3D Graph Representation Learning for Transition State Generation

University College London
Division of Medicine

This research project is submitted for the degree of
Master of Science in Precision Medicine

Candidate Number:
KVSQ4

Supervisor:
Prof. Brooks Paige

Date of Submission: November 6, 2021

Abstract

While machine learning has become a common computational framework for designing drugs, less attention has been paid to the feasibility of synthesising these molecules. By predicting transition states from reactants and products, we can characterise synthesis routes better to optimise reactions, a core aspect of synthetic feasibility. Reaction modelling research has focussed on text- and 2D graph-based representations of molecules but 3D molecular graphs can map reaction space in ways that account for spatial constraints.

In this project, we propose a transition state generation framework TSED based on 3D graph representation learning. TSED is a flexible encoder-decoder model where the encoder is a 3D graph neural network which can learn structure-preserving representations of 3D molecules. We use this encoder to create representations of reactants and products which we combine to create a transition state representation and then decode to a transition state geometry. Given that datasets in reaction modelling are typically small and biased to classes of reactions, we provide uncertainty estimates on the existing model in the literature, ts_gen, and on two implementations of our framework. We find that ts_gen produces variable outputs and thus, is underdetermined by its training data, presenting avenues for future work in uncertainty-quantified transition state generation.

Since readers may be unfamiliar with machine learning or reaction chemistry, we have included a glossary in Appendix A to use as a reference for key concepts.

Acknowledgements

To my supervisor **Brooks**, thank you for taking on an ML-curious student from the Division of Medicine. I really enjoyed our discussions and your ability to solve problems I'd been thinking about for a week in our hour-long meetings helped me complete this project.

To my friend **Dario Mongiardi**, thank you for providing helpful resources at the start of this project and entertaining my (in hindsight!) disjointed thoughts on chemistry. To **Lucky Pattanaik**, thanks for sending me your code - you saved me an enormous amount of time! I'd also like to thank **Gregor Simm** for his words earlier in the project which helped shape its eventual direction.

Abbreviations

AI artificial intelligence. 1

BMA Bayesian model average. 16, 24

CASP Computer-assisted synthesis planning. 5, 14, 24

CI confidence intervals. 25, 27

DMTA design-make-test-analyse. 1, 35

EGNN Equivariant Graph Neural Network. 13

G2C graph-to-coordinates. 21–23, 26

GNN graph neural network. 2–4, 11, 14, 22–24

GRL graph representation learning. 3, 9, 22, 34

IDM interatomic distance matrix. 18, 19

MAP maximum a posteriori. 15

MIT Massachusetts Institute of Technology. 4, 23, 24, 32

MITD MIT Dataset. 21

ML Machine learning. 2, 14, 18, 21

MLE maximum likelihood estimation. 14

NLWLS non-linear weighted least squares. 21–23

TS transition state. 2, 5, 6, 8, 9, 20–24

TSED transition state encoder-decoder. 23, 31, 32

UQ uncertainty quantification. 14, 24

Contents

Chapter 1

Introduction

1.1 Motivation

Generative models[1] have become a design framework for molecular synthesis problems in medicinal chemistry and materials science. Although research has progressed rapidly in this subfield, very few publications have verified their results in the lab due to equipment and material costs (Coley 2021). And the pharmaceutical industry has been slower to adopt these techniques because logistical considerations like viable synthesis routes and safety assessments have received less attention (Bender & Cortes-Ciriano 2020). In response to this, Bender & Cortes-Ciriano (2021) have recommended shifting the focus of artificial intelligence (AI) in drug discovery from "what drugs can we make?" to "what is the most effective drug for our task, and how do we produce it?" - more simply, from "what can we do?" to "what should we do?". Taking a birds-eye view of the design-make-test-analyse (DMTA) cycle in drug discovery (shown in Figure 1.1), "what should we do?" translates to integrating the *design* and *make* sections for more feasibility-aware traversal of chemical space. To this end, newer generative models have sought to integrate synthesisability into their data processing (Bradshaw et al. 2019, 2020).

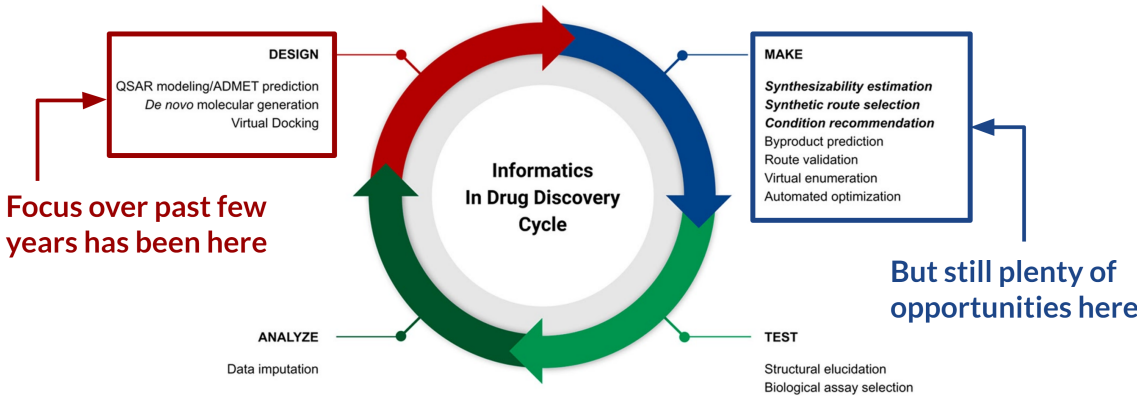


Figure 1.1: The design-make-test-analyse cycle in drug discovery. Our problem falls under synthetic route selection and automated optimisation in the blue box. Adapted from Struble et al. (2020).

[1]Models that learn the structure of input data in order to generate realistic data not present in the original input.

Another key aspect of maximising synthetic feasibility involves finding optimal reaction pathways for known drug molecules. Given the starting reactants are also known, this problem is equivalent to modelling reaction systems in chemical space. For accurate modelling, we need to calculate hundreds of reaction rate coefficients but direct calculations are often imprecise with non-negligible errors and high uncertainty (Bhoorasingh & West 2015). By mapping out reaction space, we can estimate these rate coefficients more accurately. Using the toolkit of transition state (TS) theory, one method to do this is by interpolating between reactants and products along a reaction pathway by predicting TSs.

Using conventional computational chemistry methods, predicting TSs is tractable for molecules with fewer than 50 atoms but for larger drug molecules, calculations become expensive and the methods cannot scale. With progress in graph neural network (GNN) research and publication of quantitative reaction datasets, Machine learning (ML) has become a scalable alternative for modelling chemical reactivity (Jorner, Tomberg, Bauer, Sköld & Norrby 2021). GNNs are useful in chemistry because they can learn memory-efficient representations of molecules that encode structural information. Most research in this area has focussed on representing molecules as *SMILES*[2] strings or 2D graphs since they are easier to process with standard GNN, but the field is shifting towards 3D graphs where we can also define how atoms and bonds relate to each other in space.

Since TSs do not have well-defined Lewis structures[3], it is difficult to interpolate between reactants and products using these standard representations. TSs can be distinguished more easily from other reaction molecules by considering molecular geometries. By parameterising our chemical space with geometry features and utilising newer physics-inspired GNNs that can handle spatial information, we can frame our problem as a 3D graph representation learning problem for TS geometry prediction.

1.2 Objective & Challenges

An intuitive way to think about our problem is a more involved averaging problem. Imagine working in video stabilisation - you may have a sequence that is choppy because a frame is missing between another pair of consecutive frames i and j. To stabilise the video, you could insert a frame in between i and j that is the average of both of them so that the transition between the frames is smoother, as shown in Fig 1.2. TS prediction is a similar problem in that we are trying to

[2]SMILES is a chemical language for describing molecules and reactions.
[3]Lewis structures are diagrams that show bonding between atoms.

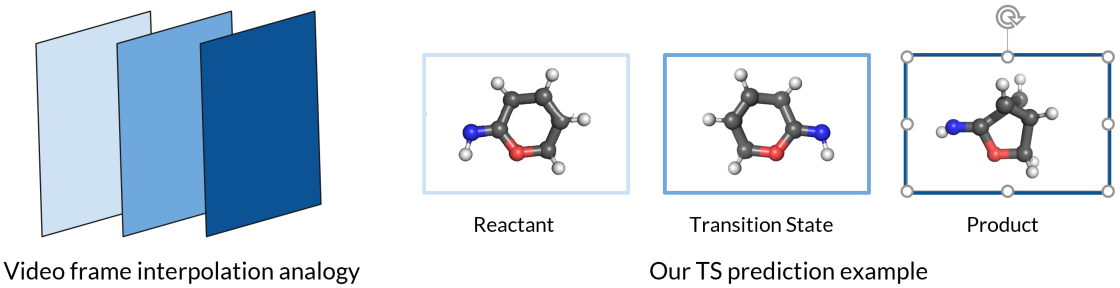


Figure 1.2: Video frame stabilisation analogy for our transition state geometry prediction task. Both are complex averaging problems.

find the "in-between" stage of a reaction start and end point, given by a reactant and a product. Like video stabilisation, a simple averaging procedure is unable to capture the complexity of the path between start and end points. Averaging reactant and product positions is insufficient to describe atomic movements and bonds breaking and forming as a reactant transforms into a product. For this reason, statistical methods that can deal with more complex search spaces - namely, machine learning - serve as a useful framework to frame interpolation between reactant and product geometries.

Consequently, the key challenge for us is to constrain the large parameter space spanned by possible 3D representations of reaction molecules and process the molecules within this space efficiently. To do this, we formulate our problem through the lens of graph representation learning (GRL) and use novel GNNs designed to process 3D graph-structured data.

Another key challenge is dealing with a biased reaction data which serves as a proxy to more diverse drug reaction data. In this project, we have used one of two available reaction geometry datasets which contain less than 20,000 datapoints and have been created on specific classes of molecules (Grambow et al. 2020). Our reaction molecules contain fewer than 25 atoms which is well below most traditional small molecule drugs. Given this constrained dataset, we aim to provide good uncertainty estimates on our predictions so that this proof-of-concept model can generalise better to future scenarios where more transition state data is available.

1.3 Report Outline & Contributions

In this project, we investigate the question: "*Can we design a one-step machine learning workflow with uncertainty quantification to generate 3D graphs of transition states from 3D graphs of its reactant and product?*"

To answer this, we have the following contributions:

- PyTorch implementation of TensorFlow model for transition state generation by authors at the Massachusetts Institute of Technology (MIT), `ts_gen`.
- Stability testing of the MIT model using Bayesian approaches.
- Representation learning framework for transition state generation.
- Empirical comparison of two 3D GNN encoders within our framework for transition state generation and corresponding uncertainty quantification.

We start by describing the theoretical foundations for our project in Chapter 2, and how our transition state prediction problem fits into the overall fields of computer-assisted synthesis planning, graph machine learning for molecules, and uncertainty quantification. We build off this foundation in Chapter 3 and summarise related machine learning research for 3D molecule generation and uncertainty-quantified reaction prediction.

In Chapter 4, we relate the theoretical foundations and related work to formulate transition state generation as a 3D representation learning problem. We also outline our experiments to answer the above question, giving the results and discussion in Chapter 5. We finish in Chapter 6 with a review of our project's limitations and possible avenues for future work.

Chapter 2

Background

2.1 Computer-Assisted Synthesis Planning

Computer-assisted synthesis planning (CASP) is broadly defined by four tasks: retrosynthesis, reaction prediction, reaction condition prediction, and reaction optimisation (Thakkar et al. 2021). These tasks combine to form systems for predicting and optimising synthesis routes to desired molecules. A related task underpinning synthetic route design is characterising the reaction space - the relevant subspace of chemical space for a reaction. While our problem of transition state geometry prediction falls loosely under reaction prediction and optimisation, we characterise it more as an exploration of reaction space because this enables us to frame it in the context of existing methods in computational chemistry (Section 2.1.2) and representation learning in machine learning (Section 2.2.1).

2.1.1 Transition States and Optimisation

A transition state (TS) is an "in-between" stage of a reaction between a reactant and product. The reactant bonds are partially broken and product bonds are partially made. The partial nature of the TS corresponds to unstable behaviour and a short lifetime (measured in femtoseconds), preventing it from being isolated during a reaction.

Looking at the reaction energy diagram in Figure 2.1, we see that a TS is a short-lived configuration of atoms at a local maximum reaction coordinate. A similar but distinct concept to a TS is a reactive intermediate which, instead of being a maximum, is an energy minimum - this level of stability means that intermediates are theoretically isolable unlike TSs. Examples of intermediates include free-radicals and carbocations.

While the above energy diagram shows a simple two-dimensional curve through chemical space, TS interpolation usually entails finding stationary points on multidimensional functions, also known as *optimisation* (Fletcher 2013). Molecules exhibit properties that make optimisation difficult using standard first order methods. The coordinates of atoms that dictate the property can be highly coupled and

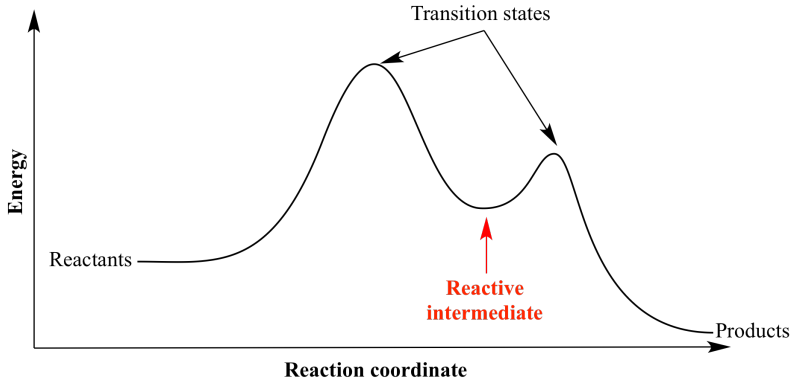


Figure 2.1: An example reaction energy diagram with two steps indicated by the presence of two transition states (as local maxima) and one reactive intermediate (as a local minimum). We aim to parameterise the energy by features from 3D molecular graphs.

so optimising along one reaction variable in the parameter space can greatly minimise progress in another. To get round this, we focus on the multidimensional curve describing reaction progression parameterised by features from 3D molecular graphs: atom features, bond features, and distance geometry features (we describe this in more detail in Section 2.2). Here, the local optima still correspond to TSs, but we need different methods to explore the associated geometry-based reaction space.

2.1.2 Reaction Space Exploration

Most traditional algorithms for reaction space exploration build off quantum chemistry principles (Unsleber & Reiher 2020), ranging from molecular dynamics to growing string methods. These methods are situated on the left of Fig 2.2. Molecular dynamics and similar forcefield-based methods require exploration of potential energy surfaces which are reliant on expensive density functional theory calculations. While these methods give mechanistic insight, they have poor scalability and do not generalise well to unseen reaction spaces. Further along the spectrum (but still on the left-hand side), we have reaction path models like the growing string method. These methods utilise quantum chemistry-based heuristics, like graph transformation rules, in order to speed up reaction exploration (Simm et al. 2018). While these methods are more efficient than pure quantum chemistry methods, they still lack the generalisability needed for complete coverage of chemical space, and are challenging to compose sequentially for realistic multistep reaction pathways.

In contrast to traditional reaction space exploration which has mechanistic methods for singular scenarios, ML methods are generally applicable for all types of reactivity problems at the cost of mechanistic interpretation. These methods are

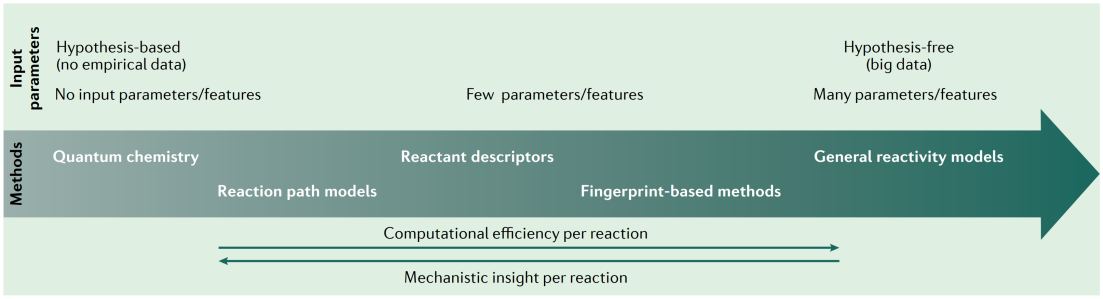


Figure 2.2: The spectrum of computational methods for modelling chemical reactivity. The models on the left give mechanistic insight but are computationally inefficient. Models on the right are generally applicable but can lack mechanistic interpretation and require more data for high performance. Image from (Jorner, Tomberg, Bauer, Sköld & Norrby 2021).

situated on the right hand side of the Fig 2.2 spectrum. ML scales better to larger datasets and their flexibility allows them to take different representations of data as input, which we discuss in Section 2.2.1. In particular, graph machine learning algorithms that preserve molecular properties and structure in their processing operations provide an alternative framework to classical methods which are still rooted in chemistry (Atz et al. 2021). Beyond scalability, automated error correction and uncertainty calibration are two of the key targets for reaction exploration algorithms (Unsleber & Reiher 2020). Both have well-formulated corollaries in machine learning - "learning" algorithms correct for performance errors by definition, and uncertainty calibration is possible by quantifying the statistical reliability of outputs using Bayesian frameworks.

2.1.3 Representing Chemical Data

The starting point for ML algorithms in chemistry is selecting an appropriate representation of molecular data (or *molecular fingerprint*) as shown in Figure 2.3. Vector formats are the simplest representation with each element representing the presence of a chemical substructure. Strings, either in SMILES or InChI formats, are text-based representations of molecules which can be processed by natural language processing algorithms (Schwaller et al. 2019). While strings encode molecular connectivity, 2D molecular graphs, with nodes and edges representing atoms and bonds, give a more intuitive representation of how atoms relate to each other. 3D graphs go one step further and include positional information about how atoms relate to each other in physical space.

Reactions are given as combinations of reactant, product and intermediate molecules, and potentially other reaction conditions. If the reaction data is just made up of molecules, we can represent reactions (i.e. produce *reaction fingerprints*) using one of the above representations for each molecule. For both molecular and reac-

tion fingerprints, these features can be hand-crafted before being used as input or learned during the ML processing, as described in Section 2.2.1.

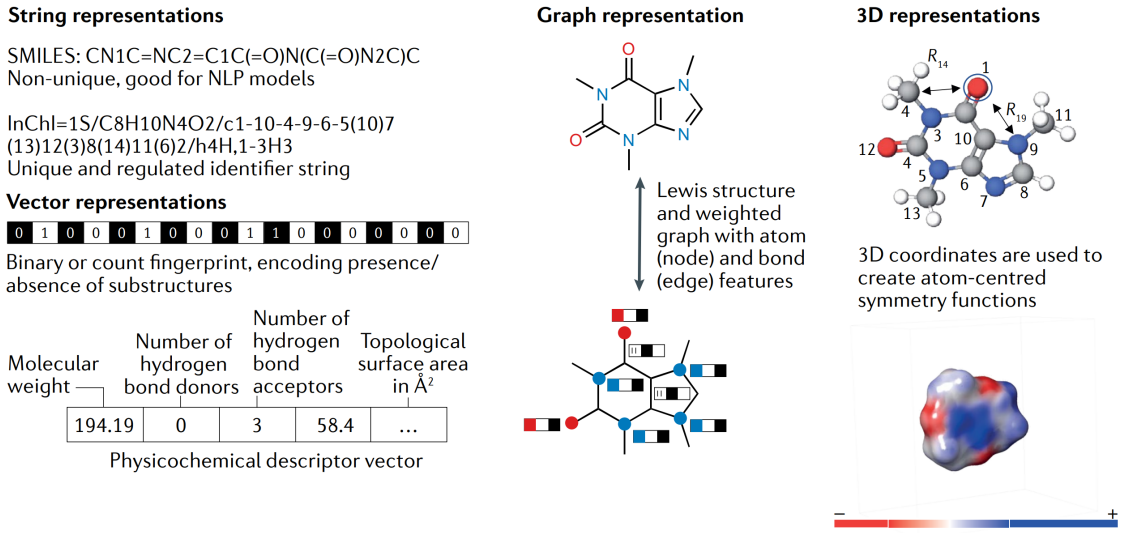


Figure 2.3: A caffeine molecule expressed as string, vector, 2D graph, and 3D graph representations. Image from Jorner, Tomberg, Bauer, Sköld & Norrby (2021).

2.2 Graph Machine Learning for Molecules

Machine learning is a subset of artificial intelligence that aims to describe patterns in a given dataset. *Learning* is the process by which algorithms iteratively improve by responding to their performance on a given task. These algorithms are based on statistical models containing unknown or uncertain parameters and so learning can be defined as discovering the model and associated parameters that improve performance (van der Wilk 2019).

Graph machine learning is machine learning on graph-structured data (Gaudelet et al. 2021). A graph is a mathematical formalism of a network. Networks appear in various incarnations, from biological pathways in the body, to social networks connecting users on Facebook and Twitter. Molecules are small networks composed of atoms and bonds. In this project, we are studying reactions, each composed of three molecules: a reactant, a product, and a TS.

2D graphs are defined by a set of vertices V and a set of edges E such that $G_{2D} = (V, E)$. 3D graphs are given by $G_{3D} = (V, E, \{\mathbf{r_i}\}_i^n)$, where $\mathbf{r_i}$ is position vector of each node in the overall graph, usually given in the Cartesian format (x, y, z). These 3D graphs, also known as geometric or coordinate-based graphs, go one step beyond the classical 2D graph and capture spatial information in the data. For problems that can be framed in terms of relative positions in space, like TS

geometry prediction, these graphs prove useful. We have three 3D graphs for the reactant, product, and TS. Each graph describes the molecule's atoms in terms of their chemical properties (the vertices $\mathcal{V}$), the bonds that connect them (the edges E), and the positions of each atom in space (the position vectors $\{\mathbf{r}_i\}$).

2.2.1 Graph Representation Learning

Representation learning builds off the premise that complex, high-dimensional data can be *encoded* as (i.e. transformed into) an *embedding*. An embedding is an alternative expression of data that preserves the data's original structure. Consequently, representation learning tasks often aim to learn embeddings so they can be used for other downstream tasks in a computationally-efficient manner (Bengio et al. 2013).

Different types of data require different encoding procedures. Graphs are powerful not just because of their generality, but because they capture relationships between vertices - they capture *structure* in the data. Learning over graphs is enhanced by identifying and utilising this structure. There are two main ways to do this: with hand-crafted features or by learning structure-preserving representations. While hand-crafted features can be effective, they are often tedious to create and cannot adjust to the learning process. Graph representation learning (GRL) is the alternative approach which aims to learn embeddings that preserve data structure by utilising inductive biases (Hamilton 2020).

Graph Encoder-Decoder Models

The encoder transforms input data into the embedding space and the decoder transforms embedding data into the space required for a given task. An graph encoder-decoder model is designed to encode structural features of graphs. Namely, the encoder transforms the node features, edge features, and adjacency matrix into their respective embedding spaces. These feature embeddings can then be combined in some way to get a single graph embedding. The decoder can then be used in two ways: to reconstruct the original graph or for another task. If the decoder reconstructs the original graph (i.e. original node, edge, and adjacency matrix features) from the embedding, the overall model is called an *autoencoder*. If the decoder decodes to anything other than the original structure for another task, the overall model is called an encoder-decoder model. While the encoder and decoder are usually given as neural networks, they can take on other models that are able to produce the same required output type.

2.2.2 Relational Inductive Biases

While a core aim of machine learning is developing systems that generalise to new environments, giving a model unlimited capacity[1] makes it prone to overfitting. Inductive biases are a set of constraints which limit the model's output to realistic values for the problem domain, which (almost paradoxically) improves its generalisability. We can impose inductive biases in different ways - there are classical methods like regularisation while Bayesian methods place a prior over model parameters.

The shift towards representation learning and deep learning in the early-mid 2010s has inspired work investigating *relational* inductive biases (RIBs) which are model constraints on the structure of data (Battaglia et al. 2018). We are concerned with two concepts within RIBs: *equivariance* and *invariance* for data transformations. If a transformation is equivariant, transforming the input data will transform the model output in the same way. If a transformation is invariant, transforming the input data will produce the same model output.

Formally, we can represent these transformations as elements of a group $\mathbf{G}$. We are concerned with linear group actions since they can be represented using linear algebra and so integrate well with machine learning. Linear group actions are given as $\rho : \mathbf{G} \to \mathbb{R}^{n \times n}$. A function f is $\mathbf{G}$-equivariant if the group action affects the output in the same way such that $f(\rho(g(x))) = \rho(g)f(x)$. A function f is $\mathbf{G}$-invariant if its output is unaffected by the group action on the input such that $f(\rho(g(x))) = f(x)$ (Atz et al. 2021).

A well-known example is the translational equivariance of convolutional neural networks (CNNs) on images. CNNs apply a translationally-equivariant operation called a convolution which means if you shift an image by x units, then the convolved image is proportionally shifted by x units. Since we have 3D molecular graphs as input, we are mainly concerned with the {equi/in}-variance of the 3D rotation group (also known as the Special Euclidean 3 or SE(3) group) - essentially, if we rotate an input molecule, the output should be the same or transformed appropriately. By building this {equi/in}-variance into deep learning transformations (i.e. neural network layers), we produce a robust model grounded in chemical reality.

[1] Capacity refers to a model's complexity, often given crudely as the number of model parameters. High capacity allows a model to learn a wide class of functions which becomes an issue if it simply learns a one-to-one mapping from training examples to their solutions (i.e. it overfits).

2.2.3 Graph Neural Networks

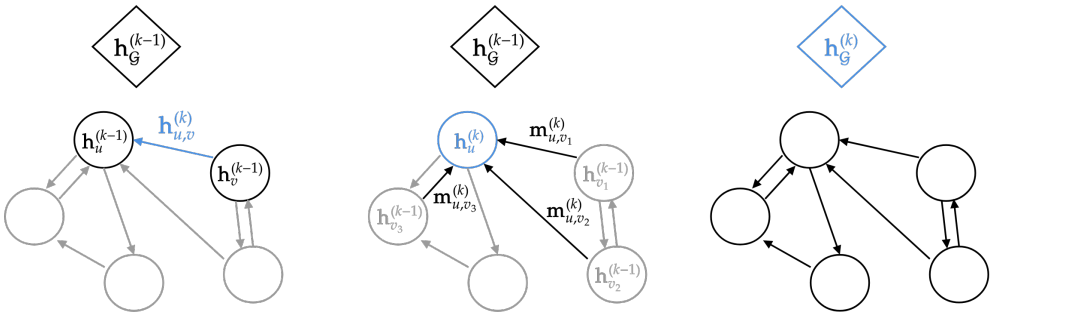
Graph neural networks (GNNs) are an extension of classical neural networks for graph-structured data. The defining characteristic of the GNN framework is *message passing* where vertices send messages as vectors to other vertices and are then updated using neural networks (Gilmer et al. 2017, Hamilton 2020). By iteratively performing the message passing procedure multiple times, vertices are updated to encode more information about their local graph neighbourhood, in a similar way that image convolutions encode information about a subgrid of an overall image grid. The major difference between operations on graph-based data and grid-based data is that graph operations need to be permutation-invariant i.e. the operations are independent of the order of neighbouring nodes since there is no canonical way of ordering them like in images.

Node features describe the information associated with each node and edge features describe how nodes are related to each other. Message vectors which are dependent on both features are better at capturing the structure of the original graph. Battaglia et al. (2018) combined this idea with a graph update function for a generalised message passing formulation. Given a 2D input graph $\mathcal{G}_{2D} = (\mathcal{V}, \mathcal{E})$, node features $\{\mathbf{n}_u\}\ \forall u \in \mathcal{V}$, and edge features $\{\mathbf{e}_{u,v}\}\ \forall u, v \in \mathcal{V}$, and local node neighbourhoods $\mathcal{N}(u)\ \forall u \in \mathcal{V}$, the message passing aims to produce node embeddings $\mathbf{h}_u\ \forall u \in \mathcal{V}$, edge embeddings $\mathbf{h}_{u,v}\ \forall (u,v) \in \mathcal{V}$, and a whole graph embedding $\mathbf{h}_{\mathcal{G}}$. The kth message passing iteration is given by the following four functions, illustrated by Figure 2.4:

$$
\begin{aligned}
\mathbf{h}_{u,v}^{(k)} &= \underset{\textit{edge embedding}}{\text{UPDATE}}\left(\mathbf{h}_{u,v}^{(k-1)},\ \mathbf{h}_{u}^{(k-1)},\ \mathbf{h}_{v}^{(k-1)},\ \mathbf{h}_{\mathcal{G}}^{(k-1)}\right) \\
\mathbf{m}_{\mathcal{N}(u)}^{(k)} &= \underset{\textit{node messages}}{\text{AGGREGATE}}\left(\{\mathbf{h}_{u,v}^{(k)},\ \forall v \in \mathcal{N}(u)\}\right) \\
\mathbf{h}_{u}^{(k)} &= \underset{\textit{node embedding}}{\text{UPDATE}}\left(\mathbf{h}_{u}^{(k-1)},\ \mathbf{m}_{\mathcal{N}(u)}^{(k)},\ \mathbf{h}_{\mathcal{G}}^{(k-1)}\right) \\
\mathbf{h}_{\mathcal{G}}^{(k)} &= \underset{\textit{graph embedding}}{\text{UPDATE}}\left(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \{\mathbf{h}_{u}^{(k)},\ \forall u \in \mathcal{V}\},\ \{\mathbf{h}_{u,v}^{(k)},\ \forall (u,v) \in \mathcal{E}\}\right)
\end{aligned}
$$

Message Passing over 3D Molecular Graphs

Unlike standard frameworks for embedding 2D graphs, embeddings of 3D graphs need to preserve the relative positional information of the input nodes. Essentially, there is an added level of structure in the data that needs to be considered. Since 2018, different architectures inspired by chemical physics have been published to deal with 3D molecular graphs. These architectures incorporate transformations that are equivariant to the rotational group SE(3) (described in Section 2.2.2).



(a) Update edge embedding (b) Aggregate messages and update node embedding (c) Update graph embedding using updated feature embeddings

Figure 2.4: Feature embedding updates in the kth (i.e. in a single) message passing iteration. Blue tells us the feature embedding being updated, black the other feature embeddings and processing involved, and grey, the unused features. The box with $\mathbf{h}_{\mathcal{G}}$ represents the overall graph embedding. Image inspired by Battaglia et al. (2018).

Since position vectors expressed as Cartesian coordinates are not equivariant to translation and rotation, alternative SE(3)-equivariant spatial representations are required. Spherical coordinates are one system that can convert the original Cartesian position vectors $\mathbf{r_i}$ for each atom i to features describing SE(3)-equivariant relationships between atoms. Spherical coordinates describe atomic relationships by 3-tuples $\mathbf{s_{ijkl}} = (d_{ij}, \theta_{ijk}, \phi_{ijkl})$ between four atoms i, j, k, and l. d is the bond length between two atoms, θ is the bond angle between three angles, and ϕ is the torsion (or dihedral) angle between four atoms.

This generalised formulation with three types of equivariant features is memory-intensive and potentially unnecessary for our task. We simplify the general framework for spherical message passing proposed by Liu et al. (2021) to just use interatomic distance vectors $\{\mathbf{d}\}$ instead of the full 3-tuple. Hence, we describe a 3D molecular graph as $\mathcal{G}_{3D} = (\mathcal{V},\ \mathcal{E},\ \mathbf{d}, \mathbf{g})$, where $\mathcal{V}$ are the atom features, $\mathcal{E}$ are the bond features, $\{\mathbf{d_{i,j}}\}$ are the interatomic distances for each atom pair i and j, and $\mathbf{g}$ is the global graph feature vector. With this framework the general message-passing functions are as follows for the kth iteration:

$$
\begin{aligned}
\mathbf{h}_{u,v}^{(k)} &= \underset{bond\ embedding}{\text{UPDATE}}\left(\mathbf{h}_{u,v}^{(k-1)},\ \mathbf{h}_{u}^{(k-1)},\ \mathbf{h}_{v}^{(k-1)},\ \mathbf{d}_{u,v}^{(k-1)},\ \mathbf{h}_{\mathcal{G}}^{(k-1)}\right) \\
\mathbf{m}_{\mathcal{N}(u)}^{(k)} &= \underset{atom\ messages}{\text{AGGREGATE}}\left(\{\mathbf{h}_{u,v}^{(k)},\ \forall v \in \mathcal{N}(u)\}\right) \\
\mathbf{d}_{u,v}^{(k)} &= \underset{coordinate\ embedding}{\text{UPDATE}}\left(\mathbf{d}_{u,v}^{(k-1)},\ \mathbf{h}_{u,v}^{(k)},\ \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) \\
\mathbf{h}_{u}^{(k)} &= \underset{atom\ embedding}{\text{UPDATE}}\left(\mathbf{h}_{u}^{(k-1)},\ \mathbf{m}_{\mathcal{N}(u)}^{(k)},\ \mathbf{h}_{\mathcal{G}}^{(k-1)}\right) \\
\mathbf{h}_{\mathcal{G}}^{(k)} &= \underset{3D\ graph\ embedding}{\text{UPDATE}}\left(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \{\mathbf{h}_{u}^{(k)},\ \forall u \in \mathcal{V}\},\ \{\mathbf{h}_{u,v}^{(k)},\ \forall (u,v) \in \mathcal{E}\}\right)
\end{aligned}
$$

In this project, we focus on two existing architectures which implement this interatomic distance-based framework: SchNet (Schütt et al. 2018) which is SE(3)-invariant and EGNN which is SE(3)-equivariant (Satorras et al. 2021). Both these networks are also respectively invariant and equivariant for permutation (ordering of atoms).

SchNet. The SchNet library uses continuous filter convolutions to process 3D molecular graphs for energy prediction. Classical convolutional layers use discrete filter tensors on regularly-structured data like images (represented by a regular rectangular pixel grid). This is not applicable to molecules because atoms can be located at seemingly arbitrary positions without any obvious structure, which causes classical convolutions to change rapidly when moving from atom to atom. SchNet's continuous convolution gets around this problem by continuously sampling in regular distances from the centre of the molecule outwards enabling it to learn local relationships between atoms (Figure 2.5). SchNet does not explicitly update coordinate embeddings.

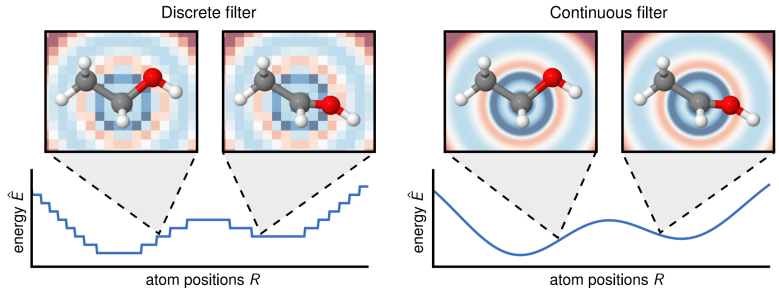


Figure 2.5: Visualisation of how the continuous convolution layer in SchNet works. The discrete filter (left) cannot capture small changes in atomic positions which gives a discontinuous potential energy surface while the continuous filter (right) can, giving a continuous potential energy surface. Image from Schütt et al. (2018).

Equivariant Graph Neural Network (EGNN). EGNN is a library for equivariant processing of graph-structured data. An equivariant graph convolution layer is used to update coordinate and atom embeddings given a 3D molecular graph. EGNN updates the position of each atom by the radially-normalised interatomic distances to preserve translational and rotational equivariance. EGNN differs from SchNet by including an explicit coordinate embedding update which interacts with the edge embedding update.

2.3 Uncertainty Quantification for Molecular GNNs

Chemical variability of molecules and precision limits of measuring instruments mean that chemical data are not always error-free, and are better described as estimates with noise (Ching et al. 2018). Using deep neural networks to learn over imprecise data has the ability to propagate noise through the model to the output. Molecular GNN models are also prone to overfitting because structural similarities, such as molecular scaffolds, make it hard to distinguish between representations. Overfitting results in inadequate uncertainty estimates, or worse, overconfidence in incorrect predictions. Similar concepts of structural overfitting apply in CASP research, where datasets are limited in size and only available for specific classes of molecules and reactions.

As described in Section 2.1, the key aim for ML in CASP is a tighter feedback loop between wet lab experimentation and computational science. For integrated model deployment with humans, we need reliability assessment on our outputs, and this is formulated mathematically as uncertainty quantification (UQ). UQ can help gauge prediction reliability and model robustness outside the training data domain when faced with new scaffolds in molecular property prediction (Hirschfeld et al. 2020), or with different reaction spaces in CASP (Schwaller et al. 2019, Jorner, Brinck, Norrby & Buttar 2021).

2.3.1 Neural Networks as Probabilistic Models

Viewed as a probabilistic model, neural networks (NN) are expressed as $P(y|x, \boldsymbol{\theta})$, where y is the data's label, x is the observed data, and $\boldsymbol{\theta}$ are the NN parameters. There are two main approaches to learning over this probabilistic formulation - *frequentist* and *Bayesian*.

The frequentist approach assumes the parameters of the model are fixed and the data is a random variable. The standard learning process is *maximum likelihood estimation* (*MLE*). Given data $\mathcal{D} = \{(y_i, x_i)\}_i$, we aim to learn model parameters which maximise the likelihood $P(\mathcal{D}|\boldsymbol{\theta})$:

$$\begin{aligned}
\boldsymbol{\theta}_{MLE} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ P(\mathcal{D}|\boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \prod_i P(y_i|x_i, \boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \log \prod_i P(y_i|x_i, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \sum_i \log P(y_i|x_i, \boldsymbol{\theta})
\end{aligned}$$

The Bayesian framework makes the opposite assumptions - the data is assumed fixed and model parameters are random variables. We place a prior distribution over the model parameters $P(\boldsymbol{\theta})$, and, given data $\mathcal{D}$, we aim to learn a suitable posterior distribution $P(\boldsymbol{\theta}|\mathcal{D})$. Intuitively, the Bayesian approach tells us our degree of *belief*[2] in a model and can be iteratively updated based on new observations in order to quantify uncertainty in the learning process. This updating is facilitated by Bayes' theorem which lets us relate the posterior and prior:

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{D})}$$

where $P(\boldsymbol{\theta})$ is the prior describing our degree of belief in $\boldsymbol{\theta}$ before observing $\mathcal{D}$, $P(\boldsymbol{\theta}|\mathcal{D})$ is the posterior describing our degree of belief in $\boldsymbol{\theta}$ after observing $\mathcal{D}$. We relate the posterior and prior by the likelihood $P(\mathcal{D}|\boldsymbol{\theta})$ which measures how well $\boldsymbol{\theta}$ explains $\mathcal{D}$, normalised by the marginal likelihood $P(\mathcal{D})$.

Often, Bayesian learning is viewed as solving the *maximum a posteriori* (MAP) objective. Placing a prior $P(\boldsymbol{\theta})$ on the parameters produces a MAP objective that is equivalent to the MLE objective with an additional regularisation term:

$$\begin{aligned}
\boldsymbol{\theta}_{MAP} &= \underset{\boldsymbol{\theta}}{\text{argmax}}\ \log P(\boldsymbol{\theta}|\mathcal{D}) \\
&= \underset{\boldsymbol{\theta}}{\text{argmax}} \log P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\text{argmax}}\ \log P(\mathcal{D}|\boldsymbol{\theta}) + \log P(\boldsymbol{\theta}) = \boldsymbol{\theta}_{MLE} + \underset{\boldsymbol{\theta}}{\text{argmax}}\ \log P(\boldsymbol{\theta})
\end{aligned}$$

However, Wilson & Izmailov (2020) argues that marginalisation over many parameter configurations is the defining characteristic of Bayesian learning. Wilson states that although the MAP objective involves a prior, a posterior, and invokes Bayes' rule, by aiming to learn a point estimate, it becomes a classical approach. We adopt Wilson's view of Bayesian learning as based in marginalisation and thus distribution, and describe this in the following section.

2.3.2 Approximate Bayesian Inference

As neural networks get deeper and incorporate different components into their architecture, the Bayesian formulation requires learning a computationally-intractable number of parameters. In these cases where we cannot compute our beliefs, we turn to methods for approximate inference. There are two methods we use in this project: deep ensembles(Lakshminarayanan et al. 2017) and dropout (Srivastava et al. 2014). Deep ensembles average several different networks trained on dif-

[2]Belief and uncertainty are two sides of the same coin. Both are represented by probability distributions and tell us how much we (think we) do or do not know about our data.

ferent hyperparameter initialisations while dropout averages the space of weight parameters for one model. Both aim to build a posterior distribution for the neural network, called a *Bayesian model average* (BMA), that encodes different configurations of model parameters.

We formalise these concepts by describing how we approximate the Bayesian posterior. Given our dataset $\mathcal{D}$ of inputs X and outputs Y, we can invoke Bayes' theorem to find the posterior $P(\boldsymbol{\theta}|\mathcal{D})$. We then marginalise over $\boldsymbol{\theta}$ to obtain the integral for the Bayesian model average (i.e. the distribution over all possible parameter settings) which represents model uncertainty $P(y|x, \mathcal{D})$. Finally, we sample k parameter configurations from this distribution and weight them equally in a sum to approximate the integral:

$$
\begin{aligned}
P(\boldsymbol{\theta}|\mathcal{D}) &= \frac{P(Y|X, \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(Y|X)} && \text{[Invoke Bayes' Theorem]} \\
P(y|x, \mathcal{D}) &= \int P(y|x, \boldsymbol{\theta})\ P(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} && \text{[Marginalise over } \boldsymbol{\theta} \text{ to get BMA]} \\
&\approx \frac{1}{k} \sum_{k} P(y|x, \theta^k) && \text{[Approximate with weighted sum]}
\end{aligned}
$$

Deep Ensembles. Deep ensembles, or simply 'ensembles', train the same network many times with different random initialisations to find the MAP (i.e. point) estimates of multiple local optima (Fig 2.6). Averaging the parameters of the output models gives us an approximate BMA. The advantage of the deep ensemble is that by averaging over potentially wide-ranging local optima, we gain a comprehensive understanding of model space and hence a better approximation to the BMA (Wilson & Izmailov 2020). The disadvantages are the large computational cost of running the model multiple times, and that the added benefit of wide coverage of the model space may not be realised if there are clearer global optima.

Dropout. Each neuron (i.e. linear unit) in a neural network learns to detect features in the input data by modifying its weight parameter. With a small labelled training set and deep network, there are many configurations of neuron weights that can fit the data and some may overfit. One of the reasons this overfitting arises is because of neuron co-dependence - rather than each neuron contributing equally to learn separate features of the input, neurons learn to predict accurately only when combined with other neurons. This co-dependence can be mediated by randomly "turning off" some of these neurons during training in a process called dropout, shown in Figure 2.7. When training with the standard stochastic gradient descent algorithm, we give an upper bound to the regularisation term of the objective to prevent weights from growing too large. While deep ensembles

give a broader coverage of model space, they incur significant computational cost with multiple training sessions. Dropout only requires one training session and so is a more efficient averaging procedure.

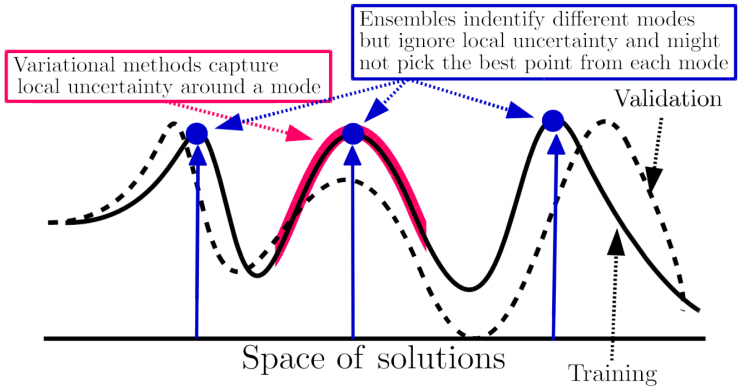
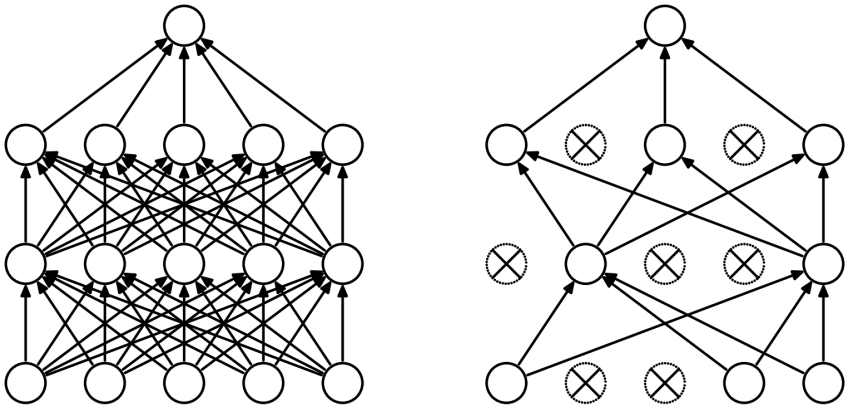


Figure 2.6: Example of a deep ensembles on a 1D solution space compared to variational methods, another class of approximate inference methods. Deep ensembles give a broader coverage of solution space. The x-axis are model parameter values while the y-axis plots the objective function values. Image from Lakshminarayanan et al. (2017).



(a) Standard Neural Net (b) After applying dropout.

Figure 2.7: A standard neural network (a) and then with dropout (b). Crossed neurons have been dropped out. Image from Srivastava et al. (2014).

Chapter 3

Related Work

In this chapter, we summarise work from ML for molecular structure processing and reaction modelling.

3.1 Molecular Structure Generation

A molecule's geometry is the three-dimensional configuration of its atoms. As such, the geometry has a strong effect on the molecule's properties. Classically, we describe geometries as conformations which are the three-dimensional coordinates of the molecule's atoms, although other representations such as spherical coordinates discussed in Section 2.2.3 can help preserve molecular structure better. Traditional methods for reaction modelling and conformation generation (Section 2.1.2), such as molecular dynamics simulations, often involve expensive energy calculations. ML methods can scan molecular representations to produce appropriate initial guesses for geometries using distance-geometry based features, incurring smaller computational cost. Here, we outline research taking in 2D graphs as input molecular representations to output geometries.

Before 2019, molecular structure prediction was geared more towards predicting stable protein folding dynamics but the development of generative GNN models have provided an avenue to focus instead on predicting distributions over stable molecular conformations. Simm & Hernandez-Lobato (2020) use a combination of dense layers and message passing NNs wrapped in a conditional variational autoencoder[1] to generate distributions over interatomic distances given a 2D molecular graph. The interatomic distance matrix (IDM) is a representation that is rotationally- and translationally-invariant producing a more robust model than previous work which generated distributions over coordinates. This model is able to predict molecules with complex structures such as rings which are common in drug-like molecules.

Turning to prediction for less distinct structures, Pattanaik, Ingraham, Grambow

[1]Variational autoencoders (VAEs) are autoencoders that produce a distribution over the embedding space. Variational methods are another form of approximate Bayesian inference and can be seen compared to deep ensembles in Figure 2.6. Conditional VAEs model a distribution based on some random variable - in this case, a molecular graph.

& Green (2020) predicted the geometries of transition states by combining a GNN for 2D molecular graphs with a differentiable multi-dimensional scaling procedure to produce refined geometries. This work was published with an associated dataset (Grambow et al. 2020) which together provide the overall task for our project. We revise the problem formulation through the lens of representation learning and uncertainty quantification in Section 4.2. Related to this, Lemm et al. (2021) created a graph-to-structure model to transform molecular graphs to distance matrices. Notably, this model uses kernel ridge regression instead of more involved deep learning methods. Their model is flexible over many structures, showing good performance on molecules without well-defined Lewis structures like TSs and carbenes.

3.2 Autoencoding 3D Molecular Graphs

While the previous section described work that took 2D molecular graphs as input to output 3D molecular graphs, this section describes the case where both the input and output are 3D graphs. Unlike the physics-inspired 3D GNNs described in Section 2.2.3, these models do not incorporate equivariant processing, however, they draw more on graph representation learning (Section 2.2.1) to learn structure-preserving 3D embeddings based on rotationally- and translationally-equivariant features.

Winter et al. (2020) developed an autoencoder to process 3D molecular graphs, independent of the number of atoms or bonds in a molecule. They use a special representation for the molecule called a Z-matrix which is the set of bond lengths, bond angles, and torsion angles. They use graph attention (GAT) layers with edge-conditioned convolution (EConv) layers to incorporate bond information between atoms into the processing. By composing one EConv layer with multiple GAT layers, atoms can express information about higher-order neighbours.

Nesterov et al. (2020) developed a variational autoencoder to encode molecules into a smooth embedding space called *3DMolNet*. Instead of using Z-matrices, their model processes a representation consisting of a nuclear charge matrix, IDM, and bond matrix. Where Winter and colleagues encode and decode each component of their representation separately, Nesterov and colleagues encode the representation as one before decoding each component separately. 3DMolNet also adds a coordinate-based penalty to the probabilistic loss function to better reconstruct the IDM. The authors use a multidimensional scaling procedure for the coordinate refinement in a similar manner to Pattanaik, Ingraham, Grambow & Green (2020).

3.3 Uncertainty-Quantified ML for CASP

Reaction templates are classes that group together reactions with similar properties. Computational reaction prediction methods can be split into template-based and template-free methods depending on whether they do or do not use hand-crafted templates. In recent years, ML has been used predominantly for template-free methods, categorised further into sequence- or graph-based methods. The state-of-the-art model in sequence-based methods is the Molecular Transformer, which uses an encoder-decoder model based around recurrent neural networks[2] and attention layers[3] on SMILES (text-based) data of reactions (Schwaller et al. 2019). The Molecular Transformer was able to predict a range of organic reactions covering aspects of chemoselectivity, steroselectivity, and regioselectivity. Beyond comparison with traditional quantum chemistry methods and human chemists, this model also used BMA techniques to produce uncertainty estimates.

Schwaller et al. (2021) continued their work investigating language models for chemical reactivity in a recent *Nature* paper. They found that the transformer models could learn fingerprints for reaction classes that performed better on downstream tasks than traditional hand-crafted fingerprints. The attention layers were used to describe learning dynamics which discovered and magnified subtle differences in reaction classes for clearer differentiation between reactions. They created a reaction atlas to map out the space encoded by their learned fingerprints. Although we aim to focus on 3D space and simpler reaction trajectories, this work provided clear inspiration for parameterising the reaction space with learned representations of data.

Jorner, Brinck, Norrby & Buttar (2021) developed a hybrid ML-mechanistic modelling approach for prediction of activation energies. They assessed their proof-of-concept model on the nucleophilic aromatic substitution (S_NAr) reaction - a widely-used reaction for commercial drugs. The hybrid model consists of a Gaussian process regression model trained on extracted mechanistic features of the reactant, product, and TS of a reaction, and is compared across different subsets of features and common reaction fingerprints. The authors find that in the common low-data setting of 50-150 datapoints, hybrid models perform better than either purely mechanistic or ML models, suggesting model selection across a low-data spectrum. They note that widespread use of hybrid models can be realised by better prediction of TSs for enhanced feature selection.

[2] Recurrent neural networks are a class of neural networks used for processing data ordered in a sequence like text e.g. in medicine to analyse medical notes.

[3] Attention layers are used to pinpoint important parts of a data structure. In a sequence, they are used to find important sections of the sequence.

Chapter 4

Our Approach

4.1 Data

Two quantitative reaction modelling datasets have emerged over the past two years (Grambow et al. 2020, von Rudorff et al. 2020). We use the dataset published by Grambow et al. (2020) which describes reactant, product, and TS geometries for 16,000 small organic reactions. The reactants in this dataset are sampled from GDB-7, a subset of a larger quantum chemistry dataset called GB-1723 (Ruddigkeit et al. 2012).

All of the reactions in this dataset have unimolecular reactants, but some also have multi-molecular products. Since we are trying to demonstrate a proof-of-concept framework with our work, we take a simpler subset of reactions that have unimolecular products. We use the same subset and splits developed by the authors of the dataset in a subsequent ML-focussed paper (Pattanaik, Ingraham, Grambow & Green 2020), and refer to this as the MIT Dataset (MITD). The MITD contains gas-phase reactions with a maximum of seven select heavy atoms (carbon, oxygen, and nitrogen) in each molecule and reflects the reaction diversity of the full dataset. The MITD is characterised by a broad array of reaction templates[1] which vary across the types of and quantity of bond changes.

4.2 Building on the MIT Model ts_gen

Pattanaik, Ingraham, Grambow & Green (2020) (the MIT authors) created a model called ts_gen to transform 2D reactant and product graphs into a TS geometry. We refer to this model interchangeably as the "MIT model" or ts_gen throughout this project. ts_gen is composed of three main steps, as shown in Figure 4.1: an initial data processing procedure followed by the two-step graph-to-coordinates (G2C) algorithm composed of a GNN followed by a non-linear weighted least squares (NLWLS) procedure. The data processing averages 2D graphs of reactants and products and then feeds this into the G2C algorithm to create a TS geometry.

[1] We can group reactions together in a reaction template (i.e. a class of reactions). The general reaction templates here are based on connectivity of atoms in the reaction centre, atom identity, charge, aromaticity, and bond type.

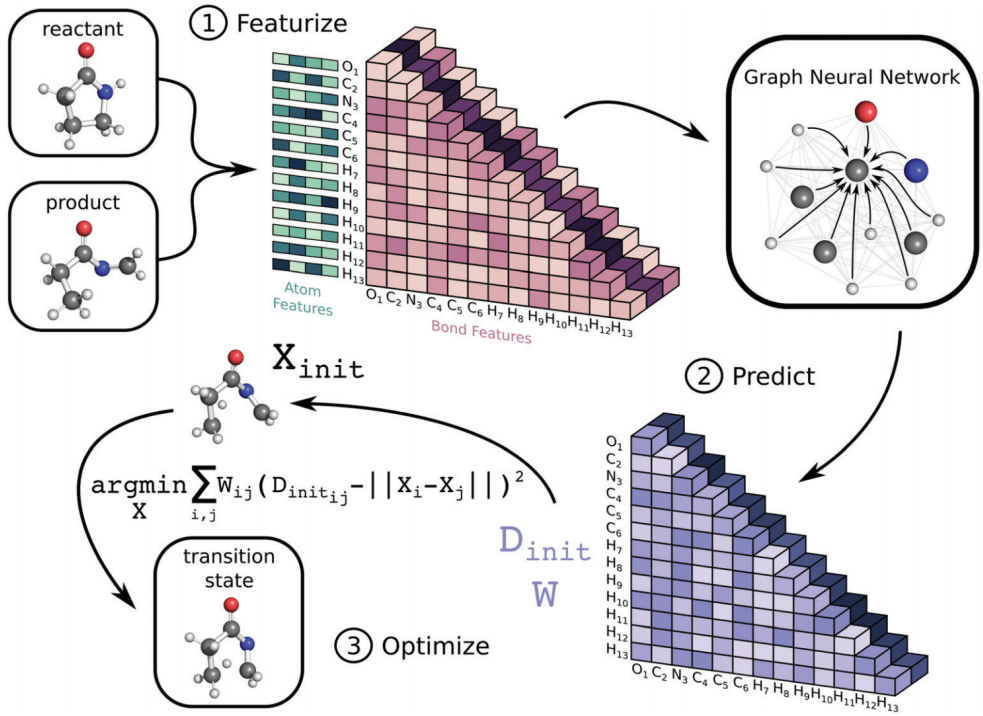


Figure 4.1: Visualised workflow of ts_gen. (1) Given 3D reactants and products, they are converted to 2D molecular graphs and averaged to get an initial 2D TS graph. (2) The TS graph is updated by a GNN and then passed through a standard dense layer to get a distance matrix D_{init} and corresponding weight matrix W. (3) The D_{init} and W are passed to a non-linear weighted least squares algorithm to produce TS geometry coordinates X which are converted to a final TS distance matrix D_{fin}. Steps (2) and (3) make up the G2C algorithm. Image from Pattanaik, Ingraham, Grambow & Green (2020).

The GNN of the G2C algorithm outputs an initial distance matrix D_{init} and corresponding weight matrix W. The D_{init} matrix here represents an initial guess for the TS geometry. The D_{init} and W are then fed into the NLWLS procedure where the W is used to refine the D_{init} matrix to produce the final TS geometry D_{fin}.

To the best of our knowledge, this is the first machine learning study investigating transition state geometry generation. Given their problem formulation, we propose two main ways to build on their algorithm so that it is more robust for downstream tasks such as reaction barrier prediction: performing uncertainty quantification and working directly in coordinate space. We summarise these ideas in a TS generation framework based on 3D GRL.

4.2.1 Uncertainty Quantification

The GNN in the G2C algorithm outputs a weight matrix W which is used in the subsequent NLWLS procedure. These weights act as a measure for how much the NLWLS algorithm should care about different parts of the D_{init}. The W and

D_{init} matrix are the same dimensions and so each time the G2C algorithm learns an element of the initial distance matrix $D_{init}[i, j]$, it also learns a corresponding weight $W[i, j]$. This is similar to learning uncertainty in the D_{init} matrix for each interatomic dimension. To better understand this property of the model, we can stability test ts_gen with Bayesian model averaging methods described in Section 2.3.2 to analyse the impact of the W matrix and reliability of predictions. As a preliminary for this task, we perform an ablation study[2] on ts_gen to determine if any components of their algorithm contribute significantly to output variation.

4.2.2 Working directly in 3D space

ts_gen has a pre-processing step to average fully-connected 2D graphs for the reactant and product to get a fully-connected 2D graph for the TS to feed into the G2C algorithm. The two-step G2C procedure allows the MIT model to learn based on the output of both the GNN (the D_{init} and W) and NLWLS (the D_{fin}). Although this approach is effective in producing similar distributions to the ground truth, the authors note that their model is unable to learn orientation constraints of TSs, which we hypothesise occurs due to the processing in 2D space rather than 3D space. There is also an element of redundancy in the model given that both parts of the two-step procedure output a version of the D - ideally, for a more interpretable model, we can build the geometry refinement in step two directly into step one by making a better estimate of D_{init}.

We aim to show that a one-step formulation based directly in 3D space has the potential to resolve both these issues. While we discuss autoencoders for 3D molecules in Section 3.2, we hypothesise that using the 3D GNNs described in Section 2.2.3 that process interatomic distances will deal with positional- and orientation-based issues more robustly through SE(3)-equivariant processing. If successful, our output TSs will be closer to the ground truth removing the need for the NLWLS procedure and making our approach one-step.

4.2.3 TS Generation as a 3D GRL Problem: TSED

Inspired by the general goal of identifying optimal reaction fingerprints and visualisation of reaction space from text-based representations (Sections 2.1.3 and 3.3), we express the MIT problem formulation as a 3D GRL problem. We want to learn 3D-based reaction fingerprints that can help us interpolate through reaction space and better reconstruct the TS geometries. We create a framework called transition state encoder-decoder (TSED) made up of an encoder, a combination

[2] An ablation study tests an architecture when certain components are removed to see how much each component contributes to a prediction. It acts as an approximate method to determine causality in a model.

function, and a decoder. As mentioned in Section 2.2.1, encoder-decoder models allow us to learn representations of initial data which can serve as reaction fingerprints.

The encoder, given by a 3D GNN, is used to learn representations of reactants and products and follows the interatomic distance-based message passing scheme described in Section 2.2.3. We combine the reactant and product representations using a combination function to get a TS representation, and then decode this representation to a distance matrix for the TS. Repurposing the 3D GNNs in Section 2.2.3 as encoders, we can create and compare two different implementations of our framework.

3D GNNs produce complex output distributions based on consecutive stochastic computations. We can compute these distributions tractably using the BMA procedures described in Section 2.3.2 to create distance matrices that express a variety of model instances. This complements related work in UQ for CASP (Section 3.3). Although the MIT model uses the input data in a different format, we can also compare uncertainty here to determine how Bayesian marginalisation can improve model calibration across two similar limited data settings.

We identify two tasks for our framework:

1. **Model Performance**. Do our embeddings decode well to the 3D molecular structure of molecules? Are our predictions stable across multiple experiments?

2. **TS Interpolation**. Can we interpolate between reactants and products in reaction space?

4.3 Evaluation Strategies

The goal of this work is to generate transition state geometries given 3D graphs of reactants and products. We aim for comparable performance with the MIT model with uncertainty estimates and clear TS on reaction core distributions with uncertainty estimates and show clear TS interpolation in embedding space between reactants and products.

Loss function. Both the MIT model and TSED framework generate predicted distance matrices D_{pred}. We compare these matrices to the ground truth D_{GT} using the root mean squared error, given as $RMSE = \frac{1}{N^2}\sum_{ij}(D_{pred}[i,j] - D_{GT}[i,j])^2$. We note that we experimented with training on reactant and product ground truth distance matrices too, but found the sole TS to perform better for our task.

Interatomic distance distributions. The reaction core is the set of atoms that change positions and bonds that break or form during the reaction from reactant to product. We adopt the focus of distributions over interatomic distances emphasised in Pattanaik, Ingraham, Grambow & Green (2020) and Simm & Hernandez-Lobato (2020) - for each reaction, we plot differences in interatomic distances of the reaction core as a distribution against the ground truth. We are not expecting any changes to atom positions for atoms outside the reaction core and this should be prevented by our loss function which considers all atoms. However, we note that this is a limitation of our visualisation strategy in an effort to make figures clearer.

Uncertainty calibration with reliability diagrams. For our task, we not only want to predict the interatomic distances, but also the associated probability of these predictions. The probability gives us a measure of uncertainty/confidence in our prediction. To assess distribution uncertainty, we create reliability diagrams. We follow the method used by Lamb & Paige (2020) to generalise reliability diagrams to the regression setting. This method looks at confidence intervals (CI) around the Bayesian predictive mean. Given a maximum size for the CI, we can plot the proportion of predictions in our test set that fall within each CI minus the size of the CI against the size of the CI.

Uncertainty estimation with box plots. Elements of the weight matrix act as a proxy for describing uncertainty in interatomic dimensions of molecules. Since we do not have ground truth values for the weights (since they are dependent on the distance matrix), we compare how our weights are distributed for atoms based on their interatomic topological distance (i.e. the number of bonds between them). Following the lead of Pattanaik, Ingraham, Grambow & Green (2020), we bin the different topological distances and observe error bars in the resulting boxplots as another uncertainty measure.

Chapter 5

Experiments and Results

5.1 Stability Testing the MIT Model

We perform two experiments here to investigate the stability of the MIT model. First, we conduct an ablation study to determine how the different model components contribute to the output. Second, we perform an UQ study using our two BMA methods comparing deep ensembles of the standard ts_gen model against the model with dropout. For both of these tasks we use the same 0.889 training split as the MIT model to get 6739 training and 842 test reactions. Where necessary, we describe each networks' hyperparameters as batch size b, hidden layer size h, number of GNN iterations g, and number of hidden layers l. Unless stated otherwise, we train all networks for 100 epochs using the Adam optimiser.

5.1.1 Ablation Study

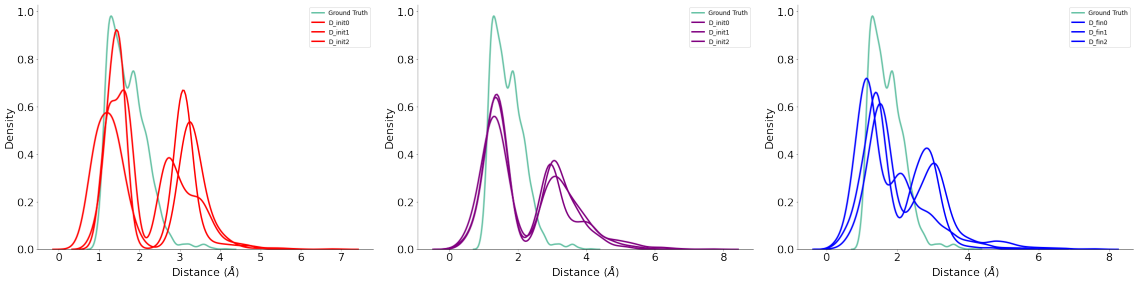
Framework and Evaluation

Although the G2C algorithm has two main steps, we can break it up further into three sequential stages. In order of processing, the GNN and distance/weight predictions (the $init$ stage), the power law iteration to initialise coordinates (the pow stage), and the optimisation loop to refine the coordinates as part of the non-linear weighted least squares algorithm (the fin stage). We evaluate the model for each of these three ablation stages to get a D matrix given as D_{init}, D_{pow}, and D_{fin} respectively. We train 3 networks for each stage using the same 3 hyperparameter configurations each time. The D_{fin} has an additional hyperparameter for the length of the optimisation loop which we set to 20 steps. We plot 3 distributions for each ablation against the ground truth distribution.

Results

Results are shown in Figure 5.1. Although bimodality is present in all distributions, each progressive stage produces a distribution closer to the (almost) unimodal ground truth. The left-most mode in the D_{init} distributions are closer to the ground truth than later stage distributions, suggesting that the optimisation loop

smoothes out interatomic distances for better overall performance at the expense of strong early guesses. The distributions start off with high uncertainty at the $init$ stage, coalesce around the coordinate initialisation in pow, and then expand outwards again slightly for the final stage fin. The minimal variation in distributions about the pow stage suggests the power iteration to initialise coordinates is a stable process. However, there is sufficient evidence to suggest uncertainty arises from the initial GNN and final optimisation loop.



(a) D_{init} distribution - no power iteration or optimisation loop.

(b) D_{pow} distribution - no optimisation loop.

(c) D_{fin} distribution - full G2C algorithm.

Figure 5.1: Three reaction core distance distributions for each of the G2C's sequential processing steps. Left is D_{init}, middle is D_{pow}, right is D_{fin}.

5.1.2 Uncertainty Quantification

Framework and Evaluation

Using 8 random hyperparameter initialisations, we train 8 networks of the standard ts_gen model. We average these distributions in two ways. Firstly, we average over each individual interatomic distance prediction to get a standard average. We then average the predictive distributions to get an ensemble. If these two averages do not match up well, it indicates that predictions for individual elements are more uncertain than they appear from individual networks, giving an indication of network stability. We repeat this formulation with a dropout version of ts_gen training 8 networks on the same 8 hyperparameter seeds. Specifically, we place dropout layers with a probability equal to 0.5 over each GNN layer and the final dense layer used to create the D_{init} (i.e. all the layers involved in Step (2) of Figure 4.1).

While plotting the distance distributions gives us a qualitative measure of uncertainty, we use reliability diagrams as a quantitative measure. Looking at the ground truth distribution, we assume an underlying Gaussian likelihood. As described in Section 4.3, we find the mean μ and standard deviation σ of our distribution and choose to investigate a maximum CI of $\mu + 2\sigma$. On the x-axis, we plot the CI size, and on the y-axis, we plot the proportion of distance predictions falling

within each CI, minus the CI size. We summarise overall reliability performance by calculating the miscalibration area under each curve.

Although there is no ground truth for the weight matrices W (since it is dependent on the D_{init}), we compare our standard and dropout ensemble Ws with the best MIT model's W to gauge weight uncertainty.

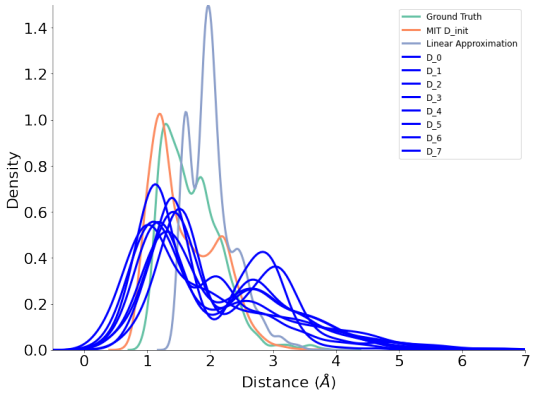
Results

Uncertainty across distances. The distribution plots are shown in Figure 5.2 against the distributions of ground truth TSs, the best D_{init} published by the MIT authors (which was trained with $\{e = 200, b = 8, h = 256, g = 3, l = 2\}$), and the linear approximation[1] of reactants and products. We observe uncertainty across both the standard and dropout ensembles with the dropout showing more variation amongst individual runs as expected, given that hyperparameters *and weights* have been modified. The average and ensemble distributions in both the standard or dropout models do not match up completely giving further indication for inherent model uncertainty. Our reliability diagrams and table of results are shown in Figure 5.3 and Table 5.1, respectively. Since the curves are above the perfect calibration line ($y = 0$), we find that Gaussian likelihoods induce underconfidence across our predictions.

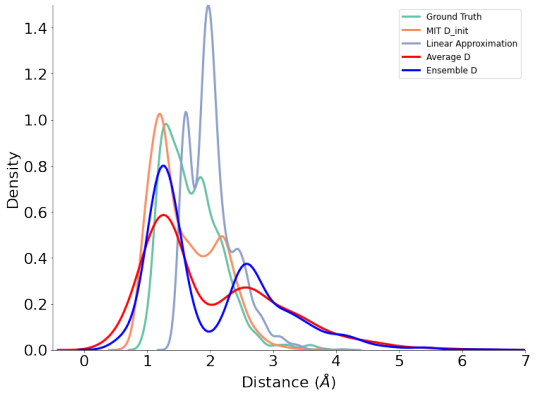
Table 5.1: Table showing the accuracy (as root mean squared error, RMSE) and miscalibration area (MA). We calculate the RMSE over the interatomic distance distributions (i.e. over every distance prediction), giving the accuracy as the mean and standard deviation of the resulting error distribution. Accuracy is calculated as the mean and standard deviation of interatomic distance MAs are calculated with Gaussian likelihoods and multiplied by 100. HP ID = Hyperparameter Seed ID.

HP ID	Accuracy		Miscalibration Area	
	Standard	Dropout	Standard	Dropout
0	0.93 $\pm$0.80	1.88 $\pm$1.05	15.33	12.51
1	1.00 $\pm$0.91	1.91 $\pm$0.94	13.75	12.67
2	0.84 $\pm$0.65	1.68 $\pm$1.15	18.52	10.59
3	0.85 $\pm$0.63	0.93 $\pm$1.59	17.70	17.73
4	1.00 $\pm$0.86	1.65 $\pm$0.95	13.42	12.24
5	0.98 $\pm$0.74	1.78 $\pm$1.04	13.75	13.97
6	1.10 $\pm$0.93	1.59 $\pm$1.00	10.61	12.91
7	1.05 $\pm$0.85	1.83 $\pm$1.09	11.41	7.46
Ens	0.79 $\pm$0.49	1.16 $\pm$0.64	16.57	15.09

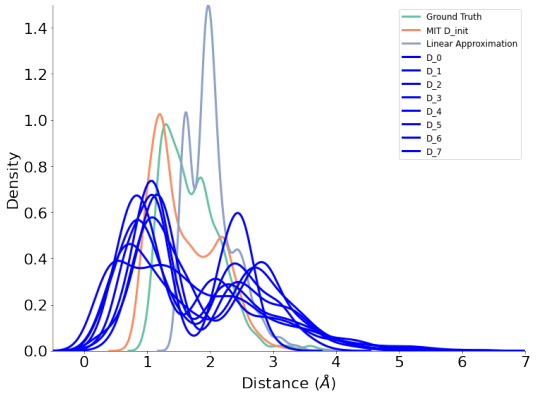
[1]The linear approximation/interpolation is the average of the coordinates between reactant and product geometries. This is the most basic baseline model.



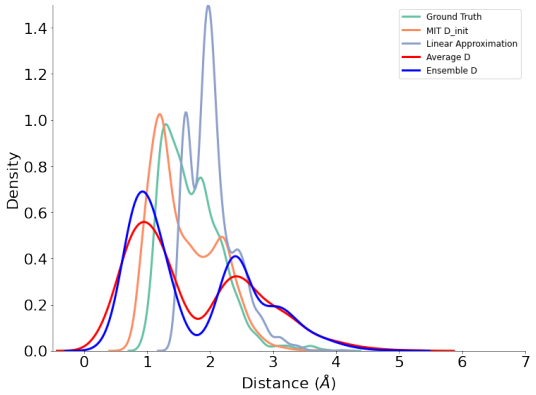
(a) 8 distributions from 8 individual runs of standard ts_gen model.



(b) Average and ensemble distributions of (a).



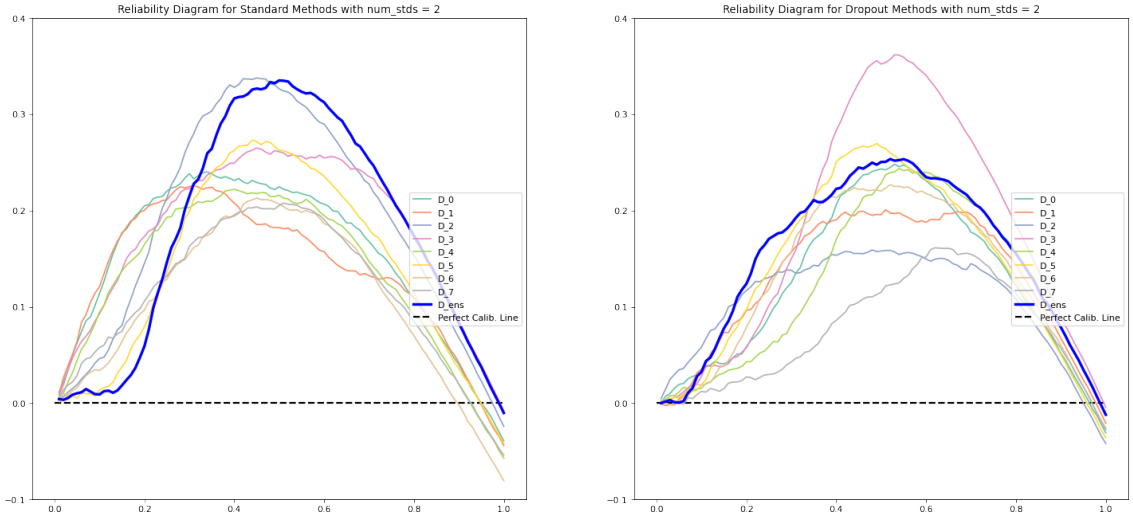
(c) 8 distributions from 8 individual runs of ts_gen with dropout with $p = 0.5$.



(d) Average and ensemble distributions of(c).

Figure 5.2: Interatomic distance distribution plots for reaction core. Top row is standard ts_gen model and bottom row is ts_gen with dropout probability of 0.5. The average distributions describe the literal average of the observed distributions while the ensemble distributions show the average over each individual interatomic predictions.

Uncertainty across weights. Our results are shown in Figure 5.4 and the error bars in Table 5.2. We normalise weights extracted from the network based on the maximum weight for each molecule and compare these weights based on the average topological distance (number of bonds between atoms the weight corresponds to) between reactant and product. The best MIT model in Figure 5.4 shows a clear downward trend indicating that the model has learned to preserve distances of atoms which are topologically close together when refining the initial distance prediction D_{init}. Although our ensembles in Figure 5.4b and 5.4c demonstrate a similar trend, there is noticeable variation. The standard ensemble in Figure 5.4b has a quick dropoff in weight priority after the first bin, with smaller quartiles and confidence intervals suggesting overconfidence in incorrect predictions. In the dropout ensemble in Figure 5.4c, the interquartile range reflects the MIT model better however the overall confidence intervals stay large for all bins indicating a high level of uncertainty.



(a) Ensemble of standard ts_gen model

(b) Ensemble of ts_gen with dropout $p = 0.5$

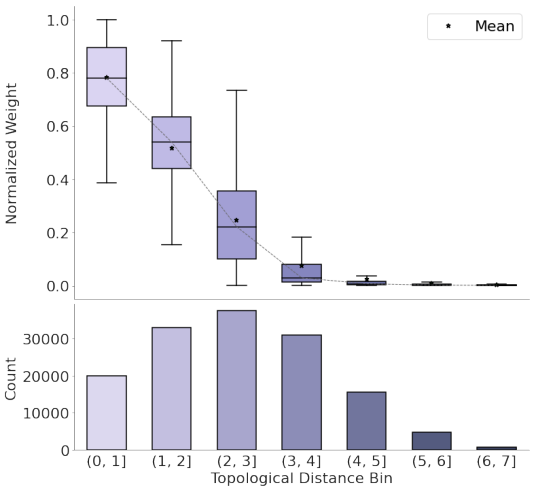
Figure 5.3: Reliability diagrams for 8 individual runs of the standard and dropout ts_gen model given Gaussian likelihoods.

Table 5.2: Table showing how the error bars of the weight plots in Figure 5.4 are distributed. TDB = Topological distance bin. DO = dropout.

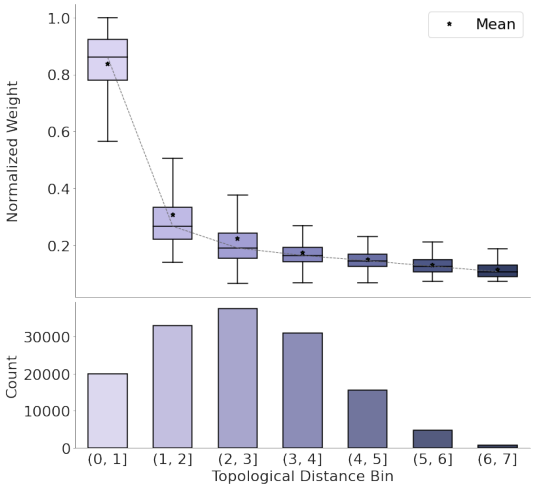
TDB	Median			Interquartile Range			95% CI Range		
	MIT	Ens	Ens DO	MIT	Ens	Ens DO	MIT	Ens	Ens DO
(0,1]	0.78	0.86	0.85	0.22	0.14	0.11	0.61	0.43	0.36
(1,2]	0.54	0.27	0.68	0.19	0.14	0.23	0.77	0.37	0.80
(2,3]	0.22	0.19	0.50	0.26	0.09	0.32	0.73	0.31	0.88
(3,4]	0.03	0.16	0.29	0.07	0.05	0.24	0.18	0.20	0.70
(4,5]	0.01	0.15	0.18	0.02	0.04	0.15	0.04	0.16	0.43
(5,6]	0.00	0.13	0.13	0.01	0.04	0.08	0.01	0.14	0.24
(6,7]	0.00	0.11	0.11	0.00	0.04	0.06	0.01	0.12	0.16

5.1.3 Discussion

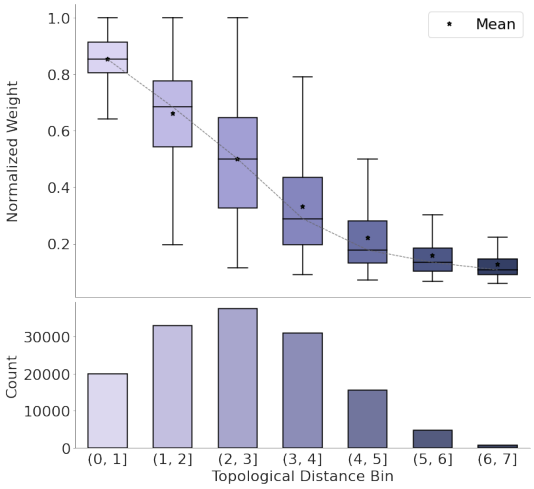
Having observed inconsistent results across ensemble comparisons for both the standard and dropout versions of ts_gen, we deduce that the distances D and weights W are undetermined by the data, pointing to uncertainty in coordinate space and weight space. Since uncertainty is given by diversity of samples, for every network there can be a range of plausible TSs. This motivates newer approaches to TS generation grounded in UQ. We discuss this in the following section as part of our TSED framework.



(a) MIT's Best Model



(b) Ensemble of standard ts_gen



(c) Ensemble of standard ts_gen with dropout $p = 0.5$

Figure 5.4: Box plots of normalised weight matrices binned by average topological distance (i.e. average number of bonds separating the atoms which the weight corresponds to) between reactant and product of test set reactions. We plot these diagrams for the weights given by the best MIT model, and our ensembles of the same 8 runs, respectively, of the standard ts_gen and ts_gen with dropout $p = 0.5$, shown in Figure 5.2.

5.2 TS Generation as a 3D GRL problem: TSED

Framework and Evaluation

We implement two versions of our TSED framework outlined in Section 4.2.3, one with SchNet as the encoder and the other with EGNN. We compare model performance based on uncertainty again, given by qualitative distribution plots. The models take different parameters but we train them both with a 3 hidden layers, again for 100 epochs using the Adam optimiser. Next we compare our learned embeddings for each reaction molecule to check if we can interpolate through reaction space.

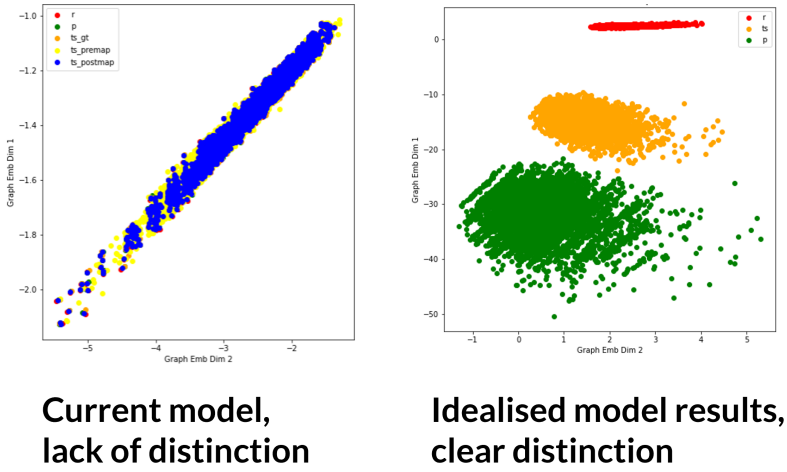
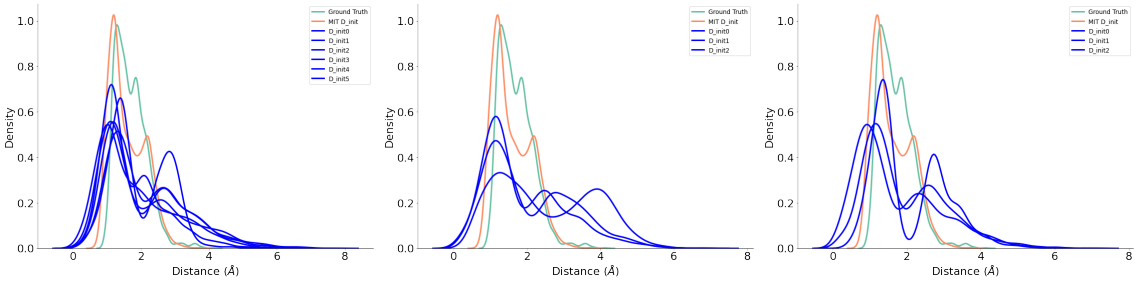


Figure 5.5: Current results for visualisation of learned reaction molecule embeddings using TSED-EGNN. We do not create clear embeddings which likely explains the performance drop when compared to the MIT model.

Results

Our distribution plots are shown in Figure 5.6. While our encoder-decoder framework is flexible, allowing different components for encoding and decoding, the produced D matrices do not reach the best D_{init} from the MIT model. Looking at our uncertainty estimates, our model is not as stable as the MIT model either. Particularly, for the SchNet implementation of TSED, the distributions become wider and flatter. We hypothesise that this is related to our poorer reaction representations reflected by the unclear reaction space interpolation. All the reactions are clustered together indicating minimal differences between representations resulting in less comparable reaction core distributions.

A related explanation could be the fact we have repurposed existing 3D GNNs into a framework rather than designing a dedicated architecture for TS generation. Attention has been used in high-performing text-based reaction models (Schwaller et al. 2019) and for 3D molecule autoencoders (Winter et al. 2020), and presents an opportunity for future work. Although we have not explored this in depth, we include an initial reaction heatmap idea for this in Figure 5.7 based off the MIT formulation where the initial data is an average between reactants and products. For future work, we plan to regularise our embeddings and plot distances between reaction molecules in an effort to map *through* embedding space.



(a) MIT Model (b) TSED-SchNet (c) TSED-EGNN

Figure 5.6: Interatomic distance distributions of reaction core using two implementations of the TSED framework. We include the MIT model as a reference for performance but note that our model uses a different format of the input data so this comparison is flawed.

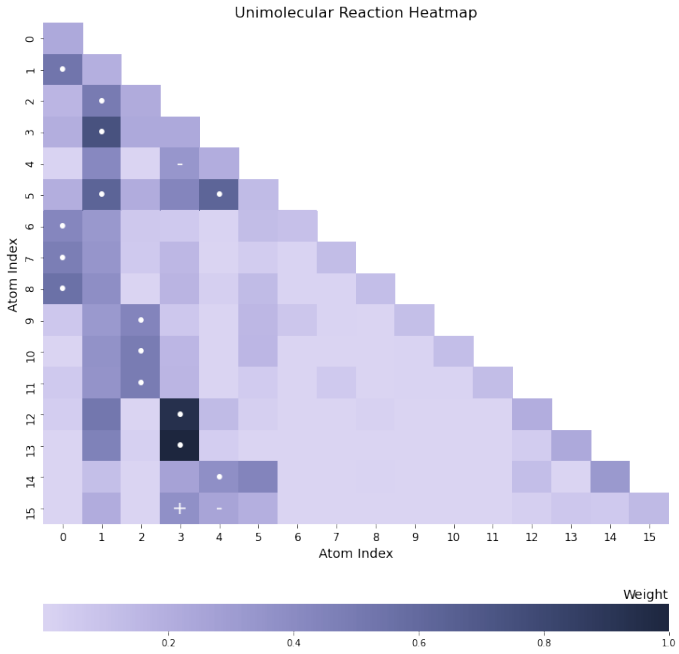


Figure 5.7: Unimolecular reaction heatmap for a reaction with 15 atoms. Describes how a reactant transforms into a product with labelled cells for atomic movements. Darker cells indicate that the corresponding atom is more important. Essentially, this is a more involved weight matrix.

Chapter 6

Conclusion

6.1 Study Limitations

The overarching aim for reaction modelling is a tight feedback loop between computational science and wet lab experimentation, driven by automation (Coley 2021). Methods accounting for engineering concerns like uncertainty calibration in low-data settings are the first steps towards this aim. Our framework is a proof-of-concept that TS generation can be framed as 3D GRL problem grounded in uncertainty estimation. However, there are several steps required to make this model usable for medicinal chemists, and we address these limitations in the following paragraphs.

Proxy data for drugs. We have a dataset containing molecules with fewer than 25 atoms to act as proxy data for drug molecules which contain greater than 100 atoms. Generally, our experiments on a dataset of 7600 reactions took an hour to run on a GPU for 100 epochs. Although scaling up to larger drug molecules would be feasible in computational time, it is insufficient for a quick computational modelling-wet lab iteration loop. In these scenarios with small datasets (<10,000 data points) and large molecules (>100 atoms), we believe transfer learning[1] or hybrid mechanistic-ML models (Jorner, Brinck, Norrby & Buttar 2021) will help implement these models in a way that improves the productivity of human chemists (Thakkar et al. 2021).

Complex reaction pathways. We do not investigate more complex reaction pathways containing multiple transition states or bimolecular reactions. Since platforms for multistep synthesis are still in preliminary phases (Coley et al. 2019), this data is not publicly available. We have used a unimolecular subset of our overall reaction dataset. During correspondence with an author of ts_gen, they mentioned extending their model for bimolecular reactions. We hope to integrate parts of our framework with their work in the future.

[1] Transfer learning is a subfield of machine learning that uses information learned from one problem in a similar but different problem.

6.2 Long-Term Future Work

Chirality-aware 3D GNNs. As mentioned in Section 3, previous ML in reaction modelling work has mapped out reaction space based on string representations (Schwaller et al. 2021). In this report, we have focussed on learning representations of 3D molecular graphs. But for a more chemically-aware model that better reflects the flexibility of molecular geometry, we need to consider chirality. Chirality is a property of chemical molecules that tells us even if two graphs are isomorphic (i.e. structurally the same), their 3D structures can be different based on ordering of carbon centre neighbours. Our 3D-based autoencoder framework accounts for geometry more than text-based frameworks but building on recent work in chirality-aware message passing would give us a more chemically-grounded representation (Pattanaik, Ganea, Coley, Jensen, Green & Coley 2020).

More comprehensive comparisons. Similar to Parsaeifard et al. (2021) who compared hand-crafted chemical representations, we could perform a comparison on learned representations from text, 3D, and chirality-aware processing on the same tasks. This would identify where each representation performs best, and how the different representations can complement each other in the overall DMTA cycle. While the representation learning models have outperformed other models in common text benchmarks (Schwaller et al. 2019), it remains to be shown whether this holds in 3D because of how recently the datasets have been published. For a broader comparison, we can compare our ML framework with traditional quantum-chemistry-principles-first reaction modelling methods outlined in Section 2.1.2.

Following the text-based path. Given its progress over the last three years, the state-of-the-art text-based representation learning model for reaction prediction, *Molecular Transformer*, presents a useful path for 3D representation work to follow. Starting off as uncertainty-calibrated reaction prediction, the model was adapted to new domains using transfer learning, and learned reaction fingerprints were used in downstream tasks to predict reaction yields and map reaction pathways across a broad chemical space (Schwaller 2021). The 3D problem is similar but more suited to fine-tuning certain chemical subspaces for specific tasks where orientation constraints are important. So far, we have shown that UQ can be integrated using Bayesian approaches. If we are able to improve the quality of our learned fingerprints, the next step would be using them for reaction barrier prediction. There is another dataset for atomistic reaction modelling which contains geometries across the chemical space of competing $E2$ and S_N2 reactions (von Rudorff et al. 2020). Competing reactions can have an impact on production yield

and span a large unique chemical space. For these reasons, we can take inspiration from the text-based work in similar domains and adapt our model to this dataset.

Bibliography

Atz, K., Grisoni, F. & Schneider, G. (2021), 'Geometric deep learning on molecular representations', *arXiv preprint arXiv:2107.12375* .

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R. et al. (2018), 'Relational inductive biases, deep learning, and graph networks', *arXiv preprint arXiv:1806.01261* .

Bender, A. & Cortes-Ciriano, I. (2020), 'Artificial intelligence in drug discovery: what is realistic, what are illusions? part 1: Ways to make an impact, and why we are not there yet', *Drug Discovery Today* .

Bender, A. & Cortes-Ciriano, I. (2021), 'Artificial intelligence in drug discovery: what is realistic, what are illusions? part 2: a discussion of chemical and biological data used for ai in drug discovery', *Drug Discovery Today* .

Bengio, Y., Courville, A. & Vincent, P. (2013), 'Representation learning: A review and new perspectives', *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1798–1828.

Bhoorasingh, P. L. & West, R. H. (2015), 'Transition state geometry prediction using molecular group contributions', *Physical Chemistry Chemical Physics* **17**(48), 32173–32182.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M. H. & Hernández-Lobato, J. M. (2020), 'Barking up the right tree: an approach to search over molecule synthesis dags', *Advances in Neural Information Processing Systems* **33**.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M. & Hernández-Lobato, J. M. (2019), 'A model to search for synthesizable molecules', *Advances in Neural Information Processing Systems* **32**.

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., Ferrero, E., Agapow, P.-M., Zietz, M., Hoffman, M. M. et al. (2018), 'Opportunities and obstacles for deep learning in biology and medicine', *Journal of The Royal Society Interface* **15**(141), 20170387.

Coley, C. W. (2021), 'Defining and exploring chemical spaces', *Trends in Chemistry* **3**(2), 133–145.

Coley, C. W., Thomas, D. A., Lummiss, J. A., Jaworski, J. N., Breen, C. P., Schultz,

V., Hart, T., Fishman, J. S., Rogers, L., Gao, H. et al. (2019), 'A robotic platform for flow synthesis of organic compounds informed by ai planning', *Science* **365**(6453).

Fletcher, R. (2013), *Practical methods of optimization*, John Wiley & Sons.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B. R., Vickers, R., Roberts, C., Tang, J., Roblin, D., Blundell, T. L., Bronstein, M. M. & Taylor-King, J. P. (2021), 'Utilizing graph machine learning within drug discovery and development', *Briefings in Bioinformatics* .

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. (2017), 'Neural message passing for quantum chemistry', *International Conference on Machine Learning* .

Grambow, C. A., Pattanaik, L. & Green, W. H. (2020), 'Reactants, products, and transition states of elementary chemical reactions based on quantum chemistry', *Scientific data* **7**(1), 1–8.

Hamilton, W. L. (2020), 'Graph representation learning', *Synthesis Lectures on Artifical Intelligence and Machine Learning* **14**(3), 1–159.

Hirschfeld, L., Swanson, K., Yang, K., Barzilay, R. & Coley, C. W. (2020), 'Uncertainty quantification using neural networks for molecular property prediction', *Journal of Chemical Information and Modeling* **60**(8), 3770–3780.

Jorner, K., Brinck, T., Norrby, P.-O. & Buttar, D. (2021), 'Machine learning meets mechanistic modelling for accurate prediction of experimental activation energies', *Chemical Science* **12**(3), 1163–1175.

Jorner, K., Tomberg, A., Bauer, C., Sköld, C. & Norrby, P.-O. (2021), 'Organic reactivity from mechanism to machine learning', *Nature Reviews Chemistry* **5**(4), 240–255.

Lakshminarayanan, B., Pritzel, A. & Blundell, C. (2017), 'Simple and scalable predictive uncertainty estimation using deep ensembles', *Advances in Neural Information Processing Systems* **30**.

Lamb, G. & Paige, B. (2020), 'Bayesian graph neural networks for molecular property prediction', *Machine Learning for Molecules Workshop at NeurIPS* .

Lemm, D., von Rudorff, G. F. & von Lilienfeld, O. A. (2021), 'Machine learning based energy-free structure predictions of molecules, transition states, and solids', *Nature Communications* **12**(1), 1–10.

Liu, Y., Wang, L., Liu, M., Zhang, X., Oztekin, B. & Ji, S. (2021), 'Spherical message passing for 3d graph networks', *arXiv preprint arXiv:2102.05013* .

Nesterov, V., Wieser, M. & Roth, V. (2020), '3dmolnet: a generative network for molecular structures', *Machine Learning for Molecules Workshop at NeurIPS* .

Parsaeifard, B., De, D. S., Christensen, A. S., Faber, F. A., Kocer, E., De, S., Behler, J., von Lilienfeld, O. A. & Goedecker, S. (2021), 'An assessment of the structural resolution of various fingerprints commonly used in machine learning', *Machine Learning: Science and Technology* **2**(1), 015018.

Pattanaik, L., Ganea, O.-E., Coley, I., Jensen, K. F., Green, W. H. & Coley, C. W. (2020), 'Message passing networks for molecules with tetrahedral chirality', *Machine Learning for Molecules Workshop at NeurIPS* .

Pattanaik, L., Ingraham, J. B., Grambow, C. A. & Green, W. H. (2020), 'Generating transition states of isomerization reactions with deep learning', *Physical Chemistry Chemical Physics* **22**(41), 23618–23626.

Ruddigkeit, L., Van Deursen, R., Blum, L. C. & Reymond, J.-L. (2012), 'Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17', *Journal of chemical information and modeling* **52**(11), 2864–2875.

Satorras, V. G., Hoogeboom, E. & Welling, M. (2021), 'E(n) equivariant graph neural networks', *International Conference on Machine Learning* .

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A. & Müller, K.-R. (2018), 'Schnet–a deep learning architecture for molecules and materials', *The Journal of Chemical Physics* **148**(24), 241722.

Schwaller, P. (2021), 'Learning the Language of Chemical Reactions–Atom by Atom. Linguistics-Inspired Machine Learning Methods for Chemical Reaction Tasks', PhD thesis, University of Bern.

Schwaller, P., Laino, T., Gaudin, T., Bolgar, P., Hunter, C. A., Bekas, C. & Lee, A. A. (2019), 'Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction', *ACS central science* **5**(9), 1572–1583.

Schwaller, P., Probst, D., Vaucher, A. C., Nair, V. H., Kreutter, D., Laino, T. & Reymond, J.-L. (2021), 'Mapping the space of chemical reactions using attention-based neural networks', *Nature Machine Intelligence* **3**(2), 144–152.

Simm, G. & Hernandez-Lobato, J. M. (2020), 'A generative model for molecular distance geometry', *International Conference on Machine Learning* .

Simm, G. N., Vaucher, A. C. & Reiher, M. (2018), 'Exploration of reaction path-

ways and chemical transformation networks', *The Journal of Physical Chemistry A* **123**(2), 385–399.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

Struble, T. J., Alvarez, J. C., Brown, S. P., Chytil, M., Cisar, J., DesJarlais, R. L., Engkvist, O., Frank, S. A., Greve, D. R., Griffin, D. J. et al. (2020), 'Current and future roles of artificial intelligence in medicinal chemistry synthesis', *Journal of medicinal chemistry* **63**(16), 8667–8682.

Thakkar, A., Johansson, S., Jorner, K., Buttar, D., Reymond, J.-L. & Engkvist, O. (2021), 'Artificial intelligence and automation in computer aided synthesis planning', *Reaction chemistry & engineering* **6**(1), 27–51.

Unsleber, J. P. & Reiher, M. (2020), 'The exploration of chemical reaction networks', *Annual review of physical chemistry* **71**, 121–142.

van der Wilk, M. (2019), 'Sparse Gaussian Process Approximations and Applications', PhD thesis, University of Cambridge.

von Rudorff, G. F., Heinen, S. N., Bragato, M. & von Lilienfeld, O. A. (2020), 'Thousands of reactants and transition states for competing e2 and s2 reactions', *Machine Learning: Science and Technology* **1**(4), 045026.

Wilson, A. G. & Izmailov, P. (2020), 'Bayesian deep learning and a probabilistic perspective of generalization', *Advances in Neural Information Processing Systems* **33**.

Winter, R., Noé, F. & Clevert, D.-A. (2020), 'Auto-encoding molecular conformations', *Machine Learning for Molecules Workshop at NeurIPS* .

Appendix A

Glossary

2D graph: a data structure with nodes and edges. Relationships between nodes are defined by graph connectivity and edges.

3D graph: a data structure with nodes, edges, and position vectors. Relationships between nodes are defined by graph connectivity, edge types, and relative position vectors of nodes.

Ablation study: removing components of an architecture to determine how each component contributes to the overall model.

Adjacency matrix: a square matrix with elements between 0 and 1 representing whether nodes in a graph are connected e.g. if A is an adjacency matrix for graph G, $A[i, j] = 1$ tells us that nodes i and j are connected while $A[i, j] = 0$ tells us they are not. Probabilistic adjacency matrices with elements between 0 and 1 give us the probability that two nodes are connected.

Approximate Bayesian inference: approximate methods for estimating joint posterior distributions over a computationally-intractable number of parameters. Often used on deep neural networks which have multiple layers each with their own set of parameters.

Attention: a deep learning concept to focus on important parts of input data during processing. Can be used to understand training dynamics.

Autoencoder: an encoder-decoder model that outputs a reconstruction of the input data.

Bayesian model averaging: a method for approximate Bayesian inference that builds a posterior distribution encoding different configurations of model parameters.

Bayesian modelling: another common approach to statistical modelling. Assumes data is fixed and model parameters are random variables. Learning process involves forming a distribution by marginalising over different weighted parameter configurations.

Belief: the flipside of uncertainty. Where uncertainty is how much we do not

know about our model, belief is how much we do know. And like uncertainty, belief is expressed as a probability distribution.

Bond lengths: the interatomic distances between all bonded atoms.

Computer-assisted synthesis planning: using computational modelling to help map out relevant chemical spaces for desired molecules and find reaction pathways to get to them.

Conformation: a spatial arrangement of atoms in a molecule, also called the molecular coordinates. Usually we talk about low-energy conformations as the set of stable positions associated with a molecule.

Dense layers: a neural network layer in which every neuron receives input from all neurons in the previous layer.

Density-functional theory: a modelling framework in the physical sciences based on electronic structure and used to describe molecules and other chemical systems.

Dimensionality (of data): number of parameters in the data.

Embed: AKA encode. Transform data from the feature space to the embedding space.

Embedding: representation of initial data with a different (often, smaller) dimensionality.

Embedding space: AKA the representation space. the space of parameters describing the representation.

Encoder-decoder: a model that encodes data to an embedding space and then decodes to a task-relevant prediction.

Equivariance: a property of a transformation to data. If transforming the input transforms the output in the same way, the transformation is equivariant.

Feature: a property of a component in a data structure. For example, a node is a component of a graph data structure. If the node has properties such as its element type and whether it is a heavy atom, these properties are considered node features.

Feature space: the parameter space of all possible values for the current set of features. This could be at the stage of input (i.e. input space), embedding (i.e. embedding space), or at any point during the processing (i.e. feature space).

Fingerprint: the name used to describe different representations of molecules or reactions (i.e. molecular fingerprints or reaction fingerprints). Usually used to

describe vector representations over text, 2D graph, or 3D graph representations but all are possible.

Frequentist modelling: the classical approach to statistical modelling. Assumes parameters of the model are fixed and data is a random variable. Learning process usually involves maximising a posterior.

Ground truth values: the values/labels (i.e. prediction labels) for raw data. Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, x are the raw data and y are the labels. When we used the ground truth values for training, we say that the ML task is *supervised*.

Group theory: a field of mathematics studying algebraic structures called groups. Groups are a convenient way of understanding symmetries which we briefly cover when discussing equivariances.

Hand-crafted feature: a data parameter that has been manually created by a human rather than learned by an algorithm during processing.

Inductive bias: constraints on a model to restrict the solution space to realistic values.

Interatomic distances: the distances between atoms. More general than bond lengths as describes all pairwise atomic distances, regardless of bonding.

Invariance: a property of a transformation to data. If transforming the input does not affect the output, the transformation is invariant. This is a special case of equivariance.

Marginalisation: generating a distribution based of a subset of parameters from an overall distribution. Marginal parameters are the subset of parameters that are retained in the new distribution.

Maximum likelihood estimation: the standard learning process in statistics that learns model parameters which maximise a likelihood function for data.

Maximum a posteriori: the traditionally-considered-Bayesian method for estimating model parameters. We adopt the view from Wilson & Izmailov (2020) that this is not Bayesian since it is not based on marginalisation.

Message passing (over graphs): the process by which nodes send vector messages to other nodes which are then updated using a transformation like a neural network.

Molecular dynamics: simulations of molecular activity, usually based on quantum chemistry-based energy calculations. This is the standard method for a lot of molecular science.

Molecular graph: a 2D or 3D graph describing a molecule. Nodes represent atoms, edges represent bonds. The 3D molecular graph additionally has positional features for atoms.

Optimisation: in complex feature spaces, optimisation can be used to find minima or maxima of multidimensional curves. We characterise multidimensional curves in reaction space that correspond reaction pathways.

Reaction coordinate: an point in reaction space, and usually on a reaction pathway, describing a stage of a reaction.

Reaction core: the set of atoms that change positions and bonds that break or form during the reaction from reactant to product.

Reaction modelling: computational modelling to better understand aspects of chemical reactions. A slightly broader concept than computer-assisted synthesis planning but used interchangeably in this project.

Reaction pathways: a curve through reaction space from reactants to products.

Reaction space: the space of parameters relevant to the reaction.

Reaction template: a class of reactions with similar properties e.g. different reactions with similar bonds broken and formed can be grouped together under a reaction template.

Recurrent neural network: a class of neural networks used for processing sequential data (i.e. where relationships between nodes can form a directed graph) like text.

Relational inductive bias: model constraints based on the relational structure of data i.e. how different components of the data relate to each other.

Representation (of data): an alternative format to express data such that the original structure of the data is preserved.

SE(3) Group: the Special Euclidean Group describing rotations and translations in Euclidean space.

SMILES: strings (i.e. text) which are used in computational chemistry to represent molecules or reactions.

Spherical coordinates: an alternative coordinate system to Cartesian coordinates which can be expressed as SE(3)-equivariant features.

Topological distance: the number of bonds between two atoms. This is an alternative measure of molecular distance to interatomic distances.

Transition state: a molecule (or set of molecules) that occurs as reactants transform into products in a reaction.

Uncertainty quantification: a set of practices for estimating the reliability of outputs. Particularly useful in low- or biased-data settings.

Variational autoencoder: autoencoders that produce a distribution over the embedding space which can be sampled from. Variational methods are another form of approximate Bayesian inference and can be seen compared to deep ensembles in Figure 2.6.