A generative graph machine learning method for transition state sampling in molecular systems

by

Saffat Bokul

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
July 2023

Thesis advisor

Dr. Lorenzo Livi

Author

Saffat Bokul

A generative graph machine learning method for transition state sampling in molecular systems

Abstract

Transition states (TS) are important molecular conformations with applications in drug development, materials science and biology. The transient nature of these conformations makes studying them extremely challenging. Simulating these states requires expensive computational algorithms whose convergence is never guaranteed. Machine learning (ML) has led to significant improvements in various domains, including computational chemistry. In this thesis, we propose a novel ML pipeline for sampling TS in molecular systems. We combine the expressive capability of graph neural networks to encode graph representations of molecular conformations which is then used for training, with generative models that are able to predict TS structure as output. Using our proposed method, we successfully demonstrate the sampling of minimum energy pathways and conformational TS in alanine dipeptide in one-shot without the need for any example TS or reaction coordinates. Our unsupervised novel ML approach shows promise in TS search and could lead the path for further innovations and analysis of more complex chemical systems.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Dr. Lorenzo Livi for providing me the opportunity to pursue my master's degree in computer science and for continually providing me with guidance, expertise and support during the degree. I am grateful to him for his patience, understanding, and belief in my abilities, without which this research would not have been possible.

I would also like to thank our collaborators in the group led by Dr. Vittorio Limongelli at The Università della Svizzera italiana, Switzerland for their expertise in computational chemistry and molecular simulations.

I am grateful to have had the amazing opportunity to work and learn alongside many talented individuals, which has had a profound impact on my overall development as a researcher and individual.

I dedicate this thesis to my loving parents and my sister for the countless sacrifices they have made to support me on this journey. Thank you for having faith in me and providing me with the strength to persevere through difficult times.

Chapter 1

Introduction

With the evolution of computational resources such as graphical processing units (GPUs) [1], availability of large datasets [2], and ML architectures [3, 4] ML has been successfully used in fields like computer vision [5] and natural language processing [6]. Today, ML techniques are increasingly being used in computational chemistry to address problems such as drug discovery and development [7], protein structure prediction [8], chemical property prediction [9], etc.

Scientists in the fields of chemistry, physics, materials science, pharmacology, and related disciplines have been trying to expedite the investigation of energy surfaces pertaining to various molecular systems. An area of particular significance entails identifying the molecular conformation that corresponds to a transition state (TS) or saddle point of a reaction [10] or events such as protein folding or unfolding [11]. The molecular conformation of a TS is desired since it could provide insight into a chemical reaction [12], or provide lead compounds for drug design [13]. For example, TS analogues can bind to and inhibit an enzyme [14]. These analogues have

the ability to emulate the structure of the TS, resulting in significantly stronger binding than substrates. This offers a tremendous potential in drug design and development for multiple targets. TS structures can also guide the optimization of reactions by enabling the design of catalysts, and development of new reaction pathways. However, searching for TS structures is often carried out using computationally intensive and non-scalable quantum mechanics-based methodologies that rely on *ab initio* calculations [15, 16, 17]. When dealing with real-world reactions involving large molecules and intricate interactions, mapping the multidimensional free energy surface (FES) becomes a formidable challenge due to the involvement of numerous degrees of freedom. Simulation-based techniques employed in this endeavor often demand considerable time, ranging from days to even months [18]. Gathering the required data necessitates the observation of numerous transitions; however, the rarity of these events poses a significant challenge. Furthermore, these simulations can become trapped in intermediate metastable states that are suboptimal [19]. Transition states also occur at extremely low concentration, which makes studying them in isolation very challenging. They can also be sensitive to environmental factors such as temperature, pressure and solvent effects making the problem even more dynamic. Even with all of the computational techniques, there is no guarantee of convergence. In brief, finding the TS on the FES is anything but a trivial task.

To address this challenge, in this research we propose a machine learning framework to generate molecular conformations corresponding to the TS structure. The proposed methodology is based on the combination of graph neural networks and the normalizing flow based Boltzmann Generator [20]. Using the Boltzmann generator,

we sample minimum energy pathways (MEP) in a chemical system. We then analyse the structures of the various conformations along this MEP to find candidate TS structures. To this end, we propose a novel graph autoencoder that allows us to bi-directionally map molecular structures to latent representations that can be processed by the boltzmann generator. We evaluate the results produced by the proposed methodology on a benchmark system, the Alanine Dipeptide, for which information about the FES and TS is readily available. We provide an end-to-end framework, starting from molecular representation to generating and validating TS conformations. Our streamlined process is fast and can sample MEPs and TSs from a fully trained pipeline. We can also generate many viable candidate TS in one-shot. We demonstrate the applicability of our approach by successfully sampling and validating a conformational TS in the alanine dipeptide system. Our proposed methodology is general enough to be applicable to virtually all molecular systems with the only requirement being the ability to evaluate the free energy related to generated configurations and having access to structural data characterizing the various conformations. Nonetheless, our proposed method represents a powerful and exciting approach to addressing a long-standing challenge in computational chemistry (and other fields) and could open the door for new developments.

1.1 Thesis Contribution

Our contribution in this thesis can be summarized as follows:

1. We propose a novel graph machine learning based autoencoder for the purpose of efficient and effective embedding generation and reconstruction of molecular

graphs.

2. We propose an ML pipeline for learning the underlying FES distribution of any chemical system from MD simulation data for sampling and exploration of both equilibrium and high energy configurations. The pipeline works in a completely unsupervised manner, without the need for an example TS or reaction coordinate.

3. We propose a workflow for generating viable minimum energy pathways for any molecular system using ML with data from MD simulation.

4. We demonstrate our novel TS sampling workflow for the sampling of a conformational TS in the alanine dipeptide system in one-shot. The generated TS has been validated with rigorous methods and therefore is realistic.

The details of our methodological contribution are provided in chapter 4 and the experimental results are discussed in chapter 5.

1.2 Thesis Organization

This thesis is organized in the following way:

1. **Chapter 2** provides the necessary background information on machine learning, deep generative models, graph neural networks and molecular dynamics and related techniques which are required to understand the work done in this thesis.

2. **Chapter 3** gives a literature review, divided into different sections.

3. **Chapter 4** discusses the proposed methodology in detail.

4. **Chapter 5** describes the dataset, discusses the experimental results and demonstrates the TS sampling workflow, selection and validation of TS candidates.

5. **Chapter 6** concludes the thesis with ideas on future directions.

Chapter 2

Background

In this chapter, we will discuss some of the necessary background information required to understand the concepts in this thesis. In the section 2.1, we discuss some basic concepts related to machine learning, in section 2.2 we discuss generative modeling in ML, in section 2.3 we discuss the core concepts of graph neural networks and finally in section 2.4 we discuss some basic ideas related to molecular dynamics.

2.1 Introduction to Machine Learning

In this section, we provide a general overview of Machine Learning (ML), deep learning and related techniques. The topics covered here are limited to the techniques used in this thesis. It starts with a general idea about Deep Learning (DL), covering what deep generative networks are and then diving into specific techniques like Normalizing flows and Autoencoders.

Deep learning is a sub-field of machine learning that focuses on training artificial

neural networks to acquire knowledge and generate predictions on complex data [21]. It represents an approach to artificial intelligence (AI) characterized by models that exhibit a greater degree of composition of learned functions or concepts compared to conventional machine learning methods. The popularity of deep learning has surged in recent years due to its capacity to tackle complex challenges across diverse domains. A defining feature of deep learning is the utilization of artificial neural networks, which are designed to emulate the structure and functionality of the human brain [22]. These networks consist of interconnected nodes (referred to as neurons) organized into layers, wherein information is processed and predictions are made based on input data [23]. The depth of a network is determined by the number of layers it possesses, hence the term "deep" learning. Deep learning models are trained by employing large amounts of data and intricate algorithms that iteratively adjust the weights and biases of each neuron within the network, with the aim of minimizing prediction errors [24]. This process, known as backpropagation, involves propagating errors backwards through the network to refine its parameters [25]. One notable advantage of deep learning over traditional machine learning is its capacity to automatically extract meaningful features from raw data, obviating the need for manual feature engineering. This attribute proves particularly advantageous in tasks such as image recognition [26] or natural language processing [27], where discerning significant features can be challenging and time-consuming. Following subsections will discuss particular technical details about some techniques used in our work.

2.1.1 Activation Functions

Activation functions are mathematical functions that inject non-linearity into the neural network, allowing it to learn intricate patterns in data. There are various manifestations that these functions can assume, with notable examples encompassing the sigmoid function, the hyperbolic tangent function, and the rectified linear unit (ReLU) function [28]. The sigmoid and hyperbolic tangent functions are S-shaped curves that map input values to a range between 0 and 1 or -1 and 1, respectively. The ReLU function is a simple threshold function that returns 0 for negative inputs and the input value for positive inputs. The selection of the activation function can greatly influence the performance of a neural network. Some activation functions may be more suitable for certain types of data or tasks than others.

GELU

GELU, short for Gaussian Error Linear Unit, is a highly effective activation function for neural networks, introduced in [29]. It is defined as the product of input x and the standard Gaussian cumulative distribution function, denoted as $\Phi(x)$:

$$\mathrm{GELU}(x) = x\Phi(x) = x \cdot \frac{1}{2}[1 + \mathrm{erf}(x/\sqrt{2})] \tag{2.1}$$

Approximation of GELU is done as following [29]:

$$\mathrm{GELU}(x) = 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right) \tag{2.2}$$

or

$$\mathrm{GELU}(x) = x\sigma(1.702x) \tag{2.3}$$

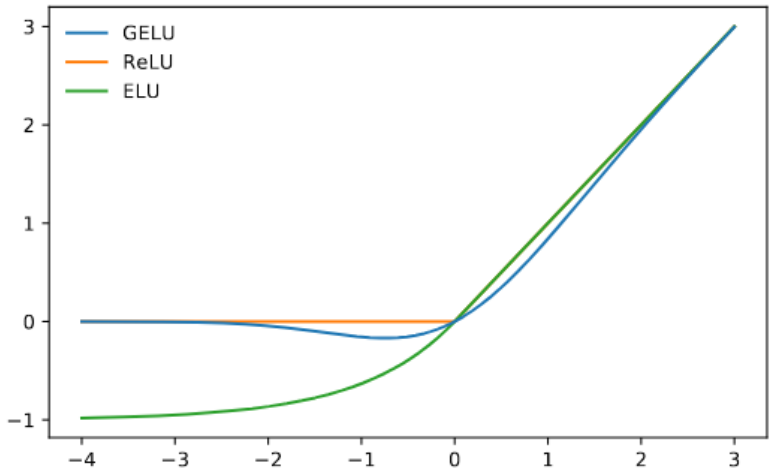


Figure 2.1: **The GELU activation function** in comparison with ReLU and ELU functions [29].

Unlike activation functions such as ReLU and ELU that gate inputs based on their sign, GELU weights inputs by their value. GELU is also characterized by being a non-convex and non-monotonic function that exhibits curvature at all points, which sets it apart from linear activation functions like ReLU and ELU that lack curvature in the positive domain. This enhanced curvature and non-monotonic nature of GELU activations enable them to potentially approximate complex functions more effectively compared to ReLU or ELU activations. Through empirical evaluations, the GELU nonlinearity has been extensively studied and demonstrated to offer notable performance enhancements across a range of tasks in computer vision, natural language processing, and speech.

2.1.2 Batch Normalization

Batch Normalization is a technique employed in deep neural networks to normalize the inputs of each layer [30]. Its primary objective is to address the issue of internal

covariate shift, which refers to the changing distribution of layer inputs during training due to parameter updates in previous layers. The Batch Normalization process involves normalizing the mean and variance of inputs within each mini-batch of data. To accomplish this, one subtracts the mean of the mini-batch from each element and then divides the result by the standard deviation of the mini-batch. By ensuring input normalization, Batch Normalization ensures that the inputs to each layer maintain a consistent distribution, irrespective of the previous layer's output distribution. Furthermore, Batch Normalization introduces two learnable parameters, namely scale and shift. These parameters enable the network to learn the optimal scaling and shifting factors for each normalized input, thereby allowing adaptation to different distributions. The advantages of Batch Normalization include accelerated convergence during training, the ability to employ higher learning rates, reduced sensitivity to parameter initialization, and improved generalization performance. Additionally, it acts as a regularization technique, reducing the necessity for other regularization methods like Dropout in certain cases.

2.1.3 Early Stopping

The purpose of early stopping is to identify the onset of "overfitting" during the training process and halt it before the model becomes overly specialized to the training data. Overfitting is a prevalent issue in machine learning, whereby a model becomes excessively specialized or "over-adapted" to the training data. As a result, the model's performance tends to degrade when applied to new, unseen data. In early stopping, after each training epoch, a prediction is made on the validation set to assess the loss

on unseen instances. It is important to note that the validation instances are solely used for evaluation and do not contribute to updating the network parameters. As the model approaches convergence, the validation loss may exhibit fluctuations around a particular value. In early stopping, a hyperparameter called "patience" is defined, which sets a threshold for the number of epochs to wait without any improvement in the validation loss. If there is an improvement compared to the best validation loss observed so far, the patience counter is reset to 0. However, if no improvement occurs within the specified patience limit, the training phase is terminated, and the testing phase commences.

2.2 Deep Generative Networks

Deep generative networks, or generative deep learning models, belong to a class of neural networks specifically designed to generate new data that closely resembles the training data they were trained on. The primary objective of these models is to learn the underlying distribution of the training data and subsequently generate new samples that exhibit similar characteristics. In this section, we will discuss Normalizing Flows 2.2.1 and Autoencoders 4.2.

2.2.1 Normalizing Flows

Normalizing Flows are generative models that are designed to learn the underlying probability distribution of a given dataset [31]. Their objective is to transform a simple base distribution, typically a standard Gaussian distribution, into a more complex distribution that closely matches the data distribution. This is achieved

by employing a series of invertible transformations, which enable mapping samples from the base distribution to samples from the desired target distribution. These transformations are specifically designed to be bijective, ensuring that they possess a well-defined inverse operation. This bijectivity property enables efficient computation of both sampling from the target distribution and evaluating its density. By applying a sequence of such invertible transformations, the model can effectively capture complex dependencies and generate realistic samples.

The change of variable formula plays a crucial role in Normalizing Flows, enabling the computation of the probability density of a sample in the target distribution based on its density in the base distribution. If we have a sample from the dataset $z \in Z$, with a simple prior distribution like a gaussian normal p_X in the prior space $x \in X$, and a invertible bijective function $f : Z \to X$ (with $g = f^{-1}$), we can define the change of variable formula that defines a model distribution Z by:

$$p_Z(z) = p_X(f(z)) \left| \det \left(\frac{\partial f(z)}{\partial z^T} \right) \right| \tag{2.4}$$

$$\log \left(p_Z(z) \right) = \log \left(p_X(f(z)) \right) + \log \left(\left| \det \left(\frac{\partial f(z)}{\partial z^T} \right) \right| \right) \tag{2.5}$$

where $\frac{\partial f(z)}{\partial z^T}$ is the Jacobian of f at z, *det* is the determinant of the jacobian and $||$ takes the absolute value. We can draw a sample $x \sim p_X$ in the gaussian space and by doing a forward pass, we can generate a sample $z = f^{-1}(x) = g(x)$ in the original space. To compute the density at a given point x in a normalizing flow, the process involves calculating the density of its image $f(x)$ in the target distribution and then multiplying it by the corresponding Jacobian determinant $det \left(\frac{\partial f(z)}{\partial z^T} \right)$.

2.2.2 Autoencoders

An autoencoder is a neural network model utilized for reconstructing its input signal [32]. It comprises two primary components: the encoder and the decoder. The encoder is responsible for compressing and encoding the input data, transforming it into a different representation space. This stage, known as the encoding phase, facilitates feature extraction. Subsequently, the decoder reconstructs the encoded data, reverting it back to its original representation. Autoencoders learn in an unsupervised manner. The training process can be formalized as following:

$$\arg\min_{E,D} E_X[\Delta(\mathbf{X}, D \circ E(\mathbf{X})], \tag{2.6}$$

where the goal of the model is to learn the function $E : \mathbb{R}^n \to \mathbb{R}^q$ (the encoder) and $D : \mathbb{R}^q \to \mathbb{R}^n$ (the decoder). E_X is the expectation over the original distribution X and Δ is the selected loss function.

2.3 Graph Neural Networks

Graph neural networks are a fundamental part of geometric deep learning (GDL). Geometric Deep Learning (GDL) is a sub-field of machine learning that concentrates on the advancement of methodologies aimed at extending the applicability of deep neural models to non-euclidean domains, including graphs and manifolds [33]. It involves adapting deep learning algorithms to work with data structures that do not have a traditional euclidean geometry, such as molecular conformations. The goal of geometric deep learning is to enable more effective analysis and prediction

of complex data in a wide range of fields. A graph neural network is a specific class of neural network architectures specifically tailored for processing and analyzing data structured as graphs. It can be efficiently utilized in various graph analysis tasks, encompassing a wide range of applications such as node classification, graph classification, network embedding, graph generation, and more. GNNs have many practical applications across different domains, including computer vision, natural language processing, social network analysis, and transportation systems.

2.3.1 Notations

Graph neural networks operate on graphs by propagating information between nodes and edges in the graph. A graph is a pair $G = (V, E)$ where each vertex (or node) $v \in V$ is connected with other vertices (or nodes) by edges $e \in E$. Each edge is an ordered pair $e_{ij} = (v_i, v_j)$, meaning that vertex v_i is connected to v_j.

A summary of basic notations for graphs [34] is given below, in order to better present the ideas in this thesis:

- $N(i) = \{j \in V \mid (i, j) \in E\}$ is the set of vertices that form the neighborhood of a node i.

- The adjacency matrix A of size $n \times n$ implies the which vertices are connected in the graph:

$$A_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E \\ 0 & \text{if } e_{ij} \notin E \end{cases} \tag{2.7}$$

- The degree matrix D has the dimension of $n \times n$. Each value d_{ii} in the diagonal

matrix denotes the degree of the vertex i. The d_{ii} is the number of attached edges on i in undirected graphs. For directed ones, we count the types of edges to differentiate between incoming and outgoing degree.

- $X \in \mathbb{R}^{n \times d}$ is vertex (node) feature matrix, with d number of features per node.
- $X^e \in \mathbb{R}^{m \times c}$ is the edge features matrix, with m cardinality of E and c number of features per edge.

2.3.2 Working Principles

The fundamental concept of GNN is to acquire a representation that encompasses both the local and global information of the graph. The GNN architecture typically consists of multiple layers, each of which updates the node representations based on the representations of their neighboring nodes and edges. The information propagation can be done through various mechanisms, such as graph convolution, message passing, or attention mechanisms. In every layer, the node representations undergo updates by assimilating information from their adjacent nodes and subsequently implementing a non-linear transformation. The aggregation function can be a simple sum or mean of the neighboring node representations [35], or a more complex function that takes into account the edge features and the direction of the edges [36]. Following are some of the GNN operators that can be used to generate embeddings and extract meaningful information.

GraphConv

The GNN operator by Morris et al. [37] is a learnable neural network that performs information aggregation from the local neighborhood of a node in a graph, yielding a new feature vector for that node. By employing an end-to-end training approach with classification or regression algorithms, the GNN can undergo training that encompasses the entire network architecture. This methodology facilitates enhanced adaptability and generalization capabilities, enabling the GNN to perform more effectively across different tasks. The operator is scalable to large graphs and flexible to different types of graphs and tasks. It takes as input the feature vectors of a vertex and its neighbors, and utilizes a neural network to combine and summarize this information. The resulting output from the neural network is then utilized to update the feature vector of the node. This process is repeated for all nodes in the graph, generating updated feature vectors that can be employed for downstream tasks such as graph classification or regression. The graph neural network (GNN) operator in the paper has be defined mathematically as following:

Let $f^{(t-1)}(v)$ be the feature vector of node v at layer $t-1$. The GNN operator computes a new feature vector $f^{(t)}(v)$ for node v at layer t as follows:

$$f^{(t)}(v) = \sigma\left(f^{(t-1)}(v) \cdot W_1^{(t)} + \sum_{w \in N(v)} f^{(t-1)}(w) \cdot W_2^{(t)}\right) \tag{2.8}$$

In equation 2.8, σ is a non-linear activation function, $W_1^{(t)}$ and $W_2^{(t)}$ are trainable weight matrices for layer t, and the sum is taken over all neighbors w of node v in the graph G [37].

SAGEConv

GraphSAGE [38] is a powerful method used to generate node embeddings in large graphs. It effectively combines information from a node's neighborhood, considering both the graph's structure and node attributes. This approach allows for the creation of meaningful node embeddings that can be utilized in downstream tasks. The SAGEConv or convolution function is a mean aggregator variant of GraphSAGE. It calculates the elementwise mean of the feature vectors from a node's local neighborhood. This mean operator shares similarities with the convolutional propagation rule employed in the transductive Graph Convolutional Network (GCN) framework [4]. In the SAGEConv algorithm, the mean aggregator is responsible for updating the node embeddings. It takes the previous layer's representation of the node, $\mathbf{h}_v^{k-1}$, and the feature vectors of its neighboring nodes, $\mathbf{h}_u^{k-1}$ for all u in $\mathcal{N}(v)$, where $\mathcal{N}(v)$ represents the neighborhood of node v. By computing the mean of these vectors, the aggregator assigns the resulting value as the new embedding for node v. Equation 2.9 shows this operation [38]:

$$\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W} \cdot \text{MEAN}\left(\left\{\mathbf{h}_v^{k-1}\right\} \cup \left\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\right\}\right)\right) \tag{2.9}$$

The mean-based aggregator utilized in SAGE-Conv can be interpreted as a linear approximation of a localized spectral convolution. It is worth noting that the mean aggregator does not involve concatenation between the previous layer's representation and the aggregated neighborhood vector, which differentiates it from other aggregators proposed in GraphSAGE.

TransformerConv

Unified Message Passing Model (UniMP), is a graph operator which combines feature and label propagation in a unified framework [39]. By leveraging a Graph Transformer network, UniMP effectively incorporates both feature and label embeddings, facilitating efficient information propagation during both training and inference stages. The Graph Transformer operation in UniMP can be represented mathematically as follows:

Let $H^{(l)} \in \mathbb{R}^{n \times d}$ be the hidden representation at the l-th layer of the Graph Transformer, where n is the number of nodes and d is the dimensionality of the hidden representation. The operation can be defined as:

$$H^{(l+1)} = \sigma\left(((1-\beta)A^* + \beta I)\, H^{(l)} W^{(l)}\right) \tag{2.10}$$

Equation 2.10 represents the propagation of information in the Graph Transformer. Here, A^* is the normalized adjacency matrix or attention matrix, $W^{(l)}$ denotes the weight matrix associated with the l-th layer, while σ represents the activation function, such as ReLU. The hyperparameter β governs the trade-off between the original hidden representation and the propagated information. The term $((1-\beta)A^* + \beta I)\, H^{(l)}$ represents the aggregation of information from neighboring nodes, where I is the identity matrix. The multiplication with $W^{(l)}$ applies a linear transformation to the aggregated information. Finally, the activation function σ introduces non-linearity to the transformed representation.

The Graph Transformer operation integrates attention mechanisms and takes inspiration from transformers to facilitate the propagation of information within a

graph. By employing attention mechanisms [40], the Graph Transformer operation computes attention weights that determine the importance of neighboring nodes for each node in the graph. These attention weights enable nodes to focus on relevant information from their neighbors during the information propagation process. This attention mechanism, coupled with the inclusion of weight matrices and activation functions, enables the Graph Transformer operation to proficiently capture and propagate information throughout the graph.

2.3.3 PyTorch Geometric

PyTorch Geometric is a framework designed to facilitate rapid representation learning on graphs, point clouds, and manifolds. It offers efficient tools and functionalities specifically tailored for these domains, enabling streamlined and effective learning processes [41]. It provides a wide range of tools for building and training graph neural networks, including various graph convolutional layers, pooling operations, and attention mechanisms. PyTorch Geometric also includes a large collection of benchmark datasets and state-of-the-art models for graph classification, node classification, and link prediction tasks. The work done in this thesis uses this framework extensively in the data representation, training and analysis stage. GraphConv, GraphSAGE, and TransformerConv described in sections 2.3.2, 2.3.2, and 2.3.2 all were implemented in Pytorch Geometric. The framework is built on top of PyTorch, which makes it seamless to use with PyTorch itself [42]. PyTorch Geometric is widely used in academic research and industry applications for tasks such as drug discovery [43], social network analysis[44] etc.

2.4 Molecular Dynamics

Molecular dynamics (MD) is a computational technique utilized to examine the behavior of atoms and molecules in a system over time [45]. It involves solving the classical Newtonian equations of motion numerically, starting from an initial state and subject to a set of appropriate boundary conditions. MD simulations can be employed to investigate various systems, including liquids, solids, and gases, as well as chemical reactions, protein folding [46]. The quality of MD simulations relies largely on how forces are specified, with some simulations utilizing empirical force fields while others utilize *ab initio* methods [47]. Despite limitations such as computational requirements and the inability to account for quantum mechanical effects, MD simulations have become a valuable tool for researchers studying complex systems at the atomic or molecular level. They enable detailed simulations that offer insights into fundamental physical and chemical processes, leading to new research paradigms. MD simulations present a means to determine free energy surfaces through the application of enhanced sampling methodologies. These methodologies utilize various biasing potentials or algorithms to enhance the exploration of the system's energy landscape and facilitate the computation of free energy surfaces. Through the integration of these techniques into molecular dynamics simulations, it is possible to acquire more precise estimations of free energy differences between different states or configurations of the system.

2.4.1 Free Energy

Free energy pertains to the energy available for performing useful work. It is commonly referred to as Gibbs free energy (G) [48]. Free energy encompasses the

system's internal energy, which accounts for molecular interactions, as well as the impact of external factors like temperature and pressure. The Gibbs free energy is defined as:

$$G = H - TS \tag{2.11}$$

Where H represents the system's enthalpy, encompassing both the internal energy and the energy required to compensate for changes in pressure and volume, T denoted the temperature (in K), and S denotes the system's entropy, which quantifies the disorder or randomness. However, we are more interested in relative free energy, which holds physical meaning in a chemical system. The difference in free energy (ΔG) between two states or configurations can provide insight into the stability of a system. We could estimate if a chemical process is favorable or not by measuring whether (ΔG) is negative or positive. While there are simplified cases or specific systems where an analytical solution for the free energy can be obtained, such as ideal gases or certain simple models, in most realistic and complex systems, an exact analytical expression for the free energy is not feasible due to the complex interplay of numerous variables within the system [49]. Scientists often rely on approximation methods and numerical techniques such as Monte Carlo simulations, molecular dynamics simulations, mean-field approximations, and various computational algorithms of the free energy through statistical sampling or iterative procedures. In the context of this thesis, free energy will refer to the difference in free energy. Meaning, the free energy of a particular conformation is assessed as a relative measure compared to the absolute minimum energy state within the system.

2.4.2 Free Energy Surface

Investigating the Free Energy Surface (FES) of a system is a subject of significance across various domains, including chemistry, materials science, and biophysics [50]. The free energy surface offers important insights into the stability and dynamics of a system. It is commonly visualized as a two-dimensional plot where the axes represent collective variables in the system and the intensity of color represent the free energy value. It can also be visualized a three-dimensional plot, with the axes representing the relevant variables (e.g., temperature and composition). Each point on the surface corresponds to a specific combination of these variables, while the elevation of the surface at that point represents the Gibbs free energy. Through the examination of the free energy surface, we can ascertain the system's most stable states and the circumstances in which these states are favored. For instance, the minima on the surface signify the equilibrium states of the system, where the Gibbs free energy attains its minimum value. Figure 5.4 (A) shows an example of a two-dimensional free energy surface in alanine-dipeptide.

2.4.3 Transition State

A transition state, also referred to as an activated complex, denotes a configuration that resides at the highest point of energy in a chemical reaction [51]. It manifests as an ephemeral and unsteady intermediate state that emerges during the conversion of reactants into products. The transition state theory, formulated by Henry Eyring and others in the 1930s [52, 53], outlines a framework for understanding the occurrence of chemical reactions at the molecular level. As per this theory, reactions progress

through a succession of steps involving the formation and breaking of chemical bonds. The transition state represents the point at which these bonds are simultaneously in the process of being broken and formed. The energy required to attain this state is known as the activation energy, dictating the pace at which a reaction transpires.

Understanding transition states is paramount for the design of novel pharmaceuticals [14], the development of more efficient industrial processes [54], and the acquisition of insights into fundamental chemical principles. Nonetheless, their transient nature renders them hard to directly observe, necessitating sophisticated experimental methodologies and theoretical models. Transition states can be explored through computational techniques such as density functional theory [55] or *ab initio* calculations [56]. They can also be experimentally discerned employing methodologies like spectroscopy [57] or kinetic analysis[57]. The following section will dive into the details about the transition state structure search.

Transition State Analogues

Transition state analogues are molecules that replicate the structure and chemical characteristics of the transition state of a specific reaction [14]. They are engineered to bind tightly to enzymes by stabilizing the conformation optimized through evolution for transition state formation. Instead of assuming the transient geometry of the transition state, transition state analogues transform the momentary transition state into a stable thermodynamic state. By forming high-affinity complexes with enzymes, transition state analogues serve as potent inhibitors, demonstrating potential in drug development for various targets, offering insights into enzyme mechanisms

and enabling the development of more efficacious drugs with reduced side effects.

2.4.4 Well-tempered MetaDynamics

MetaDynamics [58] is an improvement over molecular dynamics simulations, that provides us a quicker way to explore the free energy surface. In conventional molecular dynamics simulations, the system evolves based on equations of motion, typically governed by classical mechanics, while the potential energy is computed by considering interactions among atoms or particles. Nonetheless, in systems characterized by rugged energy landscapes, the simulation may become trapped within local energy minima, impeding the exploration of diverse states or transitions. MetaDynamics surmounts this obstacle by incorporating a biasing potential into the original potential energy function. This biasing potential, referred to as the MetaDynamics potential or history-dependent potential, is constructed as the sum of Gaussian-shaped repulsive potentials deposited throughout the simulation along selected collective variables (CVs). CVs represent a condensed set of variables that describe the pertinent degrees of freedom in the system, such as distances, angles, or torsional angles. Well-tempered MetaDynamics [59] improves over MetaDynamics. It solves the convergence problems that are often encountered in traditional MetaDynamics. This means that it provides a smoothly converging and tunable free-energy method, making it more efficient and reliable.

2.4.5 Committor Analysis

Committor analysis is a computational technique utilized to detect transitional stages in chemical reactions or complex systems [60]. The fundamental concept underlying committor analysis involves computing the committor probability for each point along the reaction coordinate. The committor probability, often represented as $Q(x)$, signifies the probability of the system reaching the final state from a specific configuration or point x before returning to the initial state. The method involves generating a multitude of trajectory samples of the system, typically through molecular dynamics simulations or other suitable techniques. By monitoring the progress of these trajectories, the committor probability function can be estimated. Analyzing the committor probabilities along the reaction coordinate enables the identification of points where the committor probability is approximately 0.5. These points correspond to transition states, where the system has an equal chance of progressing towards the initial or final state.

Chapter 3

Related Work

Since the goal is to generate molecular configurations conforming to transition states using machine learning on graphs, this work builds on the progress made in different sub-fields. Thus, this chapter has been divided into three sections. In the section 3.1, we discuss some of the prior work done in graph representation and learning. In the section 3.2, we look at some of the relevant advancements done in generative machine learning. And in the section 3.3, we diverge away from ML to discuss some of the techniques in computational chemistry and statistical mechanics.

3.1 Learning on Graphs

3.1.1 Graph Neural Networks

Neural networks have contributed much to the recent success of machine learning. Deep learning techniques have made the learning of complex data and representation much efficient. Classic deep learning methods have been most effective on data

having an underlying Euclidean or grid-like structure. To handle more complex data representations, the notion of Deep Neural Networks have been extended to graph structured data which doesn't have a typical grid-like structure [4, 33]. Numerous real-life phenomena can be logically depicted using graphs, such as molecules, where the vertices of the graph can symbolize atoms and the edges can represent bonds. Graph Neural Networks (GNNs) have been architectured to handle non-euclidean structured tasks such as node classification, link prediction, and clustering. Graph Representation Learning [61, 62, 63, 64] is an area of research focused on the advancement of machine learning techniques that aim to acquire, infer, and generalize knowledge from data structured as graphs. Graph representation learning encompasses a range of methodologies aimed at generating deep graph embeddings. These techniques primarily concentrate on converting graph structures into low-dimensional vector representations that encapsulate significant features and patterns inherent in the graph [65], extensions of convolutional neural networks to data structured as graphs [4], and neural message-passing methods inspired by belief propagation [66]. The graph convolution operation introduced by Bruna et al., known as spectral-based graph convolution, relies on the graph Fourier transform, which can be seen as an analogous concept to the Fourier transform of 1-D signals [4, 67]. Although groundbreaking, there are certain limitations to this spectral based approach, mainly in the lines of computational complexity. One of the main challenges is that they require the computation of the graph Fourier transform, which can be computationally expensive for large graphs ($O\left(n^3\right)$). Graph Estimation process as an extension of spectral Networks for graphs [68] have been proposed as an improvement. By estimating the

graph structure from data, this method is able to adapt to the specific structure of the data. Moreover, the paper introduces a novel pooling operation that diminishes the dimensionality of the graph domain, effectively mitigating the computational expense associated with the Graph Fourier Transform operation. ChebNet [69] proposed by Defferrard et al. improves upon the described approaches in several ways. This particular model offers precise control over local filters, allowing for fine-grained manipulation of filter behavior. Moreover, it achieves computational efficiency by eliminating the need for explicit utilization of the Graph Fourier basis, which was a prerequisite in previous approaches. ChebNet utilizes Chebyshev polynomial approximation [70] to perform spectral graph convolution computations. All of these contributions lead to a model that shows higher experimental accuracy than previous spectral based approaches. The Graph Convolutional Network (GCN), introduced by Kipf et al. [35] is a framework utilized for the semi-supervised node classification task on graphs. This paper uses first order Chebyshev polynomial, leading to more simplified convolution operations. FastGCN [71] was later proposed as an improvement over GCN. The approach presented in this paper extends transductive training to an inductive setting, addresses the memory bottleneck problem in GCN caused by recursive neighborhood expansion, and introduces a sampling scheme that reformulates the loss and gradient. The authors of the study also showcase the algorithm's convergence, regardless of the sample size employed during the training phase. On the other hand, non-spectral based approaches offer an alternative to spectral approaches by directly defining convolutions on the graph, focusing on spatially proximate neighboring groups. Notable methods in this category include those proposed by Duvenaud et al. [72], Atwood

& Towsley [73], and Hamilton et al. [38]. Unlike spectral approaches, non-spectral methods do not rely on the Laplacian eigenbasis, which is contingent upon the graph structure. Consequently, they can be applied to graphs with diverse structures. Nevertheless, these approaches may face limitations concerning receptive fields, which pertain to the range of input features capable of influencing the output of a layer, as well as the absence of weight sharing across distinct regions of the graph.

3.1.2 Molecular Graphs

In the case of molecular graphs, every unique permutation represents a unique physical structure. Preserving the permutation invariance is of utmost importance. Permutation invariant graph message passing networks have been proposed that strongly correlate with chemical processes [74]. This paper introduces a new framework called Message Passing Neural Networks (MPNNs) for learning on graph-structured data, including molecular graphs. MPNNs are a generalization of existing spectral convolutional neural networks on graphs. It is important to encode the structural information of a graph, i.e. which edges are more important. This is handled by very useful "attention mechanisms" which focuses on most relevant parts of the input [75, 76, 40]. Velickovic et al. extends these attention mechanisms to graphs [77] and calls them "Graph Attention Networks (GATs)". Graph Attention Networks (GATs) offer the assignment of varying weights to individual nodes within a neighborhood. GATs accomplish this objective without relying on computationally intensive matrix operations or necessitating prior knowledge of the graph structure. GATs have been employed to enhance the performance of GNNs in the field of drug discovery [78]. Within

the context of molecular representations, the attention mechanism has been utilized to obtain a context vector for each atom in a molecule by emphasizing its neighbors and local environment. This approach enables the model to capture non-local effects among atoms, facilitating the learning of both local and non-local properties within a given chemical structure. Shi. et al. proposed a Unified Message Passing model (UniMP) that combines graph neural networks (GNNs), attention mechanism, and message passing [39]. Specifically, UniMP employs a graph transformer that jointly uses label embedding to combine feature and label propagation together. The graph transformer is a variant of the transformer model that has been widely used in natural language processing (NLP) tasks such as much talked about ChatGPT [79] and in protein folding [8]. A lot of graph learning architectures have been formulated recently each having slightly different applications, such as GraphSAGE [38], GCPN [80], MONet [81], (k-GNN) [37] etc. These different graph learning architectures can be used for inductive representational learning on large graphs. Graph learning has been used for molecular representation, prediction and in general to solve problems in computational chemistry [82]. Application include use in quantum chemistry [83], drug discovery [84, 85, 9], molecular property prediction [86, 87] etc.

3.2 Graph Generation and Generative Networks

3.2.1 Autoencoders

Learning the underlying data distribution is one of the fundamental goals of ML. Autoencoders (AE) can be used for learning latent data representation in an un-

supervised fashion [88]. Autoencoders comprise an encoder and decoder network, responsible for mapping input data to a lower-dimensional latent space and reconstructing it back to the original input space, respectively. The objective is to minimize the reconstruction error while imposing constraints on the dimensionality of the latent space. Kipf et al. introduced Variational Graph Auto-Encoders (VGAE), a novel method for unsupervised ML training on graphs. The VGAE utilizes a Graph Convolutional Network (GCN) encoder and a straightforward inner product decoder [89] for learning interpretable latent representation of graphs. By employing graph autoencoders, the model becomes capable of learning representations that encapsulate the local structure of the graph. This aspect is particularly valuable for tasks like link prediction and reconstruction. Graph autoencoders have been extensively utilized for graph embedding purposes [90], graph clustering [91, 92], and molecular graph generation [93, 94] etc. Conditional variational autoencoders have been utilized to generate *de novo* molecules with specific target properties. This approach enables the generation of molecules tailored to meet desired criteria.[95].

3.2.2 Normalizing Flows

Generative modeling is being used more frequently to model a probability distribution given examples drawn from the target distribution. The aim is to learn the underlying distribution of the data and generate new samples that are similar to the original data. There exists different approaches to generative deep learning such as the aforementioned Autoencoders, GANs [96]. But neither method permits an accurate assessment of the probability density of new data points. Training these

models can present challenges due to various issues, including mode collapse, posterior collapse, vanishing gradients, and training instability. These difficulties can hinder the effective training of the models and require careful consideration and mitigation strategies [31]. Normalizing flows (NF), as a generative modeling framework, effectively addresses several challenges through its tractable distributions, allowing for precise and efficient sampling and density evaluation. NFs were introduced by Tabak et al. [97, 98] but have been used for variational inference [99], density estimation [100, 101, 102] etc. A normalizing flow (NF) utilizes a series of invertible and differentiable mappings to convert a basic probability distribution (such as a standard normal) into a more complex distribution. The evaluation of the density of a sample can be achieved by reversing the transformation process and returning it to the original uncomplicated distribution. Many paradigms of NFs have been introduced in the literature such as planar and Radial flows [99], Coupling flows [100, 103], Autoregressive flows [104] etc. These different types of flows differ in their architecture and the way they transform the input distribution. Each type of flow model possesses distinct strengths and weaknesses, and the selection of a particular flow model depends on the specific task at hand. The choice of flow model should be made by considering the specific requirements and characteristics of the task in question. Real NVP [105] is a type of normalizing flow, which employs a collection of powerful, consistently reversible, and trainable transformations to map a simple distribution to a more complex one. NVP stands for "non-volume preserving" and refers to the fact that the transformation does not preserve the volume of the data space. This is important because it allows for more flexibility in the transformation, which can lead to

better modeling of complex distributions. Normalizing flow based models have shown their capability in generating molecular graphs in an unsupervised setting [106, 107, 108]. NF and Real NVP has been extended to work with graph neural networks, which facilitates both prediction and generation of graph structures. This proposed model is called "Graph Normalizing Flows" [109]. In the context of unsupervised learning, the integration of graph normalizing flows with a graph auto-encoder yields a generative model capable of constructing graph structures. Notably, this model exhibits permutation invariance, enabling the generation of complete graphs through a single feed-forward pass.

3.2.3 Generative Chemistry

Generative chemistry has significantly propelled the domain of computational chemistry and drug discovery by introducing a more streamlined approach to designing and discovering novel molecules with desired characteristics [110]. Conventional methods typically entail the laborious and resource-intensive process of synthesizing and evaluating numerous compounds. In contrast, generative chemistry leverages artificial intelligence (AI) techniques to create virtual compounds that can undergo screening for desired properties prior to actual synthesis. Generative models in chemistry are designed to learn the probability distribution of molecular structures and their properties from large datasets. It has been shown that a generative model for molecules need far less data than previously thought [111]. Generative models have been mostly used for generation of *de novo* compounds. Autoencoders [112, 113], Recurrent Neural Networks [114, 115], Reinforcement Learning [116, 117], Generative

Adverserial Network [118, 119], and Normalizing Flow [120, 121, 122] all have been used in different ways for generation and identification of new molecules. Representation of molecules plays a pivotal role in machine learning applied to chemical physics, as the accuracy of a machine learning model relies on the descriptors employed to represent the atomic-scale structures under investigation. Jinnouchi et al. and Low et al. demonstrated that the selection of input representation bears substantial influence on the model's ability to predict structure-property relationships accurately [123, 124].

3.3 Learning the FES and Sampling TS Geometry

3.3.1 Free Energy Surface Modeling

The FES is very useful in determining certain features of a molecular system [125, 126] such as finding the minimum energy pathway (MEP) and the location of stationary structures. Thus it is important for any framework to learn the underlying FES distribution with the goal of finding accurate geometry on the FES. *Ab initio* methods exist to provide FES based on Density Functional Theory (DFT) calculations [127], but they are extremely time consuming. Feed forward neural networks have been used to model the global properties of free energy surface in low dimensional systems [128, 129] and later in higher dimensional systems [130, 131]. This is achieved by taking into account the positions of all atoms from a three dimensional grid. More recently Smith et al. [132] introduced "ANI", which builds upon previous symmetry functions to develop transferable neural network potential for organic molecules.

Faraji et al. demonstrated the power of this method for calcium flouride [133]. SchNet [134] and PhysNet[135] were also proposed around the same time in the context of modeling quantum interaction in molecules. SchNet in particular can follow many quantum-chemical constraints such as rotational equivariance and energy-conserving force predictions. Apart from using NN, kernel based models have also been used in this context. Bartok et al. proposed "Gaussian-approximated Potential" (GAP) [136], which relies on Gaussian process regression (GPR). GPR based potential learning has been used more recently with a lot of success by different groups [137, 138]. Recently, graph machine learning has been successfully used in transferring molecular knowledge from a simple to a significantly more complex system [139]. Even with all of these advances in quantum machine learning, one major challenge remains. Which is efficiently and properly modeling the FES of large and complex systems with the goal of sampling complete molecular geometries at will from any point of the FES.

3.3.2 Finding MEP and TS

In transition state theory, The minimum energy pathway (MEP) in a free energy surface is the path that connects two equilibrium states [140]. The long term mechanics of a physical system is often marked by the rare jumps between these equilibrium states or energy minimas [141]. The transition path can be used to evaluate the transition rate, which has applications in catalyst discovery. Along the MEP, there exists a high energy barrier known as the transition barrier. At this barrier, there is a stationary but transient molecular configuration which is very hard to observe since it lasts for a very short period of time. The transition state structure corresponds

to a first-order saddle point on the FES. The nudged elastic band (NEB) method is widely used for finding saddle points on free energy surface [15]. The NEB approach optimises the images (atomic position) while keeping the space between them constant. The saddle point is reached when the total of these spring forces equals zero. There have been many improvements made to the model [16, 17, 142] including an active machine learning based approach to decrease the number of *ab-initio* calculations required [143, 144]. Another similar iterative based approach for finding the MEP is the string method [145], which has also seen some improvements over the years [146, 147]. But both of these methods suffer from similar problems. These methods are computationally expensive and require extensive knowledge of the FES. Machine learning based so-called "Boltzmann Generator" is a data-based approach which could be used to explore the FES [20]. Boltzmann generator is a type of "Normalizing Flow" architecture that has the ability to sample equilibrium configurations from different low energy states in one shot. It can also generate reasonably well guesses for the MEP which could be used to approximate the transition state. Boltzmann generator has already seen applications in physics in implementing a quantum inspired numerical annealer [148] and has been improved to find minimum energy pathways [149]. But, Boltzmann generator by itself cannot provide us the transition state geometry even if a TS has been identified.

The process of finding the transition state is referred to as structure optimization. This works by starting with a reasonably well initial guess and performing optimization on the internal coordinate to land on the transition state (TS) structure. Transition states are often more difficult to find and the computational investment

required in this process is high and can become more sophisticated depending on other factors (eg. level of theory). Many classical algorithms have been developed for transition state optimization [150, 151, 152]. However, coming up with a guess TS can be a tedious process requiring manual construction, trial and error adjustments and prior knowledge about the chemical process [153]. Some computational approach for generating approximate TS structure have been developed [154, 155, 156]. These methods are very expensive in terms of computation required and a good guess is never guaranteed. Recently, machine learning has been applied to this problem to try to solve it in a data-based approach. ML for finding guess TS geometry has mostly concentrated on supervised approach, where a starting state and a final state is present and sometimes requiring example TS [157, 158, 159, 18, 160, 161]. These models work by observing the transition states of different reactions to approximate a good starting TS. This approach is not generalizable since (1) it assumes that the system is a reaction that has a starting and ending state which makes it impossible to work with other systems, (2) it is a supervised approach which requires examples transition states for training which are rare to observe, and (3) these methods often require the reaction coordinates as well.

Chapter 4

Proposed Methodology

In this chapter, we discuss our proposed methodology in detail. In the section 4.1, we describe our overall pipeline using a top-down approach. In the section 4.2, we discuss the architecture and workings of our autoencoder model. We then discuss our normalizing flow based FES sampler in section 4.3. Finally, we describe the free energy prediction model in section 4.4.

4.1 Overall Pipeline/Architecture

Our goal is to build and train a machine learning model to sample molecular conformations that correspond to the transition state structure in any molecular system. There are couple of things we have to consider when designing the whole architecture at a higher level:

1. Since we are working with molecules, we have to translate the molecular conformations into something that could be ingested into a neural network archi-

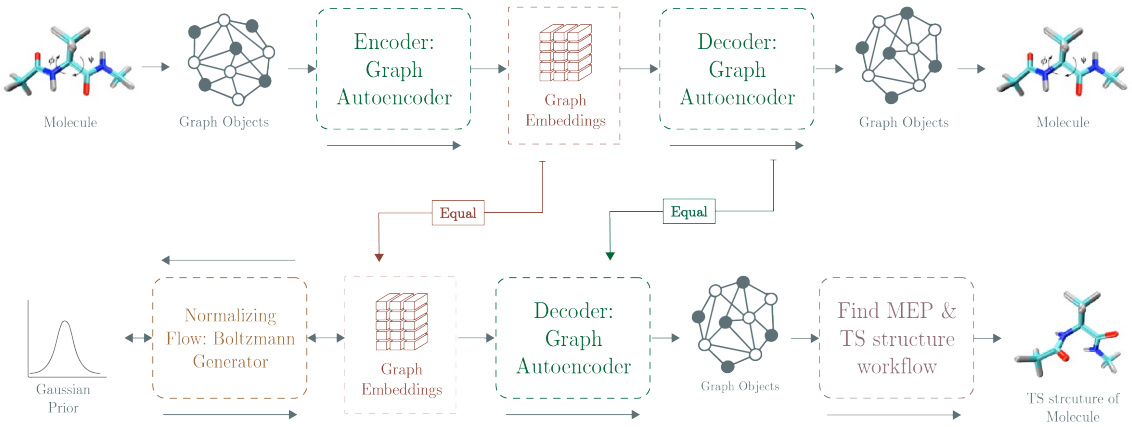


Figure 4.1: **Proposed architecture** to sample transition states in molecular systems. The architecture is divided into different sub-modules that performs specific tasks.

tecture. For this we select a molecular graph representation (described in detail in section 5.1.2).

2. The input graphs may vary in size depending on the molecular system. The whole pipeline should be general enough to handle different systems.

3. The whole pipeline should be permutation invariant, meaning it should not learn in which order data is being fed to it. We should be able to randomly feed input graphs to train and test the model.

4. Each molecular graph and each graph embedding should have a corresponding free energy value assigned to it (from MD simulation). We should not have to do any probabilistic inferencing that requires energy value prediction in the training phase (to reduce errors).

5. Different parts of the architecture should be able to interface with each other in a modular way. For example, the free energy prediction module should be

able work with the output from the resultant transition state.

6. We should be able to run the whole pipeline to sample transition states in a swift manner. The pipeline should to able to scale to larger systems.

With this general goal in mind, we design our novel machine learning architecture for sampling transition states in molecular systems. Figure 4.1 gives an overall higher level view of the whole pipeline.

4.1.1 Pipeline Details

In this section, we describe the parts of the pipeline in brief, specifically how the pipeline functions and some details about the specific modules. This section will refer to the figure 4.1 for illustration. The overall pipeline could be divided into three phases as follows:

1. **The graph autoencoder phase**: In this phase, the model is fed the input data. This input is a suitable graph representation of molecular conformations. The goal of this graph autoencoder is 1) to compress the input graphs into lower dimensional graph embeddings, that contain meaningful representation of the input graphs, and 2) to train the decoder to reconstruct molecular graphs from the graph embeddings with minimum loss in information. This decoder is later used in the downstream tasks.

2. **Boltzmann generator training**: In this phase, the graph embeddings from the previous stage are fed into the boltzmann generator (BG) to train the normalizing flow based model. The goal of the BG is to learn to transform the

distribution of the graph embeddings into a much simpler gaussian prior. Training the BG allows us to transform and interpolate in the latent space to find transition states.

3. **Sampling transition state structures**: In this final stage, we sample transition state structures. After selecting the initial and final states in the FES, we perform backward passes in the normalizing flow based model. The graph embeddings are then passed through the decoder to generate graph objects. These graph objects are then put through a workflow to find the most ideal candidates. Which are then validated to find the resultant TS.

Now, we will discuss details about our proposed architecture along with the details. Since, the entire architecture has different modules, this following texts have been divided into different sections. Section 4.2 describes the graph autoencoder model for generating graph embeddings, section 4.3 describes the Boltzmann generator of normalizing flow model, and section 4.4 describes the free energy predictor model.

4.2 Autoencoder

As discussed in section 4.1.1, training via a graph autoencoder is the first phase of the training and sampling pipeline. In this section, we will discuss about the autoencoder phase in detail.

4.2.1 Methods

The graph autoencoder (AE) follows the non-probabilistic graph auto-encoder model from [89]. Figure 4.2 shows the model architecture. We implement a non-probabilistic graph autoencoder to have a 1 to 1 correlation with the free energy value of every input molecular graph. Using a variational inference framework would require for us to predict the energy values for all of the input graphs. Since, a variational autoencoder would generate a new graph in the inference phase. This would introduce errors in two stages: once when generating a new graph from the learned mean and variance, and later when predicting the energy of the generated graph molecule. This is completely unnecessary since we have access to the free energy information correlating to every input graph. So by using a non-probabilistic model, we reduce errors on downstream tasks and as a result, the whole pipeline. The non-probabilistic model works in a very simple way, leaving the task of generating embeddings to the various graph operators. The encoder portion of the autoencoder works in the following way:

$$\mathbf{Z} = \mathrm{CONV}(\mathbf{X} * \mathbf{A}) \tag{4.1}$$

Where X is the node features and A is the adjacency list. The $CONV$ operator here are the different convolution operators that could be used. These operations give us the resultant graph embeddings Z. At each layer, the convolution output tensor goes through an activation function and batch normalization. These encoder layers are stacked and compress the data gradually. We experiment and compare two graph convolution operations here, SAGEConv and TransformerConv. We built

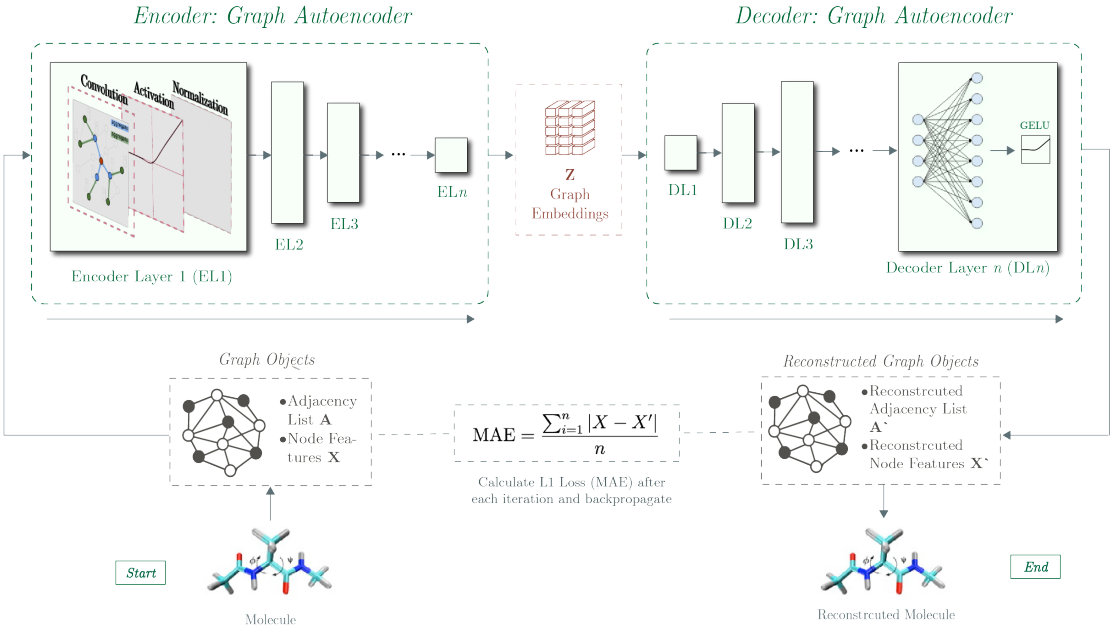


Figure 4.2: **Proposed Autoencoder Architecture.** This AE takes molecular graph representation as input to generate graph embeddings using a graph convolution operation. The embeddings are then upsampled using a linear decoder. The model is trained on every example using MAE loss.

two different models with these two different convolution operations, each with six encoder layers. Both of these models start with an input channel size of 128 and end up with a graph embedding or latent space with the size of 4. The TransformerConv model uses 1 head for attention.

The decoder portion of the model works in the following way:

$$f_i(x) = \sigma\left(w_i Z + b_i\right) \tag{4.2}$$

Which is just a standard feed forward neural network. In equation 4.2, w denotes the weight and b denoted the bias of the i-th layer. These decoder layers are stacked to reconstruct the input molecular graph. The decoder network is kept constant

for both the encoder models, starting with 4 input channels from the latent space. Which is then upsampled to 32 channels for the final layer. We use 5 layers in the decoder module with the last layer being the prediction or output layer. Unlike the traditional graph autoencoder, we do not reconstruct the adjacency list. That is because of the unique nature of our molecular configurations. In our dihedral angle representation of the molecule, the connections between the nodes stay constant, meaning the vertices do not change from frame to frame. We still use the adjacency information for embedding generation, but we do not need to go through the process of adjacency list prediction. The activation function used in both the encoder and decoder is GELU (see section 2.1.1).

4.2.2 Training

Training is done on the input molecular graphs described in section 5.1. The data is randomly ingested into the network. Since we use sin-cos decomposition where data is normalized, we do not normalize the data again. We split the data into three sets: Train set which contains 20% of the data, Validation set which contains 10% of the data and test set which contains 70% of the data. We use Adam optimizer [162] with a learning rate of 0.0001. We set a iteration number for the model of 500, but use early stopping with a patience of 50.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |X - X'|}{n} \tag{4.3}$$

Equation 4.3 shows the L1 loss of mean absolute error (MAE) used for training the model. This measures the sum of absolute error between the input node features

X and the output node features X' over n examples.

4.3 Normalizing Flow (Boltzmann Generator)

In this section we will discuss the normalizing flow model that is based on the boltzmann generator [20]. With this model, the goal is to learn the underlying FES distribution and to be able to transform a simple gaussian distribution into the target distribution. The details are below.

4.3.1 Methods & Training

The normalizing flow model is trained on the graph embeddings copied from the autoencoder model. As discussed in the previous sections, the goal of the autoencoder model is to generate graph embeddings for our downstream tasks. After training the AE for optimal reconstruction, we again cycle through all the examples in our dataset and save the graph embeddings related to every example. This collection of graph embeddings is then fed into the boltzmann generator for training. The architecture of the boltzmann generator is shown in the figure 4.3.

The Boltzmann Generator begins with a prior distribution, which is a simple gaussian distribution called $p_X(\mathbf{x})$. Random vectors x are sampled from this distribution. The sampled vectors x undergo a transformation using a neural network, denoted as $\mathrm{f}^{-1}(\mathrm{x} \rightarrow \mathrm{z})$, where f represents the transformation function. This neural network is trained to learn a mapping that converts the prior distribution samples to configurations with high boltzmann weights. The boltzmann generator used in this thesis is trained on examples, which are the graph embeddings Z. Training by example in-

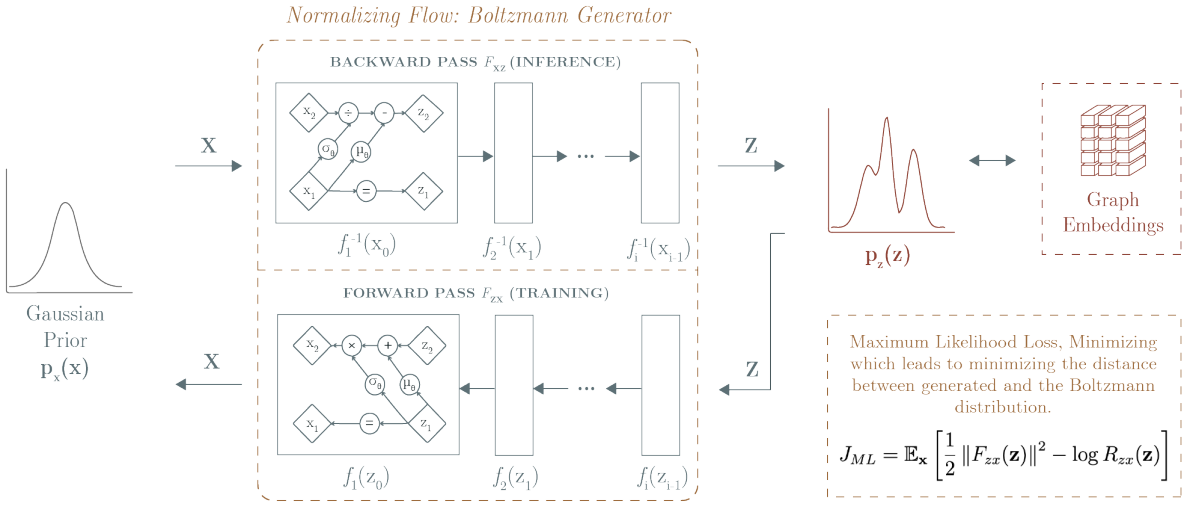


Figure 4.3: **Proposed BG architecture**. This model is trained on graph embeddings, using the maximum likelihood loss. The goal is to learn the transformation from a prior distribution X to a target distribution Z. The input data, either the graph embeddings Z or the latent vector X (depending on the direction), is split into a channel of "even" and "odd" dimensions in each RealNVP [105] block. These blocks are stacked together to form a powerful invertible neural network.

volves minimizing the discrepancy between the generated distribution $p_Z(\mathbf{z})$ and the target boltzmann distribution. This is done by minimizing the maximum likelihood objective function J_{ML}, which measures the difference between the generated distribution and the target boltzmann distribution. The training loss function is shown in equation 4.4:

$$J_{ML} = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{2}\|F_{zx}(\mathbf{z})\|^2 - \log R_{zx}(\mathbf{z})\right] \tag{4.4}$$

In this equation, $F_{zx}(\mathbf{z})$ represents the transformation performed on the latent space variables z using the trainable neural network. The energy of a configuration is proportional to the square of the norm of the force acting on it. By minimizing the squared norm of the force, the generated samples are encouraged to have a lower

energy. The associated Boltzmann weight of this transformation is denoted as $R_{zx}(\mathbf{z})$. The logarithmic term quantifies the extent to which the network scales the volume of the configuration space at a point x. The Boltzmann generator's jacobian matrix determinant quantifies the local expansion or contraction of the generated configurations. Subtracting the logarithm of the Boltzmann weight within the loss encourages the generation of samples with greater probabilities under the exact distribution. Finally we take the expectation over all the samples.

The architecture employed in this normalizing flow pipeline involves RealNVP [105] blocks. Put simply, the non-volume preserving blocks in Real NVP divide the data into two parts, keeping one part unchanged while subjecting the other to a neural network for learning a transformation. By applying distinct transformations to each data half, Real NVP enables the capturing of intricate dependencies and interactions among variables. The network learns this transformation during the forward pass. During the backward pass, the transformation is reversed by applying the inverse of the transformations used in the forward pass. This entails splitting the transformed data into two halves and sequentially applying the inverse transformation to each half. By employing the inverse transformation, the transformed data is mapped back to its original data space, allowing for the recovery of the graph embeddings Z.

We stack 10 RealNVP blocks to create our boltzmann generator architecture. The hidden layer consists a feed forward neural network with 128 channels. The prior distribution is set as a Gaussian normal distribution. We use a batchsize of 128 and train for 30,000 iterations over the data.

4.4 Energy Prediction

The energy prediction model used in this thesis reproduces the work by Demetrio [163]. To summarize, the FES prediction model uses a graph neural network to predict the free energy value of a molecular graph. The graph neural network model consists of 5 GraphConv convolution layers, the output of which is then pooled into a sequential neural network and finally flattened for a prediction. More details about the implementation could be found in the original cited thesis. The dataset used in our implementation is the alanine dipeptide MD simulation with the free energy values. In our reproduction of the model, we use the stochastic gradient descent (SGD) optimizer with a learning rate of 0.001, use a train-validation-test split of 20%-10%-70% of the dataset, and use early stopping with patience of 50. We also normalize the free energy values for more stabilization in training.

Chapter 5

Results and Discussion

In this chapter, we discuss about the dataset in section 5.1 and our experimental results for the graph autoencoder, normalizing flow model and free energy predictor model in sections 5.2,5.3, and 5.4 respectively. We also discuss about the candidate transition states and share the details about the successful sampling of a transition state structure in section 5.5.

5.1 Dataset

The dataset used in this project comes from data produced by an in-vacuum MetaDynamics simulation of alanine dipeptide. Each "frame" in the dataset refers to a snapshot of the system at a given point in time, encompassing the positions and movements of all atoms within the simulation. Each frame captures the spatial arrangement and motion of the atoms, facilitating the analysis of various system properties, such as energy, structure, and dynamics. The MetaDynamics simulation

and all the related process is complex and describing them in depth is out of scope of this thesis. In short, these are the steps that are done sequentially:

1. Conducting *in-vacuum* well-tempered MetaDynamics simulation of an initial Alanine Dipeptide representation.
2. Verifying the convergence of the simulation.
3. Estimation of the Free Energy Surface.
4. Extraction of frames, processing PDB (protein data bank) and topology files.
5. Creating the final dataset in .json form with the corresponding free energy value.

The Meta Dynamics simulation was carried out for $200ns$ with a timestep of $1fs$. The temperature of the system was set to 300°K (room temperature). This process leaves us with .json file corresponding to each file in the simulation and the free energy of that particular frame. JSON (JavaScript Object Notation) is a simple and widely adopted data format utilized for data interchange which employs a human-readable text-based syntax to represent structured data in the form of key-value pairs. We will now describe the information contained in this dataset.

5.1.1 Dataset Details

Each frame of the dataset contains the following information about alanine dipeptide at a certain moment in time:

1. Atom type (8 count)
2. Atoms (22 count)

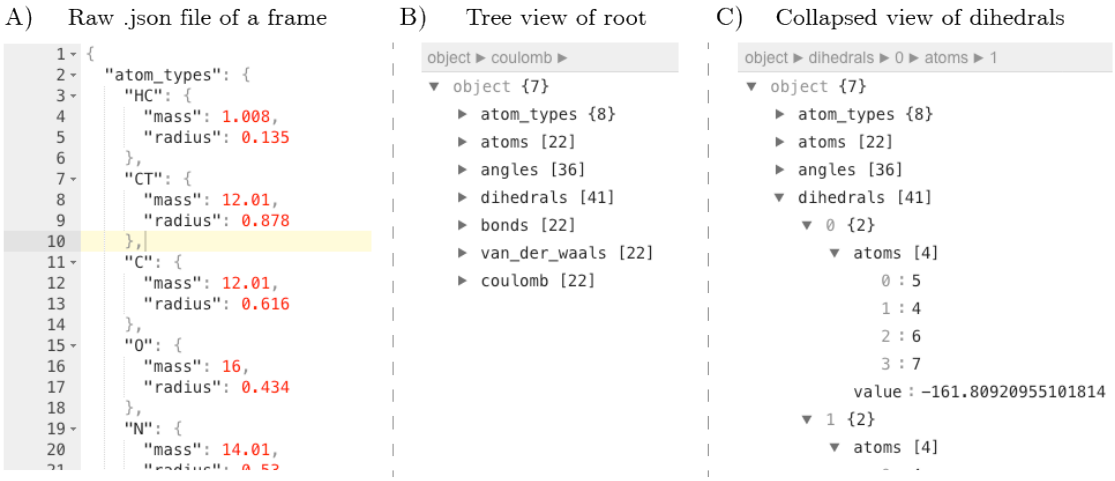


Figure 5.1: **View of the dataset**. A) The raw .json file corresponding to the first frame of the simulation, B) tree view of the root of the frame, and C) collapsed view of the dihedral branch.

3. Angles (36 count)

4. Dihedrals (41 count)

5. Bonds (22 count)

6. Van der waals force (22 count)

7. Coulomb force (22 count)

A snapshot of the data contained by the initial frame (*0.json*) has been provided in Fig 5.1.

Along with the above mentioned information, we need the free energy value of each frame. As the Free Energy Surface (FES) derived from the simulation is discretely represented as a function of the two dihedrals, determining the value for each frame requires identifying the nearest pair of ϕ and ψ, in the FES domain. Consequently,

frames with comparable yet distinct ϕ and ψ, values may share the same closest point and, thus, be assigned the same free energy label. The free energy values are saved in a .txt file, where each line contains a free energy value corresponding the .json file number. From the original 400,000 "frames" of the simulation, we select 50,000 in total. This is done to speed up training of our whole pipeline. It should also be mentioned that 50,000 frames are sufficient to represent the system in full and in the context of chemical models, it has been shown that more data is not always better [164]. The dataset contains information about 41 dihedrals of alanine dipeptide. For each dihedral d_i : the tuple of indexes of the a atoms array, named j, k, l, m, and the dihedral value are contained in the JSON file.

5.1.2 Graph Representation

The goal of any representation is to exploit the inherent structure and information. The representation should be simple, in the sense that it is reusable and extendable. We should be able to take in other datasets that were created from MetaDynamics simulation and run our algorithm on it. To tackle our problem, we need to answer a handful of questions, such as: how to represent the molecule, how to use each frame and its information to our advantage, and how to to exploit the free energy information. As we have already shared, our approach is based on graphs. That is, the input to our pipeline should be a graph and the resultant molecule should also have a graph representation. Clearly, many things could be included in the molecular representation. Information about coordinates, dihedral angles, bond information, van der waals and coulomb forces are included in the dataset. But for reliable and reproducible re-

sults, we should select the most important attributes from the dataset. If needed, more complex structural information could be in encoded in future approaches.

As we have already explored, a lot of information about alanine dipeptide could be explained as a function of the two dihedral angles ϕ and ψ. This inspired us to build the graphs around the dihedral information. We use overlap graphs, which are very common in computer algorithms and genomics research [165, 166]. An overlap or De Bruijin graph [167] is a graph structure that involves representing sequences of information as nodes in the graph. The edges between nodes are determined based on the overlap observed between adjacent sequences. Each node in the graph corresponds to a unique sequence of a specific length (n), while the edges capture the relationships and connections between these sequences. The nodes are connected to another if they share $n-1$ consecutive features. In our graph representation, the dihedrals serve as the vertices and are expressed as a sequential arrangement of atom identifiers. The graph's edges connect two dihedrals when there is an overlap of three atoms between them as shown in figure 5.2. There may be a desire to establish connections between dihedrals that have a smaller overlap of only 2 atoms. However, in order to achieve this, the introduction of edge weights would be necessary, which is currently avoided. This decision is made to maintain simplicity in the model design. Figure 5.3 shows the resulting overlap graph.

Sin Cos Decomposition

Understanding dihedral periodicity is essential for investigating molecular conformation and flexibility. Periodicity in dihedrals refers to the recurrent pattern

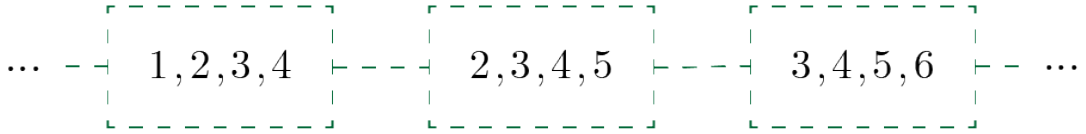


Figure 5.2: **Overlap graph** example.

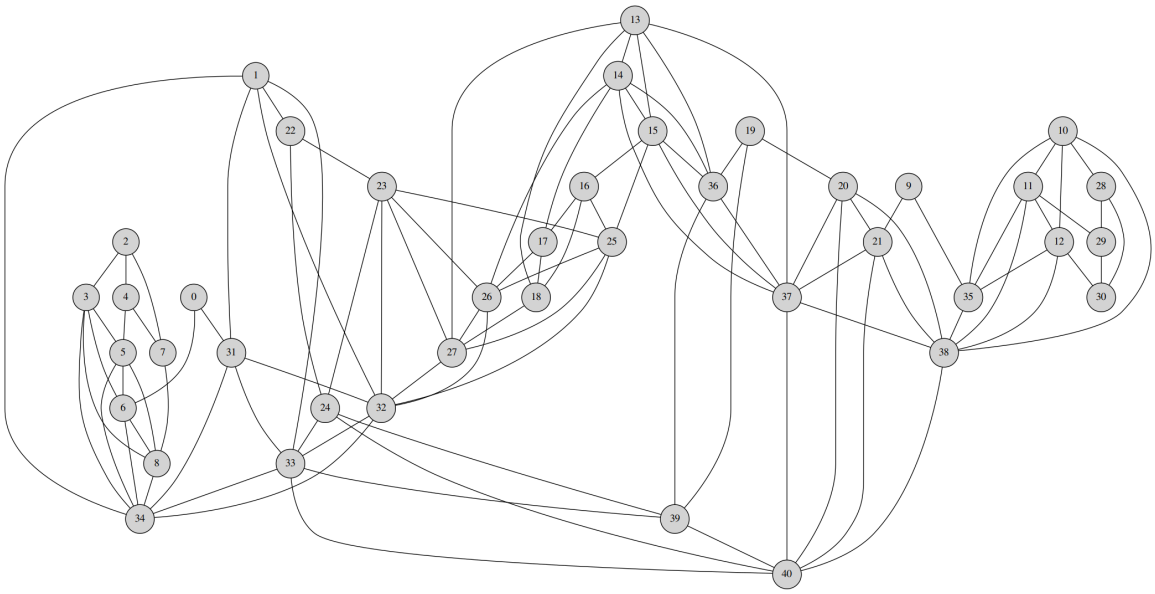


Figure 5.3: **Unweighted overlap graph** of Alanine Dipeptide.

exhibited by dihedral angles, which describe the spatial orientation of four atoms. It arises from the inherent rotational symmetry of molecules around the central bond. The dihedral angle data that we have from MD simulation ranges from -180 to 180 in degrees. The graph neural network does not know this concept inherently, and we need a way to teach the network about periodicity and we need a representation that maintains a direct correlation between slight changes in angle values and corresponding small shifts in the represented values. To address this, we employ the sin

and cos angle decomposition method. By utilizing the sine and cosine components of each angle, we encode the angle using these two values. The advantage of this approach is that both the sin and cos values undergo smooth variations in response to shifts in the angle. This ensures that even subtle adjustments in the angle values are reflected in the encoded representation. For example, since degrees lie on a plain number line domain, -180 degrees and $+180$ degrees lie at the complete opposite end of the number line. But if we encode these values with sin/cos decomposition, they will lie in the domain = $(-1^{-}, 0^{+})$ which are very close in their respective domain (circle with radius 1 and centre in (0,0)).

5.1.3 Alanine Dipeptide

The experiments conducted in this thesis are done on alanine dipeptide, which is used as the benchmark molecule. Alanine dipeptide [168] is a relatively uncomplicated system that has been extensively investigated in existing literature. It consists of a sequence comprising one amino acid and two terminal groups (ACE-ALA-NME). The selection of alanine dipeptide as the reference molecule is justified by the abundant literature data available on its Free Energy Surface (FES) and its significance as an initial stage for methodologies aimed at computing FESs for more intricate systems. The FES of Alanine Dipeptide is commonly characterized by the interplay of its two central backbone dihedrals, denoted as ϕ and ψ, as these dihedrals are recognized as slow degrees of freedom within this system (Figure 5.4). The FES exhibits two distinct energy minima, specifically referred to as C7eq and Cax, which are positioned at the coordinates (-1.45, +1.30) and (+1.22, -1.22) radians, respectively [169]. These two

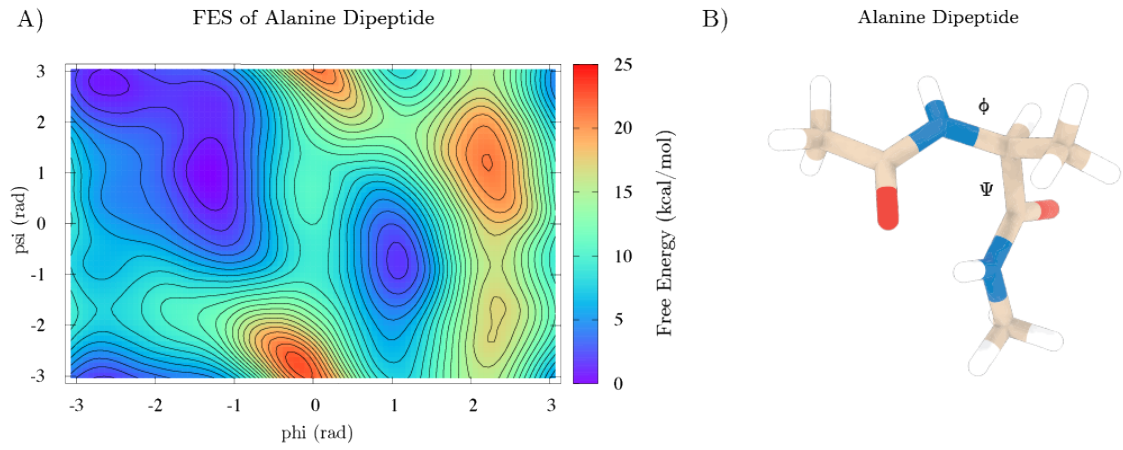


Figure 5.4: **Alanine Dipeptide**. A) The free energy surface of alanine dipeptide with its CVs in the x and y axes, and B) a sample conformation of alanine dipeptide along with its dihedral angles [139].

metastable states are separated by an energetic barrier of approximately 8 kcal/mol.

The advantage of investigating the Free Energy Surface (FES) of Alanine Dipeptide lies in the fact that the dihedrals alone offer a comprehensive depiction of the FES, making them ideal collective variables (CVs). However, in more intricate systems, relying solely on dihedral values may not provide a complete understanding of their FES. In such scenarios, alternative CVs need to be selected in order to fully explain the FES of these systems.

5.2 Autoencoder Results and Discussion

As discussed before, we train two models for the AE with different graph convolution operation: SAGEConv and TransformerConv. We will now compare and explain the results of the training. The training graph for SAGEConv model has been shown in figure 5.5 shows the training graph for AE model based on SAGEConv. Also, the

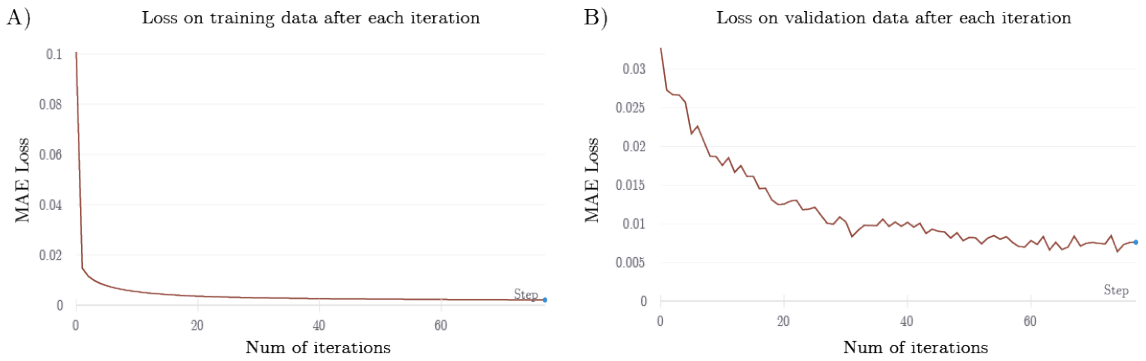


Figure 5.5: **SAGEConv based AE training**

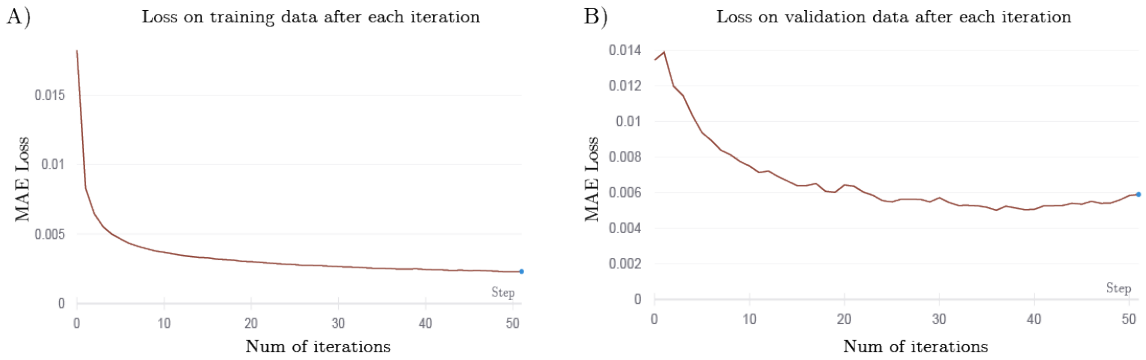


Figure 5.6: **TransformerConv based AE training**

training graph for TransformerConv model has been shown in figure 5.6 shows the training graph for AE model based on TransformerConv.

Since, we run the training with early stopping, the two models stop after certain but different iterations (epochs) over the data. SAGEConv based AE model runs for 77 epochs whereas the TransformerConv based model runs for 51 epochs. Although the SAGEconv runs for more epochs, the overall run time is almost twice as fast compared to the TransformerConv based model (see table 5.2). Although the loss

	Train Loss	**Val Loss**	**Test Loss**	**# Epochs**	**Time**
SAGEConv AE	0.002127	0.00759	0.043170	77	**1h 52m**
Transformer AE	0.002303	0.00589	**0.000465**	51	3h 50m

Table 5.1: Training results for AE

on training dataset and validation dataset are comparable for the two models, the TransformerConv based model significantly outperforms the SAGEConv on loss over the markedly larger test dataset. Since TransformerConv operation uses attention mechanism for learning on the graph, the individual operations are much slower. But it also learns the underlying graph distribution more thoroughly. For this reason, we select TransformerConv to create the graph embeddings for proceeding to the next phases.

5.3 Normalizing Flow Results and Discussion

In this sections, we discuss the results from the normalizing flow model training. The NLL/JML loss after 30,000 iterations of training is -467.635. The training graph is shown in Figure 5.7. To better understand the results from the boltzmann generator training, we can take a look at the sampling of FES from the trained BG. Figure 5.8 shows the sampling from MD simulation side by side with sampling from the BG. We sample 50,000 configurations from the boltzmann generator and compare it with the 50,000 samples from our dataset. As evident from the figure, the sampling shows that the architecture is able to capture the high and low energy regions in the free energy surface of alanine dipeptide. If we chart the energies of the sampled configurations,

Figure 5.7: **BG training graph** showing the JML or negative log-likelihood loss over 30,000 iterations.

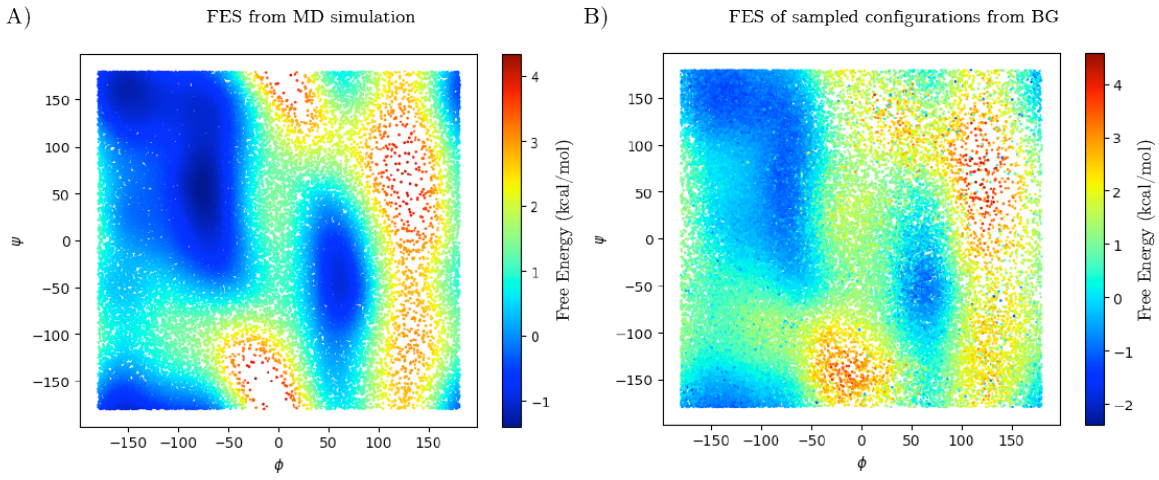


Figure 5.8: **Comparison of FES sampled from MD simulation vs BG** with 50,000 samples.

we can also see that the histogram of energies shows the same pattern. Figure 5.9 shows that the distribution of configuration energies closely overlap.

It should be noted that the sampled configurations from BG goes through the decoder first to get actual molecular graphs and then again through the free energy

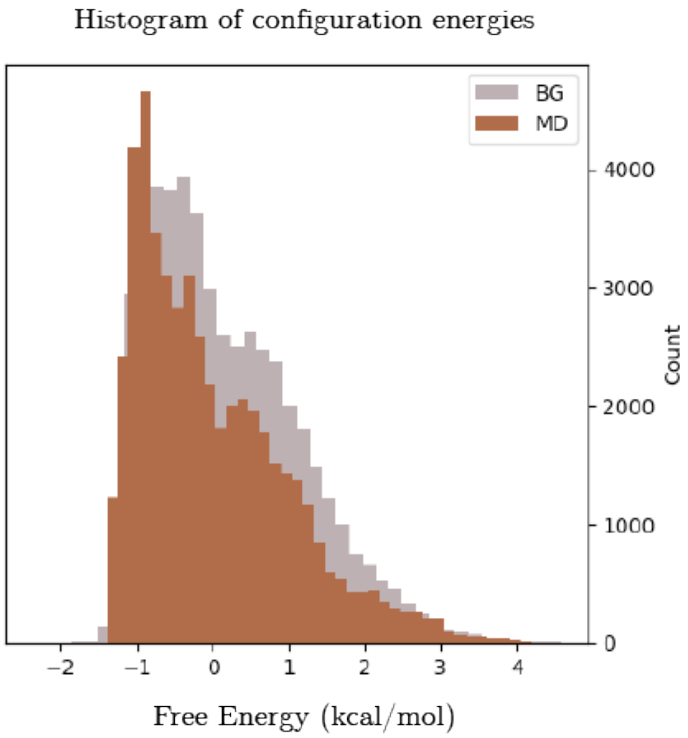


Figure 5.9: **Histogram of configuration energies** comparing BG sampling with MD sampling.

predictor model. These steps add small errors to the overall prediction pipeline.

5.4 Free Energy Predictor Model Results

We train the free energy predictor model for 111 iterations before early stopping kicks in. We get a training loss of 1.651, validation loss of 2.065 and test loss of 2.071. The training graph for the model is shows in figure 5.10.

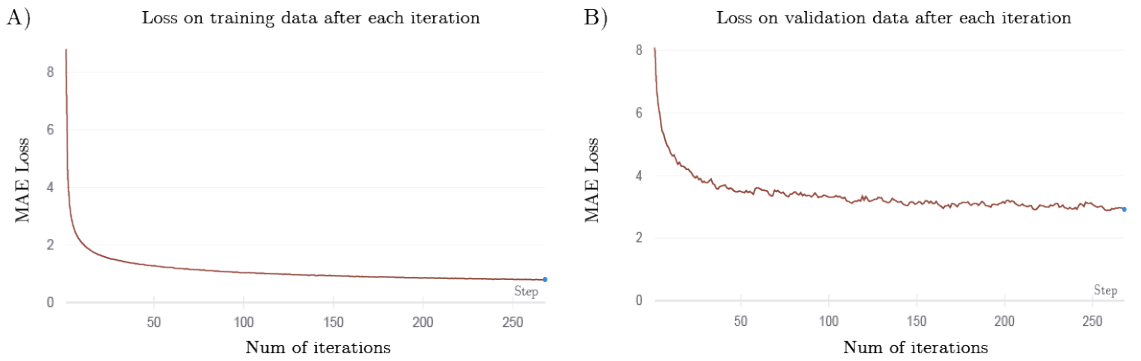


Figure 5.10: **FES prediction model** training graph.

5.5 Transition State Sampling and Validation

In this section, we are going to discuss about the process of sampling and validation of transition state candidates in the alanine dipeptide system. The details are in the following sections.

5.5.1 TS Sampling Workflow

In this section, we are going to discuss about the workflow related to sampling transition states from the trained pipeline. Similarly as before, we start with the graph embeddings correlated with every input molecular graph in our dataset, as shown in figure 5.11. These embeddings are generated by the encoder in our selected autoencoder model. Along with the embeddings, we also retain the free energy value of each molecule. We then use these free energy values and the ϕ-ψ angles to outline the minimas in the free energy surface of alanine dipeptide.

The minima region is outlined because we want a start and end point for the

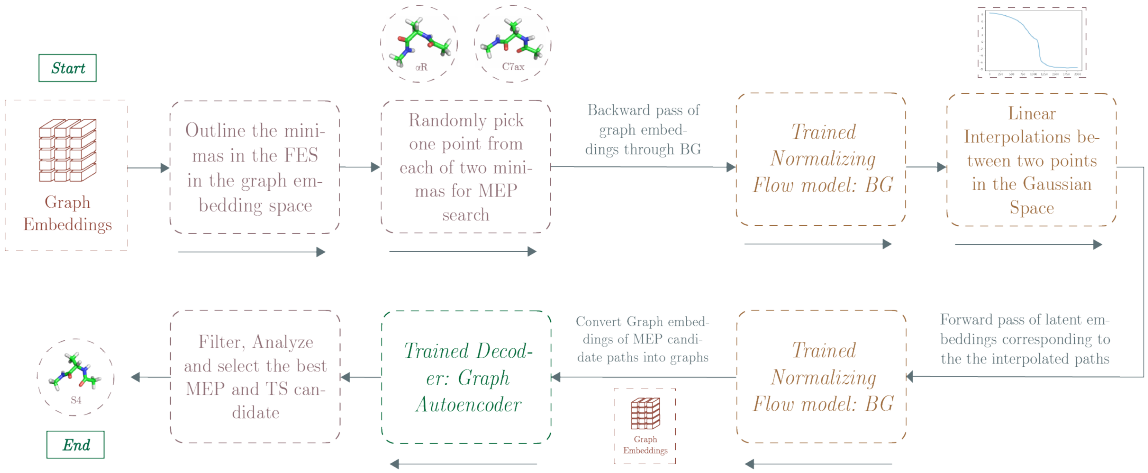


Figure 5.11: **Proposed Workflow of TS Sampling** in chemical systems. The process starts with graph embeddings and their corresponding free energy value and is passed through our trained neural network models to finally validation and resultant TS.

minimum energy pathway sampling. To outline the minima region, we first select the configurations (or embeddings) with a free energy value lower than 8.7407 Kcal/mol (FE<8.7407 Kcal/mol). Then we further filter the configurations with $70° \geq \phi \geq 40°$ and $0° \geq \psi \geq -90°$ for the C_{ax} minima and $-50° \geq \phi \geq -100°$ and $110° \geq \psi \geq 0°$ for the $C7_{eq}$ minima. This filtering process has been done visually, with the knowledge of the FES. But, since the selection is arbitrary and mostly dependent on the free energy value, it could be implemented on more complex systems as well. The resultant region is shown in figure 5.12.

We then sample 10 points randomly from each of the $C7_{eq}$ and the C_{ax} minima, which will be the starting and ending points for our potential minimum energy pathways (MEPs). These embeddings are then passed backwards through the already trained boltzmann generator. The trained boltzmann generator translates

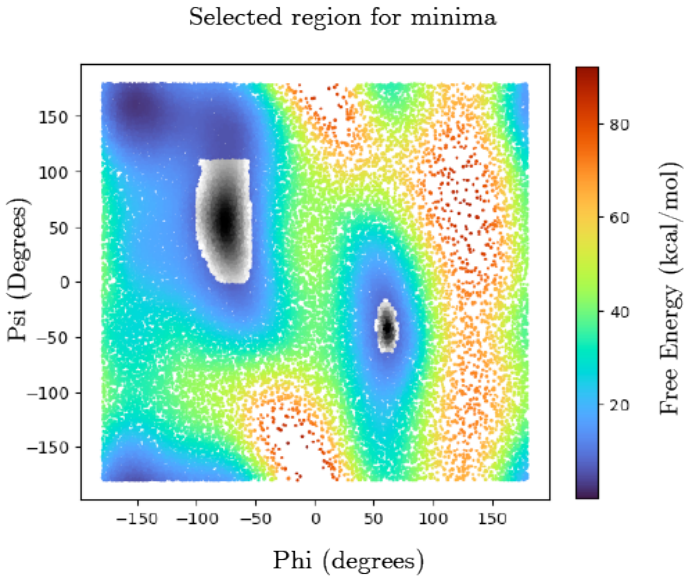


Figure 5.12: **Selected region for minima bounding** (gray region) on alanine dipeptide free energy surface

these graph embeddings onto points in the gaussian distribution. Once we are in the gaussian space, we do simple linear interpolation between the starting and ending points. 2,000 points were selected between the start and end point for the interpolation. The boltzmann generator allows us to easily do this interpolation which would be non-trivial in the original distribution space. These MEPs in the gaussian space are then forward passed through the boltzmann generator again to revert them back to the graph embedding space. From the embedding space, the embeddings are fed through the decoder to generate molecular graph representations. At this stage, we also need the free energy values for each point in the sampled MEPs. Since the 2,000 points in each MEP were interpolated in the gaussian space, the resultant graph representations corresponding to each point in the MEPs could be unique. For this reason, we use the free energy predictor model to predict the free energy values. With

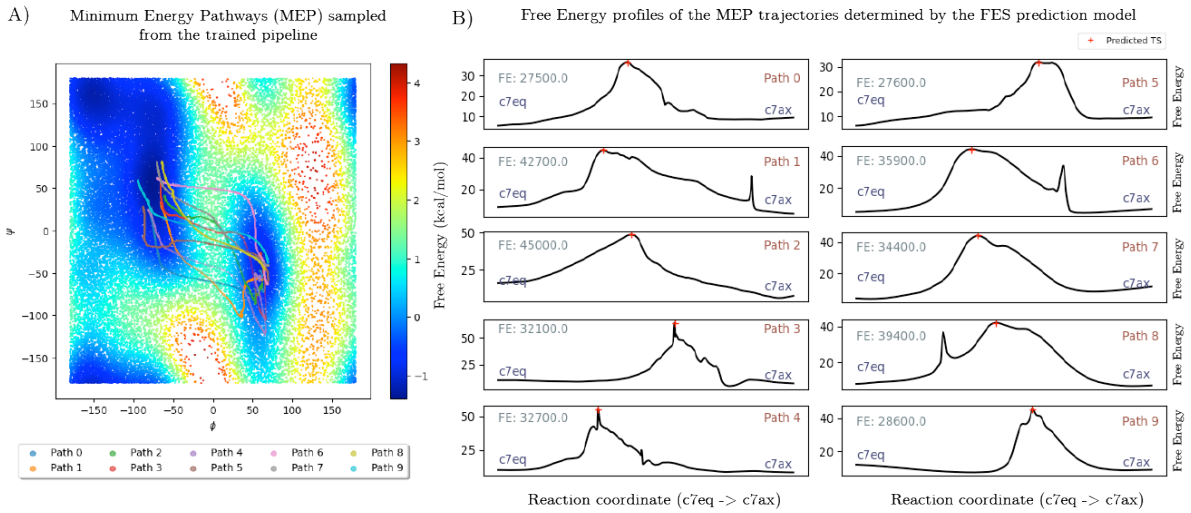


Figure 5.13: **Candidate MEPs.** A) Shows the 10 MEPs sampled from the BG overlaid on the FES of alanine dipeptide, and B) shows the free energy profile of each MEP trajectory determined by the FES predictor model along with the information about total free energy and where the transition state resides.

all of this information, we now have to filter, and select the best MEP candidates for further validation. We plot the sampled MEPs in figure 5.13.

5.5.2 Selection of TS Candidates

As the TS structure exists along the minimum energy pathway, we need to identify some promising MEPs. For this our criteria is very simple: select the path with the least cumulative energy. Using our energy predictor model, we find the energy of all the configuration along the MEP and sum them. In figure 5.13, this information for every MEP is put alongside the energy profile of each MEP with the title "FE". For example "Path 0" has a cumulative energy value of 27500.00 and "Path 1" has a cumulative energy value of 42700.00. We select the MEP with the least cumulative energy from here: Path 0. Now, we know that the TS structure is the highest

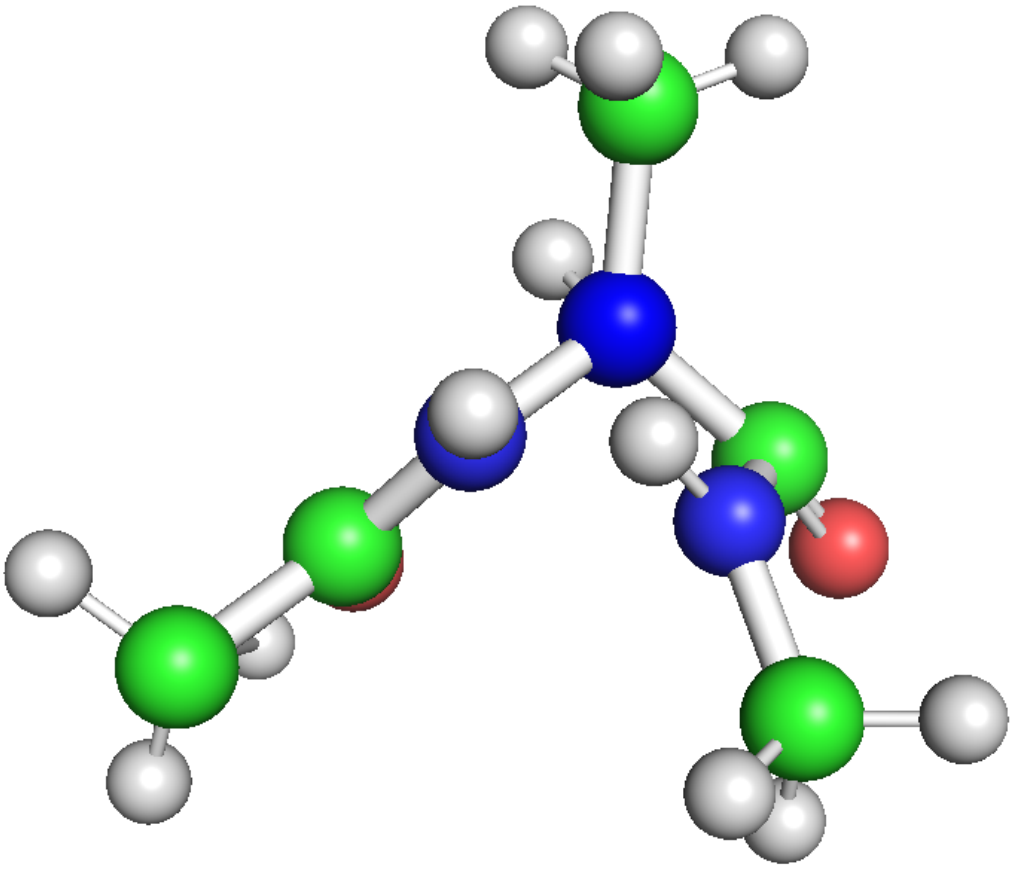
energy point along the MEP, so we identify the structure corresponding to the highest energy point along the TS. This structure from "Path 0" is named "TS0". With This candidate TS, we move on to committor analysis.

5.5.3 Validation of TS Candidates

The validation of the transition state involves committor analysis (see section 2.4.5. Our procedure for committor analysis is very simple, we start from a candidate transition state structure and launch a standard molecular dynamics simulation. We stop the simulation when the system reaches one of the minimas connected with that transition state. In the case of alanine dipeptide, where there are two well defined minima separated by a barrier, the simulation is halted if the system approaches either of these two minimas. With committor analysis, we evaluate "TS0" candidate.

"TS0" has a partition of 57 simulations in equatorial ($C7_{eq}$) vs 43 in axial ($C7_{ax}$), meaning this is essentially a transition state structure. We ran 100 different simulations to get a better statistical understanding. The resultant TS structure is demonstrated in figure 5.14.

The figure 5.15 shows the MEP on the FES and the energy profile corresponding to the TS0 structure. The transition state lies on a very high energy region on the FES. The precise coordinate of the transition state is: $\phi = 1.2608°$ and $\psi = -60.6218°$. The free energy barrier for the transition is 36.4836 Kcal/mol. The total free energy of the MEP as calculated by the energy predictor model is 27500.00 Kcal/mol.



Backbone of TS strcuture corresponding to TS0

Figure 5.14: **Resultant TS0 Structure**

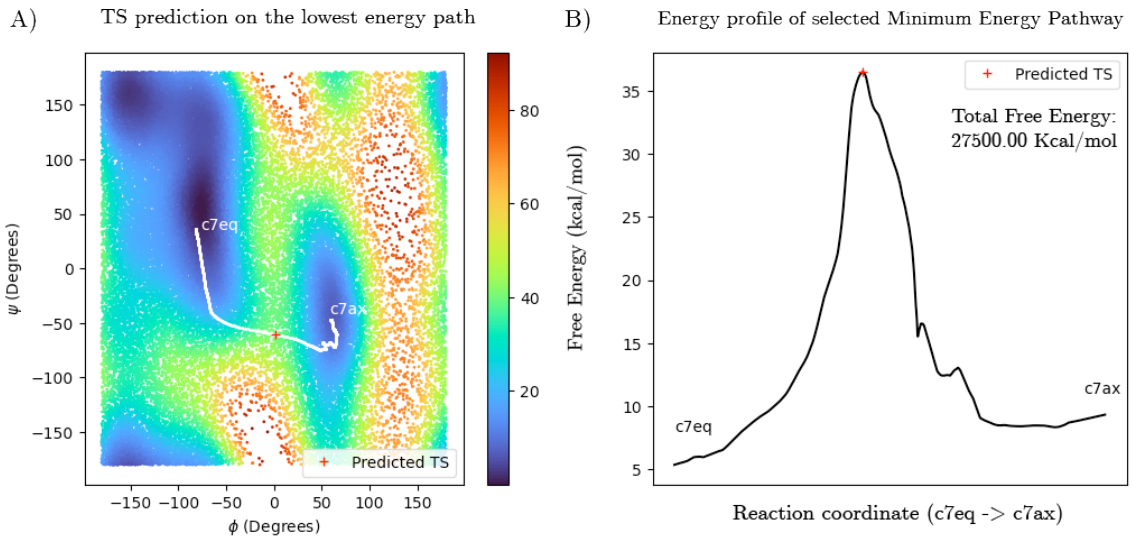


Figure 5.15: **TS0 MEP and energy profile.** A) Shows the minimum energy pathway corresponding to the TS0 structure along with the predicted TS location on the MEP, B) shows the energy profile for the MEP.

Chapter 6

Conclusion

6.1 Summary

Many critical problems in drug development, medicine, and materials science depend on creating particular molecular configuration with selected properties. Transition states are such molecular conformations that provide us knowledge of the chemical processes. The novel machine learning pipeline proposed in thesis is successfully able to sample conformational transition state in alanine-dipeptide. We propose a novel graph-autoencoder for the purpose of encoding our molecular representations. We then use the normalizing flow based boltzmann generator for learning the underlying free energy surface. The pipeline is able to predict viable minimum energy pathways only with data from MD simulations. Then, our proposed workflow efficiently samples TS from the trained pipeline in one-shot. Our novel ML model also works in a completely unsupervised manner, without needing access to any example TS conformations or reaction coordinates. This universality and flexibility trans-

lates to a framework that could tackle even more complex systems, potentially aiding researchers in many fields to accelerate their discoveries.

6.2 Future Work

One major direction for future work would be to incorporate Cartesian coordinates in the molecular graph representation. Cartesian coordinates offer a more natural way of representing molecules. By having cartesian coordinates in the node features, the graphs would very closely mimic the actual molecules. We could further incorporate the information about various forces (e.g. van-der-waals or coulomb) on the edges on the graphs. The challenge with this approach would be in handling the increased complexity in degrees of freedom. We would also need to find a suitable graph convolution operator that could work with such a representation.

Another direction of future work could be in incorporating an E(3) equivariant neural network for learning the representations [170]. These equivariant convolutions could lead to more rich representation of the molecular environment leading to better results in the downstream tasks. This framework has already shown promise and could make working with complex systems easier.

Bibliography

[1] N. Otterness, M. Yang, S. Rust, E. Park, J. H. Anderson, F. D. Smith, A. Berg, and S. Wang. "An Evaluation of the NVIDIA TX1 for Supporting Real-Time Computer-Vision Workloads". In: *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2017, pp. 353–364. DOI: 10.1109/RTAS.2017.3.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. "Spectral Networks and Locally Connected Networks on Graphs". In: *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014*. 2014.

[5] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. "Deep learning for computer vision: A brief review". In: *Computational intelligence and neuroscience* 2018 (2018).

[6] T. Young, D. Hazarika, S. Poria, and E. Cambria. "Recent trends in deep learning based natural language processing". In: *ieee Computational intelligenCe magazine* 13.3 (2018), pp. 55–75.

[7] J. Vamathevan, D. Clark, P. Czodrowski, I. Dunham, E. Ferran, G. Lee, B. Li, A. Madabhushi, P. Shah, and M. Spitzer. "Applications of machine learning in drug discovery and development". In: *Nature Reviews Drug Discovery* 18.6 (2019), pp. 463–477.

[8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, and A. Potapenko. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[9] E. N. Feinberg, D. Sur, Z. Wu, B. E. Husic, H. Mai, Y. Li, S. Sun, J. Yang, B. Ramsundar, and V. S. Pande. "PotentialNet for molecular property prediction". In: *ACS central science* 4.11 (2018), pp. 1520–1530.

[10] L. D. Jacobson, A. D. Bochevarov, M. A. Watson, T. F. Hughes, D. Rinaldo, S. Ehrlich, T. B. Steinbrecher, S Vaitheeswaran, D. M. Philipp, M. D. Halls, et al. "Automated transition state search and its application to diverse types of organic reactions". In: *Journal of chemical theory and computation* 13.11 (2017), pp. 5780–5797.

[11] A. R. Fersht. "Characterizing transition states in protein folding: an essential step in the puzzle". In: *Current opinion in structural biology* 5.1 (1995), pp. 79–84.

[12] G. F. von Rudorff, S. N. Heinen, M. Bragato, and O. A. von Lilienfeld. "Thousands of reactants and transition states for competing E2 and S2 reactions". In: *Machine Learning: Science and Technology* 1.4 (2020), p. 045026.

[13] T. N. Doman, S. L. McGovern, B. J. Witherbee, T. P. Kasten, R. Kurumbail, W. C. Stallings, D. T. Connolly, and B. K. Shoichet. "Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1B". In: *Journal of medicinal chemistry* 45.11 (2002), pp. 2213–2221.

[14] V. L. Schramm. "Transition states, analogues, and drug development". In: *ACS chemical biology* 8.1 (2013), pp. 71–81.

[15] H. Jónsson, G. Mills, and K. W. Jacobsen. "Nudged elastic band method for finding minimum energy paths of transitions". In: (1998).

[16] G. Henkelman, B. P. Uberuaga, and H. Jónsson. "A climbing image nudged elastic band method for finding saddle points and minimum energy paths". In: *The Journal of chemical physics* 113.22 (2000), pp. 9901–9904.

[17] G.-R. Qian, X. Dong, X.-F. Zhou, Y. Tian, A. R. Oganov, and H.-T. Wang. "Variable cell nudged elastic band method for studying solid–solid structural phase transitions". In: *Computer Physics Communications* 184.9 (2013), pp. 2111–2118.

[18] D. Lemm, G. F. von Rudorff, and O. A. von Lilienfeld. "Machine learning based energy-free structure predictions of molecules, transition states, and solids". In: *Nature Communications* 12.1 (2021), pp. 1–10.

[19] A. F. Voter, F. Montalenti, and T. C. Germann. "Extending the time scale in atomistic simulation of materials". In: *Annual Review of Materials Research* 32.1 (2002), pp. 321–346.

[20] F. Noé, S. Olsson, J. Köhler, and H. Wu. "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning". In: *Science* 365.6457 (2019), eaaw1147.

[21] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[22] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton. "Backpropagation and the brain". In: *Nature Reviews Neuroscience* 21.6 (2020), pp. 335–346.

[23] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[25] P. J. Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[26] M. Wu and L. Chen. "Image recognition based on deep learning". In: *2015 Chinese automation congress (CAC)*. IEEE. 2015, pp. 542–546.

[27] I. Lauriola, A. Lavelli, and F. Aiolli. "An introduction to deep learning in natural language processing: Models, techniques, and tools". In: *Neurocomputing* 470 (2022), pp. 443–456.

[28] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[29] D. Hendrycks and K. Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[30] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[31] I. Kobyzev, S. J. Prince, and M. A. Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3964–3979.

[32] P. Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 37–49.

[33] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[35] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[37] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. "Weisfeiler and leman go neural: Higher-order graph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

[38] W. L. Hamilton, R. Ying, and J. Leskovec. "Inductive representation learning on large graphs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1025–1035.

[39] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. "Masked label prediction: Unified message passing model for semi-supervised classification". In: *arXiv preprint arXiv:2009.03509* (2020).

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[41] M. Fey and J. E. Lenssen. "Fast graph representation learning with PyTorch Geometric". In: *arXiv preprint arXiv:1903.02428* (2019).

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[43] A. Jamasb, R. Viñas Torné, E. Ma, Y. Du, C. Harris, K. Huang, D. Hall, P. Lió, and T. Blundell. "Graphein-a python library for geometric deep learning and network analysis on biomolecular structures and interaction networks". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27153–27167.

[44] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, G. López, N. Collignon, et al. "Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 4564–4573.

[45] A. Kukol. *Molecular Modeling of Proteins*. Vol. 1215. Springer, 2015.

[46] D. W. Borhani and D. E. Shaw. "The future of molecular dynamics simulations in drug discovery". In: *Journal of computer-aided molecular design* 26 (2012), pp. 15–26.

[47] R. Iftimie, P. Minary, and M. E. Tuckerman. "Ab initio molecular dynamics: Concepts, recent developments, and future trends". In: *Proceedings of the National Academy of Sciences* 102.19 (2005), pp. 6654–6659.

[48] J. W. Gibbs. "A method of geometrical representation of the thermodynamic properties by means of surfaces". In: *The Collected Works of J. Willard Gibbs, Ph. D., LL. D* (1957), pp. 33–54.

[49] M. de Koning, W. Cai, A. Antonelli, and S. Yip. "Efficient free-energy calculations by the simulation of nonequilibrium processes". In: *Computing in Science & Engineering* 2.3 (2000), pp. 88–96.

[50] S. Marsili, G. F. Signorini, R. Chelli, M. Marchi, and P. Procacci. "Orac: A molecular dynamics simulation program to explore free energy surfaces in biomolecular systems at the atomistic level". In: *Journal of computational chemistry* 31.5 (2010), pp. 1106–1116.

[51] K. J. Laidler. "Just what is a transition state?" In: *Journal of chemical Education* 65.6 (1988), p. 540.

[52] H. Eyring. "The activated complex in chemical reactions". In: *The Journal of Chemical Physics* 3.2 (1935), pp. 107–115.

[53] M. G. Evans and M. Polanyi. "Some applications of the transition state method to the calculation of reaction velocities, especially in solution". In: *Transactions of the Faraday Society* 31 (1935), pp. 875–894.

[54] T. Edwards, T. Endo, J. H. Walton, and S. Sen. "Observation of the transition state for pressure-induced BO3→ BO4 conversion in glass". In: *Science* 345.6200 (2014), pp. 1027–1029.

[55] J. L. Durant. "Evaluation of transition state properties by density functional theory". In: *Chemical physics letters* 256.6 (1996), pp. 595–602.

[56] A. Komornicki, K. Ishida, K. Morokuma, R. Ditchfield, and M. Conrad. "Efficient determination and characterization of transition states using ab-initio methods". In: *Chemical Physics Letters* 45.3 (1977), pp. 595–602.

[57] M. Rico-Pasto, A. Zaltron, S. J. Davis, S. Frutos, and F. Ritort. "Molten globule–like transition state of protein barnase measured with calorimetric force spectroscopy". In: *Proceedings of the National Academy of Sciences* 119.11 (2022), e2112382119.

[58] A. Laio and M. Parrinello. "Escaping free-energy minima". In: *Proceedings of the national academy of sciences* 99.20 (2002), pp. 12562–12566.

[59] A. Barducci, G. Bussi, and M. Parrinello. "Well-tempered metadynamics: a smoothly converging and tunable free-energy method". In: *Physical review letters* 100.2 (2008), p. 020603.

[60] P. Tiwary, V. Limongelli, M. Salvalaglio, and M. Parrinello. "Kinetics of protein–ligand unbinding: Predicting pathways, rates, and rate-limiting steps". In: *Proceedings of the National Academy of Sciences* 112.5 (2015), E386–E391.

[61] P. Cui, X. Wang, J. Pei, and W. Zhu. "A survey on network embedding". In: *IEEE transactions on knowledge and data engineering* 31.5 (2018), pp. 833–852.

[62] P. Goyal and E. Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94.

[63] H. Cai, V. W. Zheng, and K. C.-C. Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9 (2018), pp. 1616–1637.

[64] D. Zhang, J. Yin, X. Zhu, and C. Zhang. "Network representation learning: A survey". In: *IEEE transactions on Big Data* 6.1 (2018), pp. 3–28.

[65] B. Perozzi, R. Al-Rfou, and S. Skiena. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[66] J. Gasteiger, J. Groß, and S. Günnemann. "Directional message passing for molecular graphs". In: *arXiv preprint arXiv:2003.03123* (2020).

[67] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[68] M. Henaff, J. Bruna, and Y. LeCun. "Deep Convolutional Networks on Graph-Structured Data". In: *arXiv* (2015). eprint: `1506.05163`.

[69] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems* 29 (2016).

[70] D. K. Hammond, P. Vandergheynst, and R. Gribonval. "Wavelets on graphs via spectral graph theory". In: *Applied and Computational Harmonic Analysis* 30.2 (2011), pp. 129–150.

[71] J. Chen, T. Ma, and C. Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling". In: *arXiv preprint arXiv:1801.10247* (2018).

[72] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. "Convolutional networks on graphs for learning molecular fingerprints". In: *Advances in neural information processing systems* 28 (2015).

[73] J. Atwood and D. Towsley. "Diffusion-convolutional neural networks". In: *Advances in neural information processing systems* 29 (2016).

[74] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[75] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[76] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. "A convolutional encoder model for neural machine translation". In: *arXiv preprint arXiv:1611.02344* (2016).

[77] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018.

[78] Z. Xiong, D. Wang, X. Liu, F. Zhong, X. Wan, X. Li, Z. Li, X. Luo, K. Chen, H. Jiang, et al. "Pushing the boundaries of molecular representation for

drug discovery with the graph attention mechanism". In: *Journal of medicinal chemistry* 63.16 (2019), pp. 8749–8760.

[79] R OpenAI. "GPT-4 technical report". In: *arXiv* (2023).

[80] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec. "Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation". In: ().

[81] J. Palowitch and B. Perozzi. {*MONET*}*: Debiasing Graph Embeddings via the Metadata-Orthogonal Training Unit*. 2020.

[82] K. Atz, F. Grisoni, and G. Schneider. "Geometric deep learning on molecular representations". In: *Nature Machine Intelligence* 3.12 (2021), pp. 1023–1032.

[83] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko. "Quantum-chemical insights from deep tensor neural networks". In: *Nature communications* 8.1 (2017), p. 13890.

[84] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, et al. "A deep learning approach to antibiotic discovery". In: *Cell* 180.4 (2020), pp. 688–702.

[85] W. Torng and R. B. Altman. "Graph convolutional neural networks for predicting drug-target interactions". In: *Journal of chemical information and modeling* 59.10 (2019), pp. 4131–4149.

[86] J. Li, D. Cai, and X. He. *Learning Graph-Level Representation for Drug Discovery*. 2017. arXiv: `1709.03741 [cs.LG]`.

[87] K. Liu, X. Sun, L. Jia, J. Ma, H. Xing, J. Wu, H. Gao, Y. Sun, F. Boulnois, and J. Fan. "Chemi-Net: a molecular graph convolutional network for accurate drug property prediction". In: *International journal of molecular sciences* 20.14 (2019), p. 3389.

[88] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[89] T. N. Kipf and M. Welling. "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (2016).

[90] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. "Adversarially regularized graph autoencoder for graph embedding". In: *arXiv preprint arXiv:1802.04407* (2018).

[91] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang. "Mgae: Marginalized graph autoencoder for graph clustering". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 889–898.

[92] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang. "One2multi graph autoencoder for multi-view graph clustering". In: *proceedings of the web conference 2020*. 2020, pp. 3070–3076.

[93] W. Jin, R. Barzilay, and T. Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *International conference on machine learning*. PMLR. 2018, pp. 2323–2332.

[94] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. "Constrained graph variational autoencoders for molecule design". In: *Advances in neural information processing systems* 31 (2018).

[95] J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim. "Molecular generative model based on conditional variational autoencoder for de novo molecular design". In: *Journal of cheminformatics* 10.1 (2018), pp. 1–9.

[96] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[97] E. G. Tabak and E. Vanden-Eijnden. "Density estimation by dual ascent of the log-likelihood". In: *Communications in Mathematical Sciences* 8.1 (2010), pp. 217–233.

[98] E. G. Tabak and C. V. Turner. "A family of nonparametric density estimation algorithms". In: *Communications on Pure and Applied Mathematics* 66.2 (2013), pp. 145–164.

[99] D. Rezende and S. Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[100] L. Dinh, D. Krueger, and Y. Bengio. "Nice: Non-linear independent components estimation". In: *arXiv preprint arXiv:1410.8516* (2014).

[101] O. Rippel and R. P. Adams. "High-dimensional probability estimation with deep density models". In: *arXiv preprint arXiv:1302.5125* (2013).

[102] V. Laparra, G. Camps-Valls, and J. Malo. "Iterative gaussianization: from ICA to random rotations". In: *IEEE transactions on neural networks* 22.4 (2011), pp. 537–549.

[103] D. P. Kingma and P. Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems* 31 (2018).

[104] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems* 29 (2016).

[105] L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803* (2016).

[106] C. Ma and X. Zhang. "GF-VAE: A Flow-based Variational Autoencoder for Molecule Generation". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1181–1190.

[107] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang. "Graphaf: a flow-based autoregressive model for molecular graph generation". In: *arXiv preprint arXiv:2001.09382* (2020).

[108] C. Zang and F. Wang. "MoFlow: an invertible flow model for generating molecular graphs". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 617–626.

[109] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky. "Graph normalizing flows". In: *Advances in Neural Information Processing Systems* 32 (2019).

[110] Q. Vanhaelen, Y.-C. Lin, and A. Zhavoronkov. "The advent of generative chemistry". In: *ACS Medicinal Chemistry Letters* 11.8 (2020), pp. 1496–1505.

[111] M. A. Skinnider, R. G. Stacey, D. S. Wishart, and L. J. Foster. "Deep generative models enable navigation in sparsely populated chemical space". In: (2021).

[112] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276.

[113] R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noé, and D.-A. Clevert. "Efficient multi-objective molecular optimization in a continuous latent space". In: *Chemical science* 10.34 (2019), pp. 8016–8024.

[114] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller. "Generating focused molecule libraries for drug discovery with recurrent neural networks". In: *ACS central science* 4.1 (2018), pp. 120–131.

[115] A. Gupta, A. T. Müller, B. J. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider. "Generative recurrent networks for de novo drug design". In: *Molecular informatics* 37.1-2 (2018), p. 1700111.

[116] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. "Molecular de-novo design through deep reinforcement learning". In: *Journal of cheminformatics* 9.1 (2017), pp. 1–14.

[117] X. Liu, K. Ye, H. W. Van Vlijmen, A. P. IJzerman, and G. J. Van Westen. "An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor". In: *Journal of cheminformatics* 11.1 (2019), p. 35.

[118] E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik, and A. Zhavoronkov. "Reinforced adversarial neural computer for de novo molecular design". In: *Journal of chemical information and modeling* 58.6 (2018), pp. 1194–1204.

[119] O. Prykhodko, S. V. Johansson, P.-C. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist, and H. Chen. "A de novo molecular generation method using latent vector based generative adversarial network". In: *Journal of Cheminformatics* 11.1 (2019), pp. 1–13.

[120] M. Kuznetsov and D. Polykovskiy. "MolGrow: A graph normalizing flow for hierarchical molecular generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 9. 2021, pp. 8226–8234.

[121] N. C. Frey, V. Gadepally, and B. Ramsundar. *FastFlows: Flow-Based Models for Molecular Graph Generation*. 2022. arXiv: `2201.12419 [physics.chem-ph]`.

[122] E. Rozenberg, E. Rivlin, and D. Freedman. "Structure-Based Drug Design via Semi-Equivariant Conditional Normalizing Flows". In: *ICLR 2023-Machine Learning for Drug Discovery workshop*. 2023.

[123] R. Jinnouchi, F. Karsai, C. Verdi, R. Asahi, and G. Kresse. "Descriptors representing two-and three-body atomic distributions and their effects on the accu-

racy of machine-learned inter-atomic potentials". In: *The Journal of Chemical Physics* 152.23 (2020), p. 234102.

[124] K. Low, R. Kobayashi, and E. I. Izgorodina. "The effect of descriptor choice in machine learning models for ionic liquid melting point prediction". In: *The Journal of Chemical Physics* 153.10 (2020), p. 104101.

[125] L Martin-Gondre, C Crespos, P Larrégaray, J. Rayez, D Conte, and B van Ootegem. "Detailed description of the flexible periodic London–Eyring–Polanyi–Sato potential energy function". In: *Chemical Physics* 367.2-3 (2010), pp. 136–147.

[126] L Martin-Gondre, C Crespos, P Larrégaray, J. Rayez, B van Ootegem, and D Conte. "Dynamics simulation of N 2 scattering onto W (100,110) surfaces: A stringent test for the recently developed flexible periodic London–Eyring–Polanyi–Sato potential energy surface". In: *The Journal of chemical physics* 132.20 (2010), p. 204501.

[127] W. Kohn and L. J. Sham. "Self-consistent equations including exchange and correlation effects". In: *Physical review* 140.4A (1965), A1133.

[128] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren. "Neural network models of potential energy surfaces". In: *The Journal of chemical physics* 103.10 (1995), pp. 4129–4137.

[129] S. Lorenz, A. Groß, and M. Scheffler. "Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks". In: *Chemical Physics Letters* 395.4-6 (2004), pp. 210–215.

[130] J. Behler and M. Parrinello. "Generalized neural-network representation of high-dimensional potential-energy surfaces". In: *Physical review letters* 98.14 (2007), p. 146401.

[131] J. Behler, R. Martoňák, D. Donadio, and M. Parrinello. "Metadynamics simulations of the high-pressure phases of silicon employing a high-dimensional neural network potential". In: *Physical review letters* 100.18 (2008), p. 185501.

[132] J. S. Smith, O. Isayev, and A. E. Roitberg. "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost". In: *Chemical science* 8.4 (2017), pp. 3192–3203.

[133] S. Faraji, S. A. Ghasemi, S. Rostami, R. Rasoulkhani, B. Schaefer, S. Goedecker, and M. Amsler. "High accuracy and transferability of a neural network potential through charge equilibration for calcium fluoride". In: *Physical Review B* 95.10 (2017), p. 104105.

[134] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in neural information processing systems* 30 (2017).

[135] O. T. Unke and M. Meuwly. "PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges". In: *Journal of chemical theory and computation* 15.6 (2019), pp. 3678–3693.

[136] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi. "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons". In: *Physical review letters* 104.13 (2010), p. 136403.

[137] A. Glielmo, P. Sollich, and A. De Vita. "Accurate interatomic force fields via machine learning with covariant kernels". In: *Physical Review B* 95.21 (2017), p. 214302.

[138] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller. "Machine learning of accurate energy-conserving molecular force fields". In: *Science advances* 3.5 (2017), e1603015.

[139] S. Heydari, S. Raniolo, L. Livi, and V. Limongelli. "Transferring chemical and energetic knowledge between molecular systems with machine learning". In: *Communications Chemistry* 6.1 (2023), p. 13.

[140] P. Pechukas. "Transition state theory". In: *Annual Review of Physical Chemistry* 32.1 (1981), pp. 159–177.

[141] X. Liu, H. Chen, and C. Ortner. "Stability of the Minimum Energy Path". In: *arXiv preprint arXiv:2204.00984* (2022).

[142] N. R. Mathiesen, H. Jonsson, T. Vegge, and J. M. Garcia Lastra. "R-neb: Accelerated nudged elastic band calculations by use of reflection symmetry". In: *Journal of chemical theory and computation* 15.5 (2019), pp. 3215–3222.

[143] A. A. Peterson. "Acceleration of saddle-point searches with machine learning". In: *The Journal of chemical physics* 145.7 (2016), p. 074106.

[144] A. Denzel, B. Haasdonk, and J. Kastner. "Gaussian Process Regression for Minimum Energy Path Optimization and Transition State Search". In: *The Journal of Physical Chemistry A* 123.44 (2019), pp. 9600–9611.

[145] E Weinan, W. Ren, and E. Vanden-Eijnden. "String method for the study of rare events". In: *Physical Review B* 66.5 (2002), p. 052301.

[146] E Weinan, W. Ren, and E. Vanden-Eijnden. "Simplified and improved string method for computing the minimum energy paths in barrier-crossing events". In: *Journal of Chemical Physics* 126.16 (2007), p. 164103.

[147] W. Ren and E. Vanden-Eijnden. "A climbing string method for saddle point search". In: *The Journal of chemical physics* 138.13 (2013), p. 134105.

[148] A. E. Ulanov, E. S. Tiunov, and A. Lvovsky. "Quantum-inspired annealers as Boltzmann generators for machine learning and statistical physics". In: *arXiv preprint arXiv:1912.08480* (2019).

[149] T. Liu, W. Gao, Z. Wang, and C. Wang. "PathFlow: A normalizing flow generator that finds transition paths". In: *Uncertainty in Artificial Intelligence*. PMLR. 2022, pp. 1232–1242.

[150] H. B. Schlegel. "Geometry optimization on potential energy surfaces". In: *Modern Electronic Structure Theory: Part I*. World Scientific, 1995, pp. 459–500.

[151] H. B. Schlegel. "Geometry optimization". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1.5 (2011), pp. 790–809.

[152] D. Sheppard, R. Terrell, and G. Henkelman. "Optimization methods for finding minimum energy paths". In: *The Journal of chemical physics* 128.13 (2008), p. 134106.

[153] G. S. Hammond. "A correlation of reaction rates". In: *Journal of the American Chemical Society* 77.2 (1955), pp. 334–338.

[154] P. M. Zimmerman. "Single-ended transition state finding with the growing string method". In: *Journal of computational chemistry* 36.9 (2015), pp. 601–611.

[155] P. L. Bhoorasingh and R. H. West. "Transition state geometry prediction using molecular group contributions". In: *Physical Chemistry Chemical Physics* 17.48 (2015), pp. 32173–32182.

[156] C. W. Gao, J. W. Allen, W. H. Green, and R. H. West. "Reaction Mechanism Generator: Automatic construction of chemical kinetic mechanisms". In: *Computer Physics Communications* 203 (2016), pp. 212–225.

[157] M. Z. Makoś, N. Verma, E. C. Larson, M. Freindorf, and E. Kraka. "Generative adversarial networks for transition state geometry prediction". In: *The Journal of Chemical Physics* 155.2 (2021), p. 024116.

[158] R. Jackson, W. Zhang, and J. Pearson. "TSNet: predicting transition state structures with tensor field networks and transfer learning". In: *Chemical Science* 12.29 (2021), pp. 10022–10040.

[159] L. Pattanaik, J. B. Ingraham, C. A. Grambow, and W. H. Green. "Generating transition states of isomerization reactions with deep learning". In: *Physical Chemistry Chemical Physics* 22.41 (2020), pp. 23618–23626.

[160] S. Choi. "Prediction of transition state structures of gas-phase chemical reactions via machine learning". In: *Nature Communications* 14.1 (2023), p. 1168.

[161] S. Kim, J. Woo, and W. Y. Kim. "Diffusion-based Generative AI for Exploring Transition States from 2D Molecular Graphs". In: (2023).

[162] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[163] D. Carrara. "Free-energy calculations using Graph Convolutional Networks". In: ().

[164] M. A. Skinnider, R. G. Stacey, D. S. Wishart, and L. J. Foster. "Chemical language models enable navigation in sparsely populated chemical space". In: *Nature Machine Intelligence* 3.9 (2021), pp. 759–770.

[165] A. Bankevich, A. V. Bzikadze, M. Kolmogorov, D. Antipov, and P. A. Pevzner. "Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads". In: *Nature biotechnology* 40.7 (2022), pp. 1075–1081.

[166] K. Dufault-Thompson and X. Jiang. "Applications of de Bruijn graphs in microbiome research". In: *iMeta* 1.1 (2022), e4.

[167] N. G. De Bruijn. "A combinatorial problem". In: *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 49.7 (1946), pp. 758–764.

[168] J. Apostolakis, P. Ferrara, and A. Caflisch. "Calculation of conformational transitions and barriers in solvated systems: Application to the alanine dipeptide in water". In: *The Journal of chemical physics* 110.4 (1999), pp. 2099–2108.

[169] P. Tiwary and M. Parrinello. "A time-independent free energy estimator for metadynamics". In: *The Journal of Physical Chemistry B* 119.3 (2015), pp. 736–742.

[170] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky. "E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials". In: *Nature communications* 13.1 (2022), p. 2453.