike specialized frameworks like Chemprop [YSJ+19, HGC+23], which primarily focuses on D-MPNN architectures for molecular property prediction with limited representation options, or GraphGPS [RGD+22], which provides general graph transformer capabilities but lacks chemistry-specific components, our framework combines the best of both worlds. It offers the chemical domain knowledge of Chemprop (support for reaction-specific features, atom mappings, and reaction representations) with the architectural flexibility of GraphGPS (modular design for swapping components like encoders, layers, and pooling mechanisms). By using Hydra [Yad19] for configuration management and component initialization rather than GraphGym [YYL20], we also gain more flexible parameter organization and simplified parallel experimentation.

6.2 Overview

Our framework is organized into three main module categories, each handling a different aspect of the reaction property prediction pipeline, all supported by the Hydra configuration system. The implementation is built on PyTorch [PGM+19] for deep learning operations and PyTorch Geometric [FL19] for graph-specific functionality.

6.2.1 Data Modules

The data module provides functionalities for loading, processing, and transforming chemical reaction datasets. It follows a simple flow: it takes a configuration dictionary as input and produces PyTorch Geometric `DataLoader` objects for training, validation, and testing as output. The configuration specifies all modules of the data pipeline: dataset, featurizer, representation, and transformation. Each of these can be easily swapped with other compatible modules. For example, we can easily use the same dataset with different graph representations or apply different transformations to the same base representation.

Dataset Module. The dataset module serves as the entry point of our data pipeline. From its configuration, the module extracts information about data sources, such as file paths and column specifications that identify where reaction SMILES and barrier heights are stored. It also determines how data should be split between training, validation, and testing sets.

Featurizer Module. Based on the featurizer configuration, this module selects predefined feature sets for atoms and bonds. The module creates specialized featurizer functions that extract these properties from RDKit molecular objects [L+06] and convert them into fixed-length feature vectors. These featurizers are then passed to the representation module, which applies them during graph construction.

Representation Module. Each representation module is a specific reaction representation class that converts reaction SMILES into its respective representations. The module handles the parsing of SMILES strings, identification of atom mappings, application of the featurizer functions to generate node and edge features, representation-specific processing, and converts the internal graph structure to a PyTorch Geometric's `Data` object.

Transformation Module. The transformation module modifies PyTorch Geometric `Data` objects after the initial graph construction. Examples of transformations include: preprocessing graph statistics (such as degree information required by PNA), computing positional encodings (such as random walk positional encodings), and modifying the graph structure (e.g., adding virtual nodes). Our code supports the sequential application of multiple transforms, each defined in a modular configuration file.

6.2.2 Model Modules

Similar to the data modules, the model construction follows a module-based design that takes a configuration dictionary as input and constructs the model from modular components. The architecture is organized into five components: encoders, layers, pooling operations, and prediction heads, all of which are put together by a model module that defines the overall architecture and forward pass. Components can be easily swapped or combined without changing the underlying codebase. For example, a message passing neural network layer can be swapped for a transformer-based attention layer, or we can add multiple positional encoders to an architecture. Each module except the head takes in a PyTorch Geometric `Batch` object and returns an updated version of this object.

Encoder Module. An encoder module transforms input features into learned embeddings. Different encoders can be configured to handle different aspects of the input, such as node features, edge features, or positional information. Multiple encoders can be applied sequentially, with their outputs usually concatenated.

Layer Module. This module defines the backbone layer used for a model. Generally, there are three types of layers: message passing neural network (MPNN) layers, attention-based layers, and hybrid layers that combine both approaches. Typical message passing and attention layers are predefined in submodules that can be used for a layer configuration. For example, a layer configuration can be defined to use the GCN architecture as an MPNN submodule and to additionally add normalization and feedforward neural networks.

Pool Module. Our framework supports different pooling strategies, including simple operations (sum, mean, max) and more complex learnable aggregations.

Head Module. This module transforms graph-level embeddings into final predictions. For our task, we usually use the same configuration which consists of a multi-layer perceptron (MLP) that maps the pooled graph representation to the reaction barrier height. A model could easily be used for graph classification instead of regression by switching out the head module.

Model Module. The model module integrates all other modules into a cohesive neural network architecture. It defines how data flows through the network during the forward pass and manages the interactions between different components. In our framework, a model typically first passes the input through a series of encoders, then through multiple layers of graph neural networks, followed by a pooling operation to obtain a graph-level representation, and finally through a prediction head to generate the output.

6.2.3 Training Modules

The training modules control how model parameters are updated during the training process. Like other parts of our code base, these modules are configured through the Hydra configuration system, making it easy to experiment with different optimization strategies without modifying the code.

Optimizer. The optimizer module defines the algorithm used to update network weights based on the computed gradients. Our framework supports various optimization algorithms from PyTorch, including Adam [KB14] and AdamW [LH17]. The optimizer configuration can be easily specified in the main configuration file, allowing users to select appropriate optimizers for specific datasets or models.

Scheduler. The scheduler module adjusts the learning rate during training to improve convergence. We support multiple scheduling strategies through PyTorch's learning rate schedulers, such as `ReduceLROnPlateau`, and custom schedulers like cosine decay with warmups. Some schedulers, such as `ReduceLROnPlateau`, require monitoring validation metrics, which our framework automatically handles.

CHAPTER 7

Experimental Evaluation

In this chapter, we give empirical evidence for the effectiveness of different graph representations and neural network architectures for predicting reaction barrier heights. Our main goal is to determine which methods give the best performance and how to achieve accurate predictions without relying on atom-to-atom mapping information.

In Section 7.1, we first lay out our experimental setup, describing our datasets used for evaluation, their properties, and their relevance to reaction barrier height prediction. We describe our hardware configuration, evaluation metrics, training procedures, and data splits to ensure reproducibility. Additionally, we outline the node and edge features used across all experiments. Section 7.2 presents our benchmarking results, starting with a comparison of MPNNs on both the CGR and DMG representations. We then evaluate the proposed graph transformations. Finally, we analyze the effectiveness of different architectural changes, from separate pooling strategies to our proposed Reaction Graph Transformer with specialized attention mechanisms.

Our key empirical findings can be summarized as follows:

- When augmented with state-of-the-art machine learning techniques, most MPNNs show comparable performance on the CGR representation, with PNA achieving the lowest MAE of 4.32 ± 0.45.
- Separate pooling of reactants and products in the DMG representation significantly improves performance, reducing MAE from ~22 to ~7.8 for all tested models.
- Our proposed Reaction Attention mechanism significantly outperforms global attention when working with the DMG representation (from 21.86 ± 0.36 to 6.31 ± 0.33), validating that targeted information exchange between reactants and products is important.

- The Reaction Graph Transformer architecture with specialized attention masking strategies slightly outperforms all other unmapped approaches, with an MAE of 6.18 ± 0.30.
- Mapped representations (CGR) still outperform unmapped ones (DMG) on diverse reaction datasets, the performance gap is considerably smaller on single reaction-type datasets.

7.1 Experimental Setup

In this section, we outline the experimental setup to evaluate the representations and neural network architectures, including details on hardware configuration, datasets, data splits, node and edge features, training procedures, and evaluation metrics to ensure the reproducibility of the reported results.

7.1.1 Hardware

All experiments were conducted on two GPU types, an NVIDIA A40 and an NVIDIA A6000. The A40 has 2 AMD CPUs (24 cores per CPU, 2 threads per core) and 48GB of GPU memory. The NVIDIA A6000 has 2 AMD CPUs (32 cores per CPU, 2 threads per core) and 48GB of GPU memory. All results are reproducible on either GPU.

7.1.2 Datasets

For experimentation, we use the four datasets presented in Table 7.1. These datasets contain different amounts of graphs, ranging from 1,264 to 11,926. Only the RDB7 dataset consists of mixed reaction types, while the other datasets focus on specific reaction types. For RDB7, we also have the reverse reactions which we add to the data. If a dataset does not have a common name in the literature, one is given.

RDB7. The RDB7 dataset, created by [SPG22a], contains 11,926 single-step chemical reactions involving H, C, N, and O atoms with up to seven heavy atoms. It evolved from earlier versions developed by [GPG20b] (RDB7-B97 and RDB7-ωB97), but with higher accuracy coupled-cluster quantum chemistry calculations (CCSD(T)-F12a/cc-pVDZ-F12//ωB97X-D3/def2-TZVP). This version provides more accurate energy values, with barrier heights differing by approximately 5 kcal mol^{-1} RMSE compared to the previous ωB97X-D3/def2-TZVP calculations. The dataset includes atom-mapped SMILES, activation energies, and reaction enthalpies, with both forward and reverse reactions available.

E2/SN2. The E2 and SN2 datasets were created by [vRHBvL20] and originally came as one dataset containing 4,466 transition state geometries for competing elimination (E2) and nucleophilic substitution (SN2) reactions. The geometries were optimized at

Table 7.1: The used chemical reaction datasets and their properties. Asterisk (*) denotes dataset names assigned by us rather than established in the literature.

Dataset		Properties		
Common Name	Level of Theory	Graphs	Reaction Type	Reverse Reactions Available
RDB7	CCSD(T)-F12a/cc-pVDZ-F12 //ωB97X-D3/def2-TZVP	11,926	Mixed	✓
E2	DF-LCCSD/cc-pVTZ //MP2/6-311G(d)	1,264	E2	-
SN2	DF-LCCSD/cc-pVTZ //MP2/6-311G(d)	2,361	S_N2	-
Cycloadd*	B3LYP-D3(BJ)/def2-TZVP //B3LYP-D3(BJ)/def2-SVP	5,269	Cycloaddition	-

MP2/6-311G(d), and the final reported energies were calculated at the DF-LCCSD/cc-pVTZ//MP2/6-311G(d) level of theory. The atom-mapped reaction SMILES were obtained from [HG21].

Cycloadd. The Cycloadd dataset was created by [SJC23] and contains 5,269 [3+2] dipolar cycloaddition reactions. Calculations were performed at the B3LYP-D3(BJ)/def2-TZVP//B3LYP-D3(BJ)/def2-SVP level of theory. Each reaction includes atom-mapped SMILES, activation energies, reaction energies, and optimized 3D geometries for reactants, transition states, and products.

7.1.3 Data Splits and Random Seeds

For the RDB7 dataset, we use the 5-fold cross-validation training, validation, and testing splits provided by [SPG22a]. These are created by using a Bemis-Murcko scaffold split [BM96] on the reactants from the forward reactions. Each forward and respective reverse reaction is put in the same set to avoid data leakage. The sets are based on an 85% training, 5% validation, and 10% testing data split. We used the E2 and SN2 dataset splits as described in [HGC+23], which follows the split sizes set by [SC22] and [HvRvL21], consisting of 1440 training and 360 validation data points for SN2, as well as 800 training and 200 validation data points for E2. Due to too few scaffolds, these datasets were split randomly. The cycloadditions dataset splits were also copied from [HGC+23], consisting of randomly split 80% training, 10% validation, and 10% test data.

7.1.4 Node and Edge Features

We adopt the same node and edge features as described in [SPG22b] (see Table 7.2), which builds upon the features used in [HGC+23]. Specifically, [SPG22b] replaced a feature that indicated whether or not a bond is located in a ring of any size with the actual ring size as a one-hot encoding for both atoms and bonds. Furthermore, atoms are encoded through their atomic number, total degree, formal charge (net electric charge), implicit hydrogen count, hybridization state (atomic orbital configuration affecting 3D geometry), aromaticity flag (stability property in ring structures), mass, and inclusion in different rings (differing in size). Bonds are encoded by their type, conjugation status[1], and ring inclusion flags. The features are obtained using RDKit [L+06].

Table 7.2: Atom and bond features used in the model, including their descriptions.

Feature	Description
Atom Features	
Atomic Number	Element types (H, B, C, N, O, F, Si, P, S, Cl, Br, I)
Total Degree	Number of bonded atoms (0–5)
Formal Charge	Formal charge (-2 to +2)
Implicit Hydrogens	Implicit hydrogen count (0–4)
Hybridization State	Hybridization type (S, SP, SP^2, SP^3)
Aromaticity	Aromaticity status (True/False)
Atomic Mass	Atomic mass (scaled $\times 0.01$)
In 3-Membered Ring	Presence in 3-membered ring (True/False)
In 4-Membered Ring	Presence in 4-membered ring (True/False)
In 5-Membered Ring	Presence in 5-membered ring (True/False)
In 6-Membered Ring	Presence in 6-membered ring (True/False)
In 7-Membered Ring	Presence in 7-membered ring (True/False)
Bond Features	
Bond Type	Bond type (single, double, triple, aromatic)
Conjugated	Conjugation status (True/False)
In 3-Membered Ring	Presence in 3-membered ring (True/False)
In 4-Membered Ring	Presence in 4-membered ring (True/False)
In 5-Membered Ring	Presence in 5-membered ring (True/False)
In 6-Membered Ring	Presence in 6-membered ring (True/False)
In 7-Membered Ring	Presence in 7-membered ring (True/False)

7.1.5 Training Settings and Evaluation Metrics

For all experiments, we maintained a specific set of training hyperparameters to ensure reproducibility and fairness, unless stated otherwise. Each model was trained with a batch size of 50 for a maximum of 200 epochs. Early stopping was used with a patience of 30 epochs, meaning training would stop if no improvement in validation loss was observed over 30 epochs. We employed a learning rate warm-up strategy over the first 10 epochs. Weight decay was set to 0.01 to mitigate overfitting, and gradient clipping with a norm value of 1 was applied to prevent exploding gradients during backpropagation.

[1]Whether a double bond is connected to other double bonds.

For evaluation, we used Mean Absolute Error (MAE) as our performance metric, which measures the average absolute difference between predicted and actual reaction barrier heights in kcal/mol. This metric is also used in related work. All reported results represent the average MAE across all cross-validation folds with the corresponding standard deviation unless stated otherwise.

7.2 Results

7.2.1 Benchmarking Message Passing Neural Networks

Table 7.3 presents the comparison of different Message Passing Neural Networks on the Condensed Graph of Reaction representation, each augmented with techniques discussed in Section 5.1.3 and under a constraint of 500k parameters (following the work of [DJL+23]).

Table 7.3: Hyperparameters and performance of Message Passing Neural Networks on CGR representation (MAE in kcal/mol).

Properties	GCN	GINE	GAT	GATv2	GatedGCN	D-MPNN	PNA
Depth	9	12	12	12	12	3	9
Hidden size	64	128	64	64	64	128	64
BN	False	False	False	False	True	False	True
Activation	ReLU	Identity	ReLU	ReLU	ReLU	ReLU	ReLU
Dropout	0.05	0.05	0.05	0.05	0.05	0.1	0.05
Residual	True	True	True	True	True	True	True
FFN	True	False	True	True	True	True	True
LR	0.001	0.0005	0.001	0.001	0.001	0.001	0.001
PE	False	False	False	False	False	False	False
Edge Info	True	True	True	True	True	True	True
# Parameters	198,017	427,265	311,681	360,833	466,049	305,921	498,113
Time/epoch	12s	10s	26s	28s	28s	5s	23s
Test MAE	4.72 ± 0.45	4.44 ± 0.43	4.72 ± 0.53	4.64 ± 0.44	4.39 ± 0.33	4.61 ± 0.44	**4.32 ± 0.45**

All MPNNs show similar performance ranging between 4.3 to 4.8 MAE. We find that our models benefit from most of the augmentations made in the work of [LSW25]. The only major difference is that positional encodings do not improve our results. This may be due to: (1) structural information already being present in the features (e.g., ring information) and (2) the CGR's inherent reaction representation which might overshadow any information coming from PEs. As in the work of [LSW25] we find that using residual connections allows for deeper networks. Only D-MPNN has 3 layers while all others have 9 or 12. The work of [HG21] also found that shallow and wide D-MPNNs perform better. This could be due to the unique interaction of the CGR and the D-MPNN message passing scheme (which works directly on the edge features).

Table 7.4: Performance of Message Passing Neural Networks on DMG representation (MAE in kcal/mol).

	GINE	GatedGCN	D-MPNN	PNA
Test MAE	22.35 ± 0.38	22.18 ± 0.33	**22.15 ± 0.49**	22.34 ± 0.38

In Table 7.4 we use the best-performing models from Table 7.3 on the DMG representation. All MPNNs perform similarly, with test MAE values clustered around 22. This significantly worse performance on the DMG could be explained by (1) the lack of information exchange between atoms not belonging to the same molecule, (2) the missing atom mapping information, and (3) a lack of information about which nodes belong to either reactants or products.

7.2.2 Separate Pooling of Reactants and Products

Table 7.5 shows the results we obtain by separately pooling the reactants and products as described in Section 5.3. We tried simple additive pooling, as well as pooling by multihead attention (PMA).

Table 7.5: Impact of pooling strategy and positional encoding on DMG performance (MAE in kcal/mol).

Pooling	PE	GINE	GatedGCN	D-MPNN	PNA
Add	No	**7.73 ± 0.34**	**7.81 ± 0.32**	7.98 ± 0.39	**7.80 ± 0.45**
PMA	No	8.21 ± 0.46	8.84 ± 0.31	9.12 ± 0.41	8.84 ± 0.31
Add	Yes	7.74 ± 0.39	8.11 ± 0.47	**7.89 ± 0.37**	7.86 ± 0.35
PMA	Yes	8.46 ± 0.36	8.97 ± 0.34	9.10 ± 0.18	8.97 ± 0.34

Using simple additive pooling provides the best results with MAE values around 7.8 across all tested models. Generally, we notice a significant reduction in MAE (from ~22 to ~7.8). This confirms our hypothesis that explicitly modeling the distinction between reactants and products provides an effective inductive bias for reaction barrier height prediction.

When using separate pooling, the model creates distinct embeddings for reactants and products, providing the prediction head with two representations that capture the features of each reaction component. This differs from global pooling where all nodes are pooled together into a single embedding. The separate pooling approach addresses an inherent limitation of the DMG: since reactants and products exist as disconnected components in the graph, there is no natural way for information to be exchanged between them during message passing. By maintaining separate embeddings, the model can first learn meaningful representations of each component independently, and then use the prediction head to learn relationships between these components that correlate with reaction barrier heights. When all nodes are pooled together without prior information exchange, the resulting embedding contains a mixture of disconnected features that fail to capture meaningful relationships between reactants and products. The slightly worse performance of PMA compared to additive pooling suggests that the simpler aggregation mechanism is sufficient for our task.

7.2.3 Virtual Node

Table 7.6 presents the results of incorporating virtual nodes into our Dual Molecular Graph representation across different message passing neural networks. As described in Section 4.3.1, virtual nodes were added to enable information exchange between the disconnected reactant and product components without requiring atom-to-atom mapping.

Table 7.6: Performance of virtual node augmentation across message passing architectures (MAE in kcal/mol).

	GINE	GatedGCN	D-MPNN	PNA
Test MAE	22.26 ± 0.33	21.80 ± 0.46	$\mathbf{11.86 \pm 0.44}$	22.33 ± 0.41

The D-MPNN architecture shows a significant improvement when combined with a virtual node, achieving a test MAE of ~11.8, outperforming other architectures that remain in the 21-22 MAE range. This might be due to how D-MPNN's message passing scheme works, as each edge, is considered a node during the message passing, the virtual node connections become many nodes during the message pass. The D-MPNN architecture then transfers the learned information of these nodes onto the virtual node (and to all other nodes if the virtual node connection is bidirectional). While the virtual node approach improves upon the baseline DMG representation (22.15 MAE for D-MPNN), it still falls short of the separate pooling strategy (7.98 MAE). This suggests that while virtual nodes do facilitate some information exchange across the reaction, they cannot fully compensate for the structural information provided by atom mappings or the explicit modeling of reactant-product distinctions.

7.2.4 Connected Dual Molecular Graph

As proposed in Section 4.3.3, we evaluate the effect of connecting reactant and product nodes in the DMG according to their atom mappings. We experimented with three different connection schemes: bidirectional connections, connections from reactant nodes to product nodes, and connections from product to reactant nodes. In the case of unidirectional connections, we pooled only the target group of the connection edges (so for connections from products to reactants, we only pool reactants). The results are shown in Table 7.7.

Table 7.7: Effect of connection direction in connected DMG on GatedGCN performance (MAE in kcal/mol).

Connection Direction	GatedGCN
Bidirectional	20.73 ± 0.38
Products to Reactants	4.87 ± 0.31
Reactants to Products	4.84 ± 0.34

The connected DMG shows improvement over the normal DMG. This confirms our hypothesis that information exchange between corresponding atoms in reactants and products provides an important inductive bias for barrier height prediction. Unidirectional connections between reactants and products significantly improve performance (MAE 4.84) compared to bidirectional connections (MAE 20.73). The similar performance of both directional approaches shows that there is no clear preferred flow.

7.2.5 Message Passing and Attention

In Section 5.4, we proposed using ideas from the transformer architecture to overcome the disconnected nature of the Dual Molecular Graph (DMG) representation. Table 7.8 shows the results of implementing different attention mechanisms in combination with message passing neural networks.

Table 7.8: Comparison of attention mechanisms in hybrid MPNN-Transformer architectures (MAE in kcal/mol).

Properties	Reaction Attention	GPS Global Attention	GPS Reaction Attention
Test MAE	6.51 ± 0.38	21.86 ± 0.36	6.31 ± 0.33

The standard GPS approach with global attention performs poorly on our task with a Test MAE of 21.86 ± 0.36, which is similar to the baseline DMG results without any specialized architecture. This suggests that simply allowing all nodes to attend to all other nodes does not provide the right inductive bias for reaction barrier height prediction. In contrast, our proposed Reaction Attention mechanism significantly improves performance, achieving a Test MAE of 6.51 ± 0.38. The improvement confirms our hypothesis that targeted information exchange between reactants and products is crucial for accurate barrier height prediction. The GPS architecture with Reaction Attention delivers the best performance (Test MAE of 6.31 ± 0.33). These results validate our design hypothesis from Section 5.6 that attention mechanisms can effectively learn to model relationships between reactants and products without explicit atom-to-atom mapping information.

7.2.6 Reaction Graph Transformer

Building on our earlier findings about the importance of effective information exchange between reactants and products, we evaluated our proposed Reaction Graph Transformer (RGT) architecture. This pure transformer-based approach leverages specialized attention-masking strategies to create chemical-aware information flows within reaction graphs. Our results are shown in 7.9.

Table 7.9: Ablation study showing the effect of different components on model performance (MAE in kcal/mol).

Model	Test MAE
RGT	6.18 ± 0.30
- Line DMG $\rightarrow$ DMG	6.24 ± 0.31
- Remove separate processing	24.85 ± 0.66
- PMA $\rightarrow$ Global Add Pool	6.73 ± 0.36
- Remove PE	6.42 ± 0.29

The RGT achieved a test MAE of 6.18 ± 0.30 on the RDB7 dataset. This represents a significant improvement over the base DMG representation (22.15 ± 0.49) and is comparable to our separate pooling strategy (7.73-7.98). To understand the contribution of different components within our RGT architecture, we conducted an ablation study shown in Table 7.10.

The most important component was the separate processing of reactants and products, without which performance degraded substantially (MAE increased to 24.85 ± 0.66). This confirms our earlier hypothesis about the importance of distinguishing between reactant and product features during model training. The choice of pooling strategy also impacts performance, with Pooling by Multihead Attention (PMA) outperforming global additive pooling by approximately 0.5 MAE. The choice of graph representation (DMG vs. Line DMG) had minimal impact on performance, with only a small improvement when using the line graph transformation. Positional encodings provided a small benefit, improving MAE by approximately 0.24, suggesting that structural information beyond local connectivity is useful but not critical.

We also investigated different attention masking schemes as shown in Table 7.10, where each letter in the mask sequence represents the mask type used in a specific layer (e.g., "L-I-L-I-L-I-L-I" means alternating Local and Inter masks across eight layers, see Table 5.1 for all masking information). The best performance was achieved with the "L-L-L-L-L-I-I-I" configuration (MAE 5.91), which first uses 5 local connectivity masks followed by inter (reactant-product cross-attention) masks in the last three layers (note the similarity to a message passing and attention hybrid model from Section 5.5).

Table 7.10: Performance of different attention masking sequences (see Table 5.1) in Reaction Graph Transformer on a single fold of RDB7 (MAE in kcal/mol).

Masks	Test Mae
L-L-L-L-L-Om-Om-Om	7.73
L-L-L-L-L-L-L-L	7.49
S-S-S-S-S-S-S-S	7.26
L-L-L-L-L-In-In-In	6.98
L-S-L-S-L-S-L-S	6.87
L-Sm-L-Sm-Om-Om-I-I-I	6.75
L-L-L-In-In-I-I-I	6.64
L-L-Sm-Sm-Sm-I-I-I	6.28
L-Sm-L-Sm-L-I-I-I	5.98
L-I-L-I-L-I-L-I	5.93
L-L-L-L-L-I-I-I	5.91

7.2.7 Performance on Different Reaction Types

Table 7.11 shows our best-performing models on each dataset. The total amount of parameters in the PNA model is 498,113 while the GRT has 1,315,042 parameters. This increase stems mostly from the attention-based components.

Table 7.11: Performance comparison of mapped vs. unmapped graph representations across reaction datasets (MAE in kcal/mol).

Dataset	PNA on CGR	GRT on Line DMG
RDB7	4.32 ± 0.45	6.18 ± 0.30
E2	2.47	2.81
SN2	2.69	2.84
Cycloadd	2.83	3.24

The PNA model combined with the CGR representation outperforms the GRT on the line DMG on every dataset. Both the CGR and DMG perform much better on the datasets containing only one specific type of reaction. This suggests that reaction barrier height prediction is a much harder task when the model has to learn multiple types of reactions, and shows that machine learning models have trouble extrapolating to unseen reaction types. This has also been shown in previous work [VZS24].

The difference between the unmapped and mapped representation is however significantly smaller on datasets containing only one specific type of reactions. This shows that for one specific reaction type we do not have to rely on atom mapping as the GRT model can obtain a meaningful representation of the reaction.

CHAPTER 8

Conclusion

The goal of this thesis was to find the right inductive bias for reaction barrier height prediction focusing on graph representations and neural network architectures. Specifically, we explored methods that do not rely on atom-to-atom mapping as this information is typically not available to chemists. First, we clearly defined the representations considered: the CGR, the DMG, and the transformed representations. We then proposed a state-of-the-art benchmark of MPNNs on the CGR representation to properly set the bar for unmapped representations. We incorporated reaction information as an inductive bias into our architectures by (1) separately pooling reactant and product nodes, (2) introducing a reaction-specialized attention mechanism, and (3) proposing a Reaction Graph Transformer that uses masks to incorporate reaction information. Our empirical findings show that augmented MPNNs perform similarly on the CGR representation. For unmapped representations, we found that the direct application of MPNNs to the DMG gives poor results, but **incorporating reaction-specific inductive biases substantially improves performance**. Specifically, separately pooling reactant and product nodes significantly reduces the MAE, while our proposed Reaction Attention mechanism further improved results. The Reaction Graph Transformer with specialized attention masking strategies achieved the best performance among unmapped approaches and is the **current SOTA for reaction barrier height predictions using non-atom-mapped representations**. While mapped representations still outperform unmapped ones on diverse reaction datasets, the performance gap narrows significantly on single reaction-type datasets. A second equally important contribution of this work is the **development of a flexible, modular codebase for reaction property prediction** that enables experimentation with different representations and architectures. This framework facilitates reproducibility and future research in this field.

8.1 Future Work

Here, we outline potential directions for future research that could improve our methods or contribute to the machine learning for chemical reactions field.

8.1.1 Building New Datasets

Similar to the advancements seen in NLP and computer vision, we need bigger and higher-quality datasets to accelerate progress in the field. Current datasets have several limitations that should be addressed in future work. First, most available datasets focus on single reaction types or limited subsets of reaction types, which restricts the generalizability of trained models. Second, there is a general lack of quality in the current benchmark datasets. Quality does not only refer to the accuracy of the computed reaction barrier heights but also to the reaction mechanisms themselves. A reaction might be valid quantum mechanically, but could never occur naturally. Analyzing these types of reactions and incorporating the insights obtained in a new dataset could drastically improve the field.

8.1.2 Graph Transformers

We argue that graph transformers are promising paradigms for the reaction property prediction. Their ability to exchange information globally offers a solution to the disconnected components in a reaction graph. A limitation of this work is that no existing graph transformers were benchmarked. A future line of research is to benchmark promising graph transformers and to incorporate reaction-specific inductive biases into the architectures, similar to the ones used in our proposed Reaction Graph Transformer.

8.1.3 Incorporating 3D Information

Reactions are inherently 3D processes and it has been shown that 3D information is important for predicting quantum mechanical properties of molecules [SBC+22, ZGD+23, LGH+24] and reactions [JZW+25, vGBB+24]. This work did not consider any 3D information and thus incorporating 3D information in our proposed methods would be a valuable research direction.

APPENDIX A

Message Passing Neural Networks

Chapter 5 presented a benchmark of several established Message Passing Neural Networks (MPNNs) applied to our task of reaction barrier height prediction. Message Passing Neural Networks form the foundation of graph-based learning by iteratively updating node representations through neighborhood information exchange. In this framework, each node's representation is updated by: (1) computing messages between connected nodes, (2) aggregating these messages, and (3) updating the node's representation with the aggregated information. After L message passing layers, each node's final representation $\mathbf{h}_u^{(l)}$ contains information from its L-hop neighborhood, which can then be used for node-level tasks or pooled into a graph-level representation. Here, we provide the specific mathematical formulations for the message passing step of each benchmarked architecture.

Graph Convolutional Networks (GCN). Graph convolutional networks [KW16] aggregate normalized neighbor features:

$$\mathbf{h}_u^{(l)} = \sigma\left(\sum_{v\in\mathcal{N}(u)\cup\{u\}} \frac{1}{\sqrt{\hat{d}_v \hat{d}_u}} \mathbf{h}_v^{(l-1)} \mathbf{W}^{(l)}\right). \tag{A.1}$$

Graph Isomorphism Networks (GIN). Shown in Equation A.2, GIN works by aggregating neighbor features with a learnable epsilon parameter, then transforming through an MLP [XHLJ18]. GIN can be extended to GINE (Equation A.3) by incorporating edge features by adding them to the neighbor node features before applying the ReLU activation [HLG+19].

$$\mathbf{h}_u^{(l)} = \mathrm{MLP}^{(l)}\left((1+\epsilon^{(l)}) \cdot \mathbf{h}_u^{(l-1)} + \sum_{v\in\mathcal{N}(u)} \mathbf{h}_v^{(l-1)}\right), \tag{A.2}$$

$$\mathbf{h}_u^{(l)} = \text{MLP}^{(l)}\left((1+\epsilon^{(l)}) \cdot \mathbf{h}_u^{(l-1)} + \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(l-1)}\right), \tag{A.3}$$

Gated Graph Convolutional Networks (GatedGCN). Residual Gated Graph Convolutional Networks incorporate gating mechanisms [BL17]:

$$\mathbf{h}_u^{(l)} = \mathbf{h}_u^{(l-1)}\mathbf{W}_1^{(l)} + \sum_{v \in \mathcal{N}(u)} \eta_{u,v} \odot \mathbf{h}_v^{(l-1)}\mathbf{W}_2^{(l)}, \tag{A.4}$$

where the gating mechanism $\eta_{v,u} = \sigma(\mathbf{h}_v^{(l-1)}\mathbf{W}_3^l + \mathbf{h}_u^{(l-1)}\mathbf{W}_4^l)$ controls information flow between nodes by applying a sigmoid function (σ) to the sum of transformed features from both the receiving and sending nodes.

Graph Attention Networks (GAT). GAT learns attention weights $e_{ij}^{k,l}$ for neighbor importance [VCC+17]. GATv2 differs from GAT by computing attention scores after applying a non-linearity to the combined features of nodes [BAY21].

$$\mathbf{h}_i^{(l+1)} = \text{Concat}_{k=1}^{K}\left(\text{ELU}\left(\sum_{j \in \mathcal{N}_i} e_{ij}^{k,l}\mathbf{U}^{k,l}\mathbf{h}_j^l\right)\right), \tag{A.5}$$

where $\mathbf{U}^{k,l} \in \mathbb{R}^{\frac{d}{K} \times d}$ are K linear projection heads.

Principal Neighbourhood Aggregation (PNA). PNA aggregates node features by: (1) processing through multiple permutation-invariant functions, (2) scaling, (3) concatenating, and finally (4) passing through an MLP [CCB+20].

APPENDIX B

Hyperparameter Search

In this appendix, we present the hyperparameter search spaces explored for each model architecture evaluated in our experiments (see Table B.1 and Table B.2). For each model, we conducted a grid search. All hyperparameter searches were performed on the first fold of the RDB7 dataset and under a constraint of 500k parameters (following the work of [DJL$^+$23]). This approach ensured fair comparison between architectures.

Table B.1: Hyperparameter searches for GCN and GINE models. Bold values indicate the best-performing configurations.

Model	Hyperparameter	Search Space
GCN	depth	[3, 6, **9**, 12]
	hidden channels	[**64**, 128, 256, 512]
	batch normalization	[True, **False**]
	activation	[**ReLU**, identity]
	dropout	[0, **0.05**, 0.1]
	residual connection	[**True**, False]
	feedforward NN	[**True**, False]
	learning rate	[**0.001**, 0.0005]
GINE	depth	[3, 6, 9, **12**]
	hidden channels	[64, **128**, 256, 512]
	batch normalization	[True, **False**]
	activation	[ReLU, **identity**]
	dropout	[0, **0.05**, 0.1]
	residual connection	[**True**, False]
	feedforward NN	[True, **False**]
	learning rate	[0.001, **0.0005**]

Table B.2: Hyperparameter searches for D-MPNN, GatedGCN, GAT, and PNA models. Bold values indicate the best-performing configurations.

Model	**Hyperparameter**	**Search Space**
D-MPNN	depth	[**3**, 6, 9, 12]
	hidden channels	[64, **128**, 256, 512]
	batch normalization	[True, **False**]
	activation	[**ReLU**, identity]
	dropout	[0, 0.05, **0.1**]
	residual connection	[**True**, False]
	feedforward NN	[**True**, False]
	learning rate	[**0.001**, 0.0005]
GatedGCN	depth	[3, 6, 9, **12**]
	hidden channels	[**64**, 128, 256, 512]
	batch normalization	[**True**]
	activation	[**ReLU**, identity]
	dropout	[0, **0.05**, 0.1]
	residual connection	[**True**, False]
	feedforward NN	[**True**, False]
	learning rate	[**0.001**, 0.0005]
GAT	depth	[3, 6, 9, **12**]
	hidden channels	[**64**, 128, 256, 512]
	batch normalization	[True, **False**]
	activation	[**ReLU**, identity]
	dropout	[0, **0.05**, 0.1]
	residual connection	[**True**, False]
	feedforward NN	[**True**, False]
	learning rate	[**0.001**, 0.0005]
	heads	[1, **4**]
PNA	depth	[3, 6, **9**, 12]
	hidden channels	[**64**, 128, 256, 512]
	batch normalization	[**True**, False]
	activation	[**ReLU**, identity]
	dropout	[0, **0.05**, 0.1]
	residual connection	[**True**, False]
	feedforward NN	[**True**, False]
	learning rate	[**0.001**, 0.0005]
	aggregators	[[**mean, max, min, std**], [mean, max, sum]]
	scalers	[[identity, amplification, attenuation], [**identity**]]

Overview of Generative AI Tools Used

I used the following AI tools to aid my research and writing:

- Chat models: Claude[1], DeepSeek R1-DeepThink[2]
- Language tools: DeepL Translate[3], Grammarly[4]

These tools helped with text restructuring, idea organization, language proofing and translation. The core ideas, analysis, and conclusions are entirely my own intellectual contribution.

[1] https://claude.ai/
[2] https://www.deepseek.com/
[3] https://www.deepl.com/
[4] https://www.grammarly.com/

List of Figures

List of Tables

List of Algorithms

Bibliography

[AEL+18] Derek T Ahneman, Jesús G Estrada, Shishi Lin, Spencer D Dreher, and Abigail G Doyle. Predicting reaction performance in c–n cross-coupling using machine learning. *Science*, 360(6385):186–190, 2018.

[AY20] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[BAY21] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[BJK+23] David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. Modelling local and general quantum mechanical properties with attention-based pooling. *Communications Chemistry*, 6(1):262, 2023.

[BJOL24] David Buterez, Jon Paul Janet, Dino Oglic, and Pietro Lio. Masked attention is all you need for graphs. *arXiv preprint arXiv:2402.10793*, 2024.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[BL17] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[BM96] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

[BM07] Rodney J Bartlett and Monika Musiał. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, 79(1):291–352, 2007.

[CABJ24] Shuan Chen, Sunggi An, Ramil Babazade, and Yousung Jung. Precise atom-to-atom mapping for organic reactions via human-in-the-loop machine learning. *Nature Communications*, 15(1):2250, 2024.

[CCB+20] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, 33:13260–13271, 2020.

[CJ22] Shuan Chen and Yousung Jung. A generalized-template-based graph neural network for accurate organic reactivity prediction. *Nature Machine Intelligence*, 4(9):772–780, 2022.

[CLLG$^+$23] Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. *arXiv preprint arXiv:2307.07107*, 2023.

[CSJ$^+$24] Gabriele Corso, Hannes Stark, Stefanie Jegelka, Tommi Jaakkola, and Regina Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(1):17, 2024.

[CSV85] Raymond E Carhart, Dennis H Smith, and RENGACHARI Venkataraghavan. Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Computer Sciences*, 25(2):64–73, 1985.

[CZZL25] Yahui Cao, Tao Zhang, Xin Zhao, and Haotong Li. Hirxn: Hierarchical attention-based representation learning for chemical reaction. *Journal of Chemical Information and Modeling*, 2025.

[DAMR19] A Filipa De Almeida, Rui Moreira, and Tiago Rodrigues. Synthetic organic chemistry driven by artificial intelligence. *Nature Reviews Chemistry*, 3(10):589–604, 2019.

[DB20] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[DBK$^+$20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[DDS16] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711. PMLR, 2016.

[DJL$^+$23] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[DLL$^+$21] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

[DQC+24] Yuheng Ding, Bo Qiang, Qixuan Chen, Yiqiao Liu, Liangren Zhang, and Zhenming Liu. Exploring chemical reaction space with machine learning models: Representation and feature perspective. *Journal of Chemical Information and Modeling*, 64(8):2955–2970, 2024.

[EPA25] EPAM. Indigo toolkit. `https://lifescience.opensource.epam.com/indigo/`, 2025. Accessed 1 March 2025.

[FL19] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[Fuj86] Shinsaku Fujita. Description of organic reactions based on imaginary transition structures. 1. introduction of new concepts. *Journal of Chemical Information and Computer Sciences*, 26(4):205–212, 1986.

[GPG20a] Colin A Grambow, Lagnajit Pattanaik, and William H Green. Deep learning of activation energies. *The journal of physical chemistry letters*, 11(8):2992–2997, 2020.

[GPG20b] Colin A Grambow, Lagnajit Pattanaik, and William H Green. Reactants, products, and transition states of elementary chemical reactions based on quantum chemistry. *Scientific data*, 7(1):137, 2020.

[GSR+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[GYG21] Johannes Gasteiger, Chandan Yeshwanth, and Stephan Günnemann. Directional message passing on molecular graphs via synthetic coordinates. *Advances in Neural Information Processing Systems*, 34:15421–15433, 2021.

[HG21] Esther Heid and William H Green. Machine learning of reaction properties via learned representations of the condensed graph of reaction. *Journal of Chemical Information and Modeling*, 62(9):2101–2110, 2021.

[HGC+23] Esther Heid, Kevin P Greenman, Yunsie Chung, Shih-Cheng Li, David E Graff, Florence H Vermeire, Haoyang Wu, William H Green, and Charles J McGill. Chemprop: a machine learning package for chemical property prediction. *Journal of Chemical Information and Modeling*, 64(1):9–17, 2023.

[HLG+19] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[HvRvL21] Stefan Heinen, Guido Falk von Rudorff, and O Anatole von Lilienfeld. Toward the design of chemical reactions: Machine learning barriers of competing mechanisms in reactant space. *The Journal of Chemical Physics*, 155(6), 2021.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[JTG23] Fabian Jogl, Maximilian Thiessen, and Thomas Gärtner. Expressivity-preserving gnn simulation. *Advances in Neural Information Processing Systems*, 36:74544–74581, 2023.

[JZW$^+$25] Yingzhao Jian, Yue Zhang, Ying Wei, Hehe Fan, and Yi Yang. Reaction graph: Toward modeling chemical reactions with 3d molecular structures, 2025.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[L$^+$06] G. Landrum et al. Rdkit: Open-source cheminformatics. `http://www.rdkit.org`, 2006. Accessed: April 29, 2025.

[LBG$^+$99] Andrew R Leach, John Bradshaw, Darren VS Green, Michael M Hann, and John J Delany. Implementation of a system for reagent selection and library enumeration, profiling, and design. *Journal of chemical information and computer sciences*, 39(6):1161–1172, 1999.

[LGH$^+$24] Shuqi Lu, Zhifeng Gao, Di He, Linfeng Zhang, and Guolin Ke. Data-driven quantum chemical property prediction leveraging 3d conformations with uni-mol+. *Nature communications*, 15(1):7104, 2024.

[LH17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[LSW24] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *arXiv preprint arXiv:2406.08993*, 2024.

[LSW25] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Unlocking the potential of classic gnns for graph-level tasks: Simple architectures meet excellence. *arXiv preprint arXiv:2502.09263*, 2025.

[LSZ+23] Baiqing Li, Shimin Su, Chan Zhu, Jie Lin, Xinyue Hu, Lebin Su, Zhunzhun Yu, Kuangbiao Liao, and Hongming Chen. A deep learning framework for accurate reaction prediction and its application on high-throughput experimentation data. *Journal of Cheminformatics*, 15(1):72, 2023.

[LWWL20] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

[MC19] Adam C Mater and Michelle L Coote. Deep learning in chemistry. *Journal of chemical information and modeling*, 59(6):2545–2559, 2019.

[MDK+22] Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Ladislav Rampášek, and Dominique Beaini. Gps++: An optimised hybrid mpnn/transformer for molecular property prediction. *arXiv preprint arXiv:2212.02229*, 2022.

[MLL+23] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pages 23321–23337. PMLR, 2023.

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[Par89] Robert G Parr. Density functional theory of atoms and molecules. In *Horizons of Quantum Chemistry: Proceedings of the Third International Congress of Quantum Chemistry Held at Kyoto, Japan, October 29-November 3, 1979*, pages 5–15. Springer, 1989.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[RBM23] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[RGD$^+$22] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[RH10] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[SBC$^+$22] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pages 20479–20502. PMLR, 2022.

[SC22] Thijs Stuyver and Connor W Coley. Quantum chemistry-augmented neural networks for reactivity prediction: Performance, generalizability, and explainability. *The Journal of Chemical Physics*, 156(8), 2022.

[Sch21] Philippe Schwaller. *Learning the Language of Chemical Reactions–Atom by Atom. Linguistics-Inspired Machine Learning Methods for Chemical Reaction Tasks.* PhD thesis, Universität Bern, 2021.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[SHR$^+$21] Philippe Schwaller, Benjamin Hoover, Jean-Louis Reymond, Hendrik Strobelt, and Teodoro Laino. Extraction of organic chemistry grammar from unsupervised learning of chemical reactions. *Science Advances*, 7(15):eabe4166, 2021.

[SJC23] Thijs Stuyver, Kjell Jorner, and Connor W Coley. Reaction profiles for quantum chemistry-computed [3+ 2] cycloaddition reactions. *Scientific Data*, 10(1):66, 2023.

[SLSL15] Nadine Schneider, Daniel M Lowe, Roger A Sayle, and Gregory A Landrum. Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *Journal of chemical information and modeling*, 55(1):39–53, 2015.

[SPG22a] Kevin Spiekermann, Lagnajit Pattanaik, and William H Green. High accuracy barrier heights, enthalpies, and rate coefficients for chemical reactions. *Scientific Data*, 9(1):417, 2022.

[SPG22b] Kevin A Spiekermann, Lagnajit Pattanaik, and William H Green. Fast predictions of reaction barrier heights: toward coupled-cluster accuracy. *The Journal of Physical Chemistry A*, 126(25):3976–3986, 2022.

[SPV^{+}21] Philippe Schwaller, Daniel Probst, Alain C Vaucher, Vishnu H Nair, David Kreutter, Teodoro Laino, and Jean-Louis Reymond. Mapping the space of chemical reactions using attention-based neural networks. *Nature machine intelligence*, 3(2):144–152, 2021.

[SVLR21] Philippe Schwaller, Alain C Vaucher, Teodoro Laino, and Jean-Louis Reymond. Prediction of chemical reaction yields using deep learning. *Machine learning: science and technology*, 2(1):015016, 2021.

[SVV^{+}23] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023.

[TRRG23] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023.

[TSCB22] Mohammadamin Tavakoli, Alexander Shmakov, Francesco Ceccarelli, and Pierre Baldi. Rxn hypergraph: a hypergraph attention model for chemical reaction representation. *arXiv preprint arXiv:2201.01196*, 2022.

[VCC^{+}17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[vGBA^{+}24] Puck van Gerwen, Ksenia R Briling, Yannick Calvino Alonso, Malte Franke, and Clemence Corminboeuf. Benchmarking machine-readable vectors of chemical reactions on computed activation barriers. *Digital Discovery*, 3(5):932–943, 2024.

[vGBB^{+}24] Puck van Gerwen, Ksenia R Briling, Charlotte Bunne, Vignesh Ram Somnath, Ruben Laplaza, Andreas Krause, and Clemence Corminboeuf. 3dreact: Geometric deep learning for chemical reactions. *Journal of Chemical Information and Modeling*, 64(15):5771–5785, 2024.

[vRHBvL20] Guido Falk von Rudorff, Stefan N Heinen, Marco Bragato, and O Anatole von Lilienfeld. Thousands of reactants and transition states for competing e2 and s2 reactions. *Machine Learning: Science and Technology*, 1(4):045026, 2020.

[VSP^{+}17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[VZS24] Sai Mahit Vadaddi, Qiyuan Zhao, and Brett M Savoie. Graph to activation energy models easily reach irreducible errors but show limited transferability. *The Journal of Physical Chemistry A*, 128(13):2543–2555, 2024.

[Wei88] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[WJW+21] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[XHLJ18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Yad19] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[YCL+21] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[YSJ+19] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[YYL20] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

[ZGD+23] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. 2023.