Machine Learning Enabled Prediction of Solvent-Based Reaction Energetics

by

Yiren Mao

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Process Control

Department of Chemical and Materials Engineering
University of Alberta

Abstract

Despite recent advancements in molecular dynamics (MD) methods, the computational costs of *ab initio* molecular dynamics simulations for explicit solvation systems are still too significant. If accuracy is to be left uncompromised, new methods must be employed to reduce computational expenses. This work focuses on the development of machine learning (ML) models as proxy models for Car-Parrinello molecular dynamics (CPMD) metadynamics simulations in condensed-phase biomass reactions.

Explicit solvation CPMD metadynamics simulation data of HMF undergoing protonation in a solution of dimethyl sulphoxide (DMSO) and water is used to train various model architectures to make time-series predictions of their probability distribution functions (PDFs). For each model architecture, three models were trained to fully predict the system, one for each of the following species: reactants, water and DMSO. Each model was tested assuming an initial simulation had been performed and the proxy models were used to complete the simulation.

The long short-term memory (LSTM) autoencoder and 3D convolutional neural networks (CNN)-LSTM autoencoder architectures failed to accurately capture PDF magnitudes and locations. A binary relevance 3D CNN-LSTM autoencoder, employing different loss functions, showed marginal improvement but struggled to predict probability locations over a large horizon. Models trained on principal component analysis (PCA)-transformed and dynamic PCA (DPCA)-transformed data showed promise in training but failed in testing. Models trained on PDFs without "dead voxels" (zero probability voxels independent of time)

and atomic Cartesian coordinates perform well during training but encounter challenges in testing due to teacher forcing. Teacher forcing is a training method that can potentially make the trained model over-reliant on ground truth, which is unavailable if the model is to be used as a proxy. Despite attempts to mitigate teacher forcing effects through scheduled sampling, no model architecture achieves reliable long-term predictions without ground truth data. However, the model trained on Cartesian coordinates demonstrated proficiency in making short-term predictions regarding the atomic configuration of the system.

Preface

This thesis is an original work by Yiren (Kevin) Mao. No part of this thesis has been previously published.

Quitters never win. Winners never quit.

Acknowledgements

I would like to first thank my academic supervisors, Dr. Vinay Prasad and Dr. Samir H. Mushrif, for giving me this incredible opportunity to further my knowledge and push the boundaries of my capabilities. I could not have done this without your patience, encouragement, and expertise. The passion you both show for your students and your work has and will continue to inspire me long after I have left the University of Alberta.

Throughout my nearly eight years at the University of Alberta, my family has unconditionally supported my education and personal growth. My parents, Quan Mao and Rebecca Li, are also two of my greatest role models. They made great sacrifices to ensure my sister and I could lead more comfortable and successful lives. They have also patiently bestowed me with their knowledge and experience so that I may achieve greater heights and pass on our family values. Thank you to my sister, Joy Mao, who is a wise, hard-working and successful person in her own right. You have always been there for me when I needed advice or someone to talk to, and it is so comforting to see our relationship get stronger.

There is an extremely special place in my heart for my dear girlfriend and future wife, Holly Chen. Through you, I gained a vast sense of purpose that fuels my passion to become the best person I can be. Through your intelligence and wisdom, I have learned much. Through your encouragement, comfort and patience, I persevered through tough times. Through your affection, care and thoughtfulness, I experienced love. I wouldn't have made it here without your undying support.

My journey from a high school graduate to a Master's graduate has been accompanied by

the most incredible cast of friends inside and outside the university. Thank you to Linus Hu and Adam Mogensen, who are like brothers to me, for over a decade of friendship; one I am confident will endure for decades to come. Thank you to my good friends Owais Siddiqui, Edward Fan, Jade Barker, Camryn Lowe, Simon Paisley, Nate Peñas, Dante Luu, Mark Sulaimanbekov, Rocky Chen, Tiger-Willow Ward, Abdullah Ahmed, Derek Zhang, Daniel Liu, Jamie Too, Tim Thomson, Levin Joshua, Lauren Cheng, Shahdab Pathan, Karthik Srinivasan, Chinmay Baliga, Seth Beck, Tanay Kumar and Devavrat Thosar. Each of these remarkable individuals has their own unique set of strengths; from these strengths, I tried to learn as much as possible. I am eternally grateful for all your friendships as they helped me become the person I am today.

I would also like to acknowledge the professors of the Chemical and Materials Engineering department who have instructed me throughout both my Bachelor's and Master's degrees, as well as the Digital Research Alliance of Canada, for the computational resources they provided for my research.

Sincerely,

Yiren (Kevin) Mao

Edmonton 2024

Table of Contents

List of Tables

List of Figures

List of Symbols

Constants

e	Euler's number - 2.71828183
$\hbar$	Planck constant - $6.62607015 \times 10^{-34} J/s$

Latin letters

b	Neuron bias
C_t	Cell state of cell t
c	Atomic orbital coefficient
D_{KL}	Kullback-Leibler divergence
E	Epoch
E	Energy
G	Gibbs free energy
H	Prediction Horizon
H	Hamiltonian operator
h_t	Hidden state of cell t
i	ith reaction
j	Mass fraction

K	Inverse sigmoid function coefficient parameter
K	Force constant
k	Apparent reaction rate constant
L	Dynamic principle component analysis lag
$\mathcal{L}$	Lagrangian
l	Sequence length
l	Bond length
m	Sigmoid function parameter
n	Samples
n	Periodicity
p	Probability
Q	Atomic charge
R	Atomic coordinates
R	Atomic distance
S	Solvent
$\mathbf{s}$	Vector of collective variables
T	Total simulation time steps
W	Neural network weights
x^t	Ground truth input at time t
x_t	Input sequence at time t
$\hat{x}_t$	Predicted input at time t
y^t	Ground truth output at time t

$\hat{y}^t$	Predicted output at time t

Greek letters

α	Leaky ReLU coefficient
Δ	Change in
ϵ	Teacher forcing ratio
ε_{dielec}	Dielectric constant
θ	Angle
μ	Fictitious mass parameter
ν	Potential
ρ	Density
σ	Kinetic solvent parameter
χ	Molecular orbital
Ψ	Wavefunction
ω	Angle of rotation

Subscripts

e	electrons
n	nuclei
t	timestep

Abbreviations

3D	three dimensional
4D	four dimensional
AI	artificial intelligence
AIMD	*ab initio* molecular dynamics
ANN	artificial neural network
BCE	binary cross-entropy
BOMD	Born-Oppenheimer molecular dynamics
BLSTM	bidirectional long short-term memory
BNN	Bayesian neural network
CMA-ES	covariance matrix adaptation-evolution strategy
CNN	convolutional neural network
CPMD	Car-Parrinello molecular dynamics
CV	collective variable
DFT	density functional theory
DIO	1,4 dioxane
DMSO	dimethyl sulphoxide

DPCA	dynamic principle component analysis
DRO	Deep Reaction Optimizer
FES	free energy surface
GRU	gated recurrent unit
GVL	γ -valerolactone
HMF	5-hydroxymethylfurfural
KL divergence	Kullback-Leibler divergence
KS	Kohn-Sham
Leaky ReLU	leaky rectified linear unit
LSTM	long short-term memory
MAPE	mean absolute percentage error
MCMM	multi-configurations molecular mechanics
MD	molecular dynamics
ML	machine learning
MLaMD	machine learning-accelerated molecular dynamics
MLIP	machine-learned interatomic potentials
MM	molecular mechanics
MTP	moment tensor potential
PCA	principle component analysis
PDF	probability distribution function
QM	quantum mechanics
QM/MM	quantum mechanics - molecular mechanics

ReLU	rectified linear unit
RMSD	root mean squared deviation
RMSE	root mean squared error
RNN	recurrent neural network
TCL	Tool Command Script
THF	tetrahydrofuran
TS	transition state
VMD	Visual Molecular Dynamics

Chapter 1

Introduction

Currently, many chemical and pharmaceutical processes occur in the presence of a solvent; however, the effect of the solvent environment is not yet fully understood at a fundamental level[1]. In addition to stabilizing the transition state (TS) and altering reaction kinetics, a solvent and its reorganization can also (i) impact reaction mechanisms and pathways, (ii) change the relative stability of the reactants, transition state and products, (iii) alter reaction thermodynamics, (iv) facilitate diffusion in porous catalysts and (v) improve selectivity[2]. These effects must be considered when potential materials and catalysts are being evaluated to achieve optimization. While empirical tests can be conducted to find the ideal solvent(s) and their corresponding concentrations for a reaction through trial and error, this would be time-consuming and expensive while not allowing us to develop a mechanistic understanding behind why a particular solvent is ideal for a given reaction[3]. For such investigations, using molecular dynamics methods becomes an attractive option.

Molecular dynamics is a computational tool that models the dynamics of chemical systems at an atomic level. This level of detail is what allows researchers to gain a deeper level of understanding of chemical systems through the use of molecular dynamics. The dynamics of a system is modeled through classical molecular dynamics (MD) or *ab initio* molecular dynamics (AIMD). Classical MD uses Newton's equations of motion to solve for the evolution of

atomic positions over time. This is often computationally inexpensive, and therefore, is able to model larger systems and access longer timescales. However, it fails to accurately model chemically complex systems such as reaction systems; thus, AIMD is necessary for modeling these systems[4]. AIMD methods, which use quantum mechanics to model molecular dynamics, is often used to study reactions over its classical counterpart. For solvent-based systems, AIMD approaches can either model the solvent molecules implicitly or explicitly. Implicit solvation models imply the presence of solvent molecules by approximating the mean force exerted on the explicitly modeled molecules, while explicit approaches model each solvent molecule within the system as well as their dynamics[5]. While implicit models can be useful by approximating reality with relatively few resources[6], they are accompanied by drawbacks. It has been shown that implicit models fail to accurately capture hydrogen bonding interactions and entropic effects on free energy[7,8]. Explicit solvation modeling, on the other hand, can accurately model solvation systems but is computationally intensive and therefore is limited in its timescale.

A part of what makes AIMD so computationally expensive is the need for the electronic structure of the system to be recalculated after each time step. Car-Parinello molecular dynamics (CPMD) addresses this issue by following the movement of the electronic structure with a classical representation[4]. The metadynamics technique can be used with CPMD to further accelerate reaction events so that they can be observed within a shorter time frame. In metadynamics, small energy potentials are added to the system so that the energy wells of the system within a user-defined coordinate space are filled. The added energy potentials are tracked to reveal the potential energy surface of the system within the coordinate space[9]. The potential energy surface of a reaction system shows the activation energy, which aids us in studying the solvent effects on reactions. Even with the CPMD formulation and the metadynamics technique, a complete simulation of a reaction system containing 232 atoms takes roughly 100,000 CPU-hours. If a study were to be conducted to find the ideal solvents and their concentrations for a given set of reactions, this alone could take years.

To address the computational cost of AIMD methods, researchers have been integrating machine learning (ML) into existing MD methods. It has already been shown that ML algorithms can support classical MD simulations by sampling protein conformations in a more efficient manner[10] and predict reaction rates[11]. However, the impact of ML has not been limited to just classical MD. ML algorithms have been used to accelerate AIMD simulations by predicting the energies and forces of atomic configurations[12,13], build more accurate density functionals[14], and model atomic interactions[15]. In the context of solvents, ML has been used to improve the accuracy of implicit solvation simulations[16] and model reactions in explicit solvation models[17,18]. For explicit solvation environments, ML has been primarily been employed to predict the potential energy surface or reaction barriers within rather than predicting the evolution of the atomic configuration. This work attempts to train a proxy ML algorithm for CPMD metadynamics simulations of explicitly solvated biomass reactions. This model would decrease the time and computational cost of studying such systems by predicting both the atomic configurations and the potential energy surface.

The condensed phase reaction under study is the protonation 5-hydroxymethylfurfural (HMF) in water and dimethyl sulphoxide (DMSO). The reaction mechanism of HMF into humins is of particular interest because humins offer very little economic value while HMF eventually forms profitable fuel additives and polymers[19]. It has been shown that the presence of DMSO as a cosolvent shields HMF molecules thereby hindering their ability to degrade to humins[20]. ML-assisted AIMD simulations could provide such mechanistic insights at a fraction of the current computational cost in a fraction of the time because a proxy ML model would act as a shortcut to the outcomes already attainable through standard AIMD methods. The increased speed and reduction in the computational cost of AIMD simulations would also make screening for cosolvents that increase the selectivity of HMF a less demanding task.

In Chapter 2 of this thesis, a literature review is performed on previous work related to the machine learning application in computational chemistry. Chapter 3 will provide

a background on molecular modeling methods, while Chapters 4 and 5 will explain how different ML proxy models were developed and tested.

Chapter 2

Literature Review

The fields of chemistry and computing have long crossed paths with each other. Initially, simple analog computers were used for basic calculations related to chemical reactions and molecular structures[21], but as both disciplines advanced, so too did their involvement with each other. More recently, thanks to rapid advances in computing power, incredibly detailed simulations can be conducted to study complex chemical systems. Implementing machine learning and artificial intelligence (AI) has also revolutionized how experimental and computational chemistry is approached.

A strength of machine learning is its ability to save time and reduce the costs of tasks it is trained for. Chemists often try to optimize reactions by adjusting one experimental variable at a time while keeping the others constant. However, this method often falls short of identifying optimal conditions and is often laborious and resource-intensive[22]. Hence, developing an ML model for optimizing chemical reactions would be of significant value. Zhou et al.[23] pursued this goal with their Deep Reaction Optimizer (DRO) model, utilizing reinforcement learning for training. Reinforcement learning models take actions based on inputs to maximize a particular "reward." In the context of chemical reactions, the DRO model adjusts reaction conditions to maximize the yield. A mixture of Gaussian density functions was used to train the model. The motivation behind this approach is based on the assump-

tion that the response surface for most reactions can be treated as a continuous function, and a combination of Gaussian density functions can approximate any continuous function. When tested on a set of previously unseen mixture of Gaussian density functions alongside several popular blackbox optimization algorithms, DRO was able to achieve optimization in fewer steps than its competitors. It was first pre-trained on simulation reaction data for four reactions to test its viability for chemical reactions. For each reaction, the model was required to adjust the flow rate, voltage and pressure based on random initial conditions to determine the maximum yield. It was found that DRO achieved optimization in fewer steps than the blackbox optimization algorithm covariance matrix adaptation-evolution strategy (CMA-ES) and the one variable at a time method that chemists typically use to try and achieve optimization[23]. The DRO model successfully identified the optimal reaction conditions by fine-tuning a few parameters that directly influenced the reaction yield. However, it is important to note that the influence of these parameters tends to be consistent across various reactions. On the other hand, altering the solvent environment in which a reaction occurs can yield varying effects for different reactions. Consequently, the DRO model does not possess the capability to foresee whether a different solvent or co-solvent combination in a specific ratio would lead to a higher yield for a given reaction.

Classical MD simulations can help provide insights into the reactant, water, and cosolvent interactions within a condensed-phase reaction system[24]. Based on this information, Walker et al.[11] hypothesized that the reaction rates of acid-catalyzed reactions for biomass-derived compounds could be predicted computationally based on three observables: i) the number of water molecules within the local solvation shell of the reactant, ii) the average lifetime of bonds between the reactant and nearby water molecules and iii) the fraction of reactant surface area exposed to nearby hydroxyl groups. These indicators were quantified and fit into a linear regression model to predict the kinetic solvent parameter σ for a given reaction. The kinetic solvent parameter σ is defined in Equation 2.1 where $k^i_{S,j}$ is the apparent rate constant for reaction i in solvent S with a mass fraction of j and $k^i_{H_2O}$ is the apparent rate

constant for reaction i in pure water.

$$\sigma^i_{S,j} = \log_{10}\left(\frac{k^i_{S,j}}{k^i_{H_2O}}\right) \tag{2.1}$$

A positive kinetic solvent parameter indicates that the solvent environment would increase the reaction rate, while a negative kinetic solvent parameter indicates the opposite. The model was trained on experimentally determined rate constants for seven biomass reactions occurring in an environment consisting of water and one of three cosolvents: 1,4 dioxane (DIO), γ-valerolactone (GVL) and tetrahydrofuran (THF). The results of the regression model demonstrate that these three descriptors are capable of predicting the kinetic solvent parameter within reasonable error, particularly for DIO cosolvent systems[11]. This shows that at least in certain cases, classical MD simulations can predict solvent effects on reaction rates without explicitly modeling the reaction mechanism. The regression model was less accurate for GVL and THF mixtures, indicating that the model has some limitations. The authors concluded that either the descriptors computed from classical MD cannot quantify reaction rates in some systems or more complex descriptors are required to capture more complex trends.

Identifying increasingly complex descriptors requires extensive human effort; therefore, a method that would bypass such rigorous methods would be of great value. Convolutional neural networks (CNN) are machine learning models that use convolutional layers to extract spatial features. It has already been shown that three-dimensional (3D) CNNs trained on protein structures are able to detect protein functional sites[25] and evaluate protein-ligand binding sites[26]. Hence, the possibility that a 3D CNN will be effective at extracting spatial features to capture trends in complex descriptors within a classical MD simulation is high. Chew et al.[3] aimed to train a 3D CNN capable of predicting the reaction rates of acid-catalyzed biomass reactions within a solvent and co-solvent system based on classical MD data. Their 3D CNN, named SolventNet, was trained to predict the kinetic solvent

parameter using 76 experimental reaction rates for seven biomass-derived model reactants along with their associated MD trajectories and tested using 32 experimental reaction rates, including reaction rates with cosolvents not seen within the training set. The model's performance was evaluated based on the predicted kinetic solvent parameter's root mean squared error (RMSE). A 4 nm cubed box within the simulation cell was voxelized into 20 voxels along each dimension, and the normalized molecule counts were stored in each voxel for each simulation time step. This procedure was repeated for the reactant, solvent, and cosolvent and stored in three channels. The voxelized occupancy distributions were further averaged over 2 ns of simulation time to minimize computation cost while not sacrificing model accuracy. SolventNet was found to predict experimental reaction rates more accurately than models based on human-selected, MD-derived descriptors and reaction rates for polar aprotic cosolvents not seen in training based on as little as 4 ns of MD simulation data. The model also gives some level of interpretability through saliency maps, which allowed researchers to identify voxels within each channel that had the most significant impact on the model's prediction[3]. SolventNet shows that machine learning can expedite solvent screening for reaction optimization using classical MD simulations with some level of interpretability. While MD simulations can access longer timescales while keeping computational expense minimal, AIMD methods are able to provide mechanistic details behind reactions, which could be invaluable to researchers. CPMD with metadynamics, a type of AIMD method, can also provide details regarding the energetics of a reaction, something classical MD cannot replicate. Since a machine learning model can reduce the computational cost of MD simulations while still being able to predict its outcomes, then it is reasonable to assume that the same idea can be applied to AIMD trajectories. The following studies showcase this concept.

Häse et al.[27] used a Bayesian neural network (BNN) to predict the dissociation time of unmethylated and tetramethylated 1,2-dioxetane and compared them to the results of AIMD simulations. In addition to accurate predictions of dissociation time, Häse et al. also aimed to interpret the trained BNN to understand the mechanistic relationship between

dioxetanes and their corresponding dissociation time. The dissociation time of 1,2-dioxetanes into two formaldehyde molecules is of particular interest because this decomposition exhibits chemiluminescent properties[27]. The chemiexcitation yield is directly proportional to the dissociation time, so studying molecular variations of 1,2-dioxetane is useful to determine which variation will produce the highest yield[28]. Two separate BNNs, named BNN1 and BNN2, were trained. BNN1 predicted the dissociation time based on the initial nuclear geometry of the system, while BNN2 used both the nuclear geometry and velocities to predict the dissociation time. By directly predicting the dissociation time based on the system's initial configuration, the BNNs bypass almost all the computational costs AIMD typically requires. When predicted and actual values were plotted against each other, the models BNN1 and BNN2 were found to have r^2 values of 0.86 and 0.97, respectively. BNN 2 also had a smaller mean absolute deviation, meaning the model had better generalizability. The models were also able to outperform AIMD in terms of computational efficiency. Each model could make predictions based on 250 different initial conditions in a few seconds, whereas it would take over one year of computation for AIMD to do the same. The weights and biases of the BNNs were initialized using the Laplace distribution[27]. It has been shown that after training a BNN initialized with the Laplace distribution, the magnitude of the weights within the BNN is related to the corresponding feature's relevancy in the final prediction[29]. Using the weights of the trained BNNs, it was found that the planarization of the two formaldehyde molecules and the shortening of their C-O bonds were associated with an earlier dissociation time[27]. This work proved that ML used on AIMD data could make accurate predictions regarding AIMD outcomes while keeping computational costs at a minimum and providing mechanistic insights regarding the studied system(s).

Eslamibidgoli et al.[30] trained a gated recurrent unit (GRU) and a long short-term memory (LSTM) model to predict various solids' AIMD trajectories and potential energy profiles. The training and testing data set was generated from AIMD trajectory data over 5000 fs on rutile $Ir_{0.75}M_{0.25}O_2(100)$ surfaces where M is one of the following metals: Co, Cr, Ir, Mn,

Mo, Nb, Ni, Pb, Pt, Rh, Ru, Sn, Ta, Ti, V and W. In addition to trajectory data for each atom within the simulated call, the potential energy profile and the temperature fluctuations for the simulations were also included as input data for each model. One iridium atom was chosen at random from the training data to test each model. The model's prediction of the Cartesian coordinates for this atom, and the total potential energy profile, was compared with ground truth data from AIMD using mean absolute percentage error (MAPE). The test MAPE for the iridium atom's coordinates was less than 0.9% for the GRU model and 1.3% for the potential energy profile, while the LSTM model recorded MAPEs of 0.2% and 2.1%, respectively. This shows that both the GRU and the LSTM models were very accurate when predicting the Cartesian coordinates and the potential energy profile. Further investigation found that the spatial distribution of atoms within the simulation cell and the distribution of temperature fluctuations and potential energy were all Gaussian. Since GRU and LSTM models tend to work better with Gaussian distributions, it makes sense why these models performed so well[30]. This work shows that GRU and LSTM models can be used as proxy models instead of AIMD simulations for solid-based systems. While the prediction horizon for these models isn't very long, future more complex models have the potential to make longer-term predictions and, therefore, save even more computational costs. This concept can also be expanded towards condensed phase systems, where the molecular interactions differ, providing an extra challenge for ML models.

Puliyanda[31] aimed to train a 3D CNN autoencoder to extract features based on the AIMD trajectory of a reactant from which the extent of solvent reorganization would be determined. AIMD simulations of the reactant and product trajectories for the pyrolytic decomposition of cellobiose at four different temperatures (100 K, 500 K, 900 K and 1200 K) were used to train the model. The root mean squared deviation (RMSD) between the encoded features of the reactant and product was fit to a probability distribution using kernel density estimation. This probability distribution function was used to determine the extent of solvent reorganization. Solvent reorganization was found to be more significant

for the cellobiose system at 100 K and 500 K and less significant at 900 K and 1200 K. The 3D CNN autoencoder model was tested using reactant trajectories of an acid-catalyzed conversion reaction of fructose to HMF in a solution of water and DMSO. A Mahalanobis distance-based classifier was used to compare the extracted features of the reactant trajectory of the new system to the extracted features of Class 1. Class 1 is the training data that exhibited the highest degree of solvent reorganization was the cellobiose system at 100 K. The model predicted solvent reorganization would decrease significantly by introducing 5% wt DMSO to a pure water solution. In comparison, DMSO concentrations beyond 5% wt would result in a linear increase in solvent reorganization. This trend agreed with free energy surface minima calculations for the reactant and product states with differing concentrations of water and DMSO. When % wt DMSO was introduced into the system, the free energy difference between the hydronium ion occupying the bulk solvent and the hydronium ion occupying the first solvation shell of fructose (ΔG) increased. This suggests the DMSO preferred to group near the fructose molecule while the water molecules stabilized the hydronium ion, resulting in a sharp decrease in solvent reorganization. When the DMSO concentration was increased beyond 5% wt, ΔG decreased while solvent reorganization increased. This suggests that the hydronium ions were forced to interact with the fructose molecule due to a lack of water molecules in the system and increasing solvent reorganization[31]. This model shows that it is possible to screen reactions for their level of solvent reorganization in different solvents. This is advantageous because if solvent reorganization is found to be minimal for a reaction under certain solvent conditions, then the mechanics and energetics of the reaction can be studied using an implicit solvent environment instead, saving computational costs and time. If solvent reorganization is significant, then a different approach must be used to study the reaction cost-efficiently. This complex problem could be tackled with machine learning as well.

Chen et al.[32] used an active learning framework with machine-learned interatomic potentials (MLIP) to accelerate explicit solvent AIMD simulations of solvated heterogeneous

catalytic systems. MLIPs are a class of ML models that relate the potential energy of a system to the coordinates of the atoms it contains[33]. It has already been shown that MLIPs are able to assist in the search of global minima[34] and transition states[35]. Typically, an accurate MLIP requires a training set consisting of thousands of Density Functional Theory (DFT) calculations, presenting a sizeable computational barrier to the approach. It is also difficult to intuitively predict what kind of training set would result in the generalizable MLIPs[32]. To rectify these obstacles, an MLIP obtained from literature, called moment tensor potential (MTP)[36], was trained "on-the-fly" through active learning[32]. What makes active learning unique is that ML models trained through active learning select the data they are trained on, which is advantageous when the availability of labelled data is limited[37]. Chen et al. used their ML model to perform machine learning-accelerated molecular dynamics (MLaMD) simulations during active learning. Poor predictions were reassessed using DFT and added to the training data pool. This ensures that the training dataset grows more comprehensive with minimal effort. As the training data expands, the need for DFT calculations to be performed decreases, thereby decreasing the computational costs of MLaMD simulations. DFT calculations are also performed periodically to ensure the continued accuracy of the ML model. This work found that their MLaMD simulations were able to (i) accurately model the molecular structure of water near a metal surface, (ii) predict adsorption energies of CO* and OH* onto Cu(111) and (iii) estimate the energy barrier of the C-H bond breaking in ethylene glycol over Cu(111) and Pd(111) with MLaMD-based metadynamics. Other advantages of this model include far fewer required DFT calculations, simulation times four orders of magnitude shorter than AIMD, and access to longer time scales[32].

Chapter 3

Molecular Dynamics Methods

Molecular dynamics (MD) is a computer simulation tool that allows dynamic molecular systems to be studied at the atomic level[38]. The ability to simulate atomic dynamics provides researchers with previously inaccessible insight regarding chemical systems at the expense of computational costs. Ever since the 1970s, MD has been used to compliment physical experiments to deepen our level of understanding[39]. With computational resources becoming increasingly more accessible, MD has started to play a more significant role in chemistry, material science and biology. It has been used to aid in drug development[40], test the properties of new materials[41] and study chemical reactions[42].

3.1 Classical Molecular Dynamics

There are two main types of MD: force field-based MD (or classical MD) and *ab initio* molecular dynamics (AIMD)[4]. Both types of MD update the spatial arrangement of atoms based on the potential energy of the system, but how they calculate the potential energy differ. Classical MD uses molecular mechanics (MM) to determine the total potential energy while AIMD uses quantum mechanics (QM)[43]. In MM, Newton's equations of motion are applied by treating each atom as a particle of a certain size and mass, and each bond as a spring with a certain length and stiffness. The energy of the forces within a system is

modeled by a set of equations called the force field, hence the name force field-based MD. The modeled forces can be divided into bonded interactions, which include stretch, bend and torsion energies, and non-bonded interactions, which include van der Waals and electrostatic energies, as indicated in Equation 3.1 [4].

$$\mathrm{E}_{MM} = \mathrm{E}_{stretch} + \mathrm{E}_{bend} + \mathrm{E}_{torsion} + \mathrm{E}_{vdW} + \mathrm{E}_{electrostatic} + (\mathrm{E}_{coupled}) \tag{3.1}$$

The terms $\mathrm{E}_{stretch}$, E_{bend}, $\mathrm{E}_{torsion}$, E_{vdW} and $\mathrm{E}_{electrostatic}$ represent the energy for bond stretching between two atoms, energy for angle bending between three atoms, energy for bond twisting, van der Waals interaction and electrostatic interaction, respectively. An example of equations for each of these terms are as follows:

$$\mathrm{E}^{12}_{stretch} = \mathrm{K}_{stretch}\left(\mathrm{l}^{12} - \mathrm{l}^{0}\right)^2 \tag{3.2}$$

$$\mathrm{E}^{123}_{bend} = \mathrm{K}_{bend}\left(\theta^{123} - \theta^{0}\right)^2 \tag{3.3}$$

$$\mathrm{E}^{1234}_{torsion} = \mathrm{K}_{torsion}\left(1 - \cos\left(\mathrm{n}\omega\right)\right) \tag{3.4}$$

$$\mathrm{E}^{12}_{vdW} = 4\mathrm{E}^{L-J}_{min}\left[\left(\frac{\mathrm{R}^0}{\mathrm{R}^{12}}\right)^{12} - \left(\frac{\mathrm{R}^0}{\mathrm{R}^{12}}\right)^{6}\right] \tag{3.5}$$

$$\mathrm{E}^{12}_{electrostatic} = \frac{Q^1 Q^2}{\varepsilon_{dielec}\mathrm{R}^{12}} \tag{3.6}$$

where $\mathrm{K}_{stretch}$, K_{bend}, and $\mathrm{K}_{torsion}$ are force constants, ω is the angle of rotation, n is periodicity, E^{L-J}_{min} is the depth of the minimum in the potential, R^0 is the distance at zero potential, Q is atomic charge and ε_{dielec} is the dielectric constant [4].

Each energy term in Equation 3.1 is only a function of atomic coordinates. Therefore, it is through the minimization of E_{MM} as a function of atomic coordinates that the lowest energy state of a given system is found [44]. There are two types of minimization algorithms: local and global. Local minimization algorithms locate the system's nearby energy minima

however, this minima may not be the lowest possible minima of the system. The global minima of the system refers to the lowest possible energy state of the system which is ideal result through minimization. Two standard methods that attempt to find the global energy minima are Monte-Carlo and molecular dynamics (MD). The Monte-Carlo method finds global minima through trial and error adjustments to a system's configuration. On the other hand, MD gives each atom within the system a velocity based on the system's temperature and classically solves for the position of each atom. MD better represents the system's behaviour in reality since it accounts for temperature-dependent dynamics. The general algorithm used to solve for atomic positions and their evolution over time can be seen in the velocity Verlet MD algorithm. For a system of N atoms $\vec{R}\,(R_1, R_2, \ldots, R_N)$ the velocity Verlet algorithm can be summarized as such[4]:

1. Start with initial positions $\vec{R}_0$ and velocities $d\vec{R}_0/dt$
2. Calculate potential energy $\mathrm{E}\left(\vec{R}\right)$ and forces $d\mathrm{E}/d\vec{R}$
3. Recalculate positions with

$$\vec{R}_{t+1} = \vec{R}_t + \left(\frac{d\vec{R}_t}{dt}\right)\Delta t + \frac{1}{2!}\left(\frac{d^2\vec{R}_t}{dt^2}\right)(\Delta t)^2$$

4. Recalculate potential energy $\mathrm{E}\left(\vec{R}_{t+1}\right)$ and forces $d\mathrm{E}/d\vec{R}_{t+1}$
5. Update velocity with

$$\frac{d\vec{R}_t}{dt} = \left(\frac{d\vec{R}_t}{dt}\right) + \frac{1}{2}\left(\frac{d^2\vec{R}_t}{dt^2} + \frac{d^2\vec{R}_{t+1}}{dt^2}\right)\Delta t$$

6. Return to step 3

Many force fields exist because parameter values for the energy terms in Equation 3.1 are derived from experimental data, which give differing results depending on the data is

used[45]. The equations used to calculate each energy term and the inclusion of extra energy terms that capture coupled interactions of forces ($E_{coupled}$) are also ways one force field differs from another. The systems each force field is best suited for depend on the experimental data they were parameterized on. Therefore, a trade-off exists for each force field where increased accuracy for one type of system means sacrificing generalizability[4].

MM methods are computationally inexpensive and can easily model large systems such as proteins and DNA. However, for chemically exotic systems where few or no accurate parameters exist, the effectiveness of MM may be reduced. It is also extremely difficult for MM to be applied to systems containing metals and reaction systems. The bonds in metal systems are not as well-defined as in organic systems. MM has no clear answer to this since it treats atoms as singular entities rather than nuclei with accompanying electronic structures. MM also breaks down when dealing with reaction systems because the transition from reactant to product changes the bonds and possibly the number of atoms they contain, meaning that separate force field energy functions are required for the reactant and product[45].

3.2 *Ab Initio* Molecular Dynamics

As discussed in Section 3.1, the general idea of MD is to solve for the potential energy, solve for the forces, update the velocities, and recalculate the position for the next timestep. In AIMD, QM is used to solve for the electronic structure of a system and its potential energy. The Shrödinger equation (Equation 3.7)[46] describes the spatial and temporal evolution of a particle's wavefunction in a potential.

$$\mathrm{H}\Psi_(R,t) = i\hbar\frac{\partial}{\partial t}\Psi(R,t) \tag{3.7}$$

Solving the Shrödinger equation with the Born-Oppenheimer approximation decouples electronic and nuclear motion, thereby allowing them to be solved separately through a quantum mechanical and a classical formulation, respectively. This allows the electronic structure to

be optimized based on the nuclear configuration after each time step. This is known as Born-Oppenheimer molecular dynamics (BOMD). The Lagrangian for BOMD is shown in Equation 3.8. Nuclei can be treated classically because they have sufficient mass to obey Newtonian physics. Therefore, only the solution to the Shrödinger equation for electrons in a system (Equation 3.9) is needed to find the potential energy[4].

$$\mathcal{L}_{BOMD} = \frac{1}{2}\sum_i m_i R_i^2 - \langle \Psi | H | \Psi \rangle \tag{3.8}$$

$$\mathrm{H}_e \Psi_e \left(R_e, R_n\right) = \mathrm{E}_e \Psi_e \left(R_e, R_n\right) \tag{3.9}$$

In the above equations, H is the Hamiltonian operator, $\hbar$ is the Planck constant, Ψ_e is the wavefunction for electrons, R_e is the electronic coordinates and R_n is the nuclear coordinates. Since Equation 3.9 cannot be solved analytically for systems with more than one electron, an approximation must be made for polyelectric systems[44].

Density functional theory (DFT) is a quantum mechanical calculation method that uses electron density instead of wavefunctions for each electron to find a system's energy. Hohenberg and Kohn showed that an external potential $\nu(\vec{R})$ can be determined if the electron density is provided and that wavefunctions can be expressed as a functional of electron density[47]. This reduces the complexity of the energy calculation since only the density function is needed[4]. Based on this idea, Kohn and Sham developed a formulation that would be able to calculate the energy of a multi-electron interacting system using DFT while keeping computational costs to a minimum[48]. Kohn-Sham DFT introduces a set of fictitious non-interacting electrons with Kohn-Sham orbitals χ_{KS} in a potential $\nu_{KS}(\vec{R})$. The Kohn-Sham orbitals are chosen so that they have the same electron density ρ as the interacting system in the real potential $\nu(\vec{R})$. Equation 3.10 shows how the energy can be calculated based on the density. The energy in Equation 3.10 needs to be minimized with respect to density because of the variational principle[4]. The variational principle states that the calculated energy based on approximate wavefunctions will always be greater than the actual energy

of the system[45]. Therefore, the ground state density needs to be calculated in an iterative manner to find the density that minimizes energy. With Kohn-Sham DFT, the potential energy of a system can now be solved quantum mechanically and AIMD can be performed.

$$\mathrm{E} = \min_{\rho} \{\langle \Psi[\rho]|H|\Psi[\rho]\rangle\} \tag{3.10}$$

AIMD holds a significant advantage over classical MD methods in that the limitations of classical MD do not apply to AIMD. Chemically exotic systems, metals systems and reaction systems can now be accurately modeled through AIMD. However, due to the substantial computation cost associated with AIMD, the steep trade-off between computational costs and accuracy makes AIMD rarely practical[4].

The hefty computational cost of AIMD comes from the need to optimize the electron density after each time step in order to ensure the system is at its energy minima[49]. To work around this, Car and Parrinello demonstrated how the movement of the optimal electronic structure between time steps could be approximated with a classical formulation (Car-Parrinello molecular dynamics (CPMD)). This is achieved by assigning the wavefunctions a fictitious mass μ which gives them fictitious kinetic energy. The extended Lagrangian for CPMD is defined as[50]:

$$\mathcal{L}_{CPMD} = \frac{1}{2}\sum_i m_i \dot{R}_i^2 + \frac{1}{2}\sum_j \mu_i \langle \dot{\chi}_j|\dot{\chi}_j\rangle - \langle \Psi|H|\Psi\rangle + \text{constraints} \tag{3.11}$$

where χ_j is the orbital of the j^{th} electron and the constraints are external or internal constraints placed on the system. The CPMD Lagrangian results in the following Euler-Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}_{CPMD}}{\partial \dot{R}_i}\right) - \frac{\partial \mathcal{L}_{CPMD}}{\partial R_i} = 0$$
$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}_{CPMD}}{\partial \dot{\chi}_i^*}\right) - \frac{\partial \mathcal{L}_{CPMD}}{\partial \chi_i^*} = 0$$

$$
\begin{aligned}
m_i\ddot{R}_i &= -\frac{\partial}{\partial R_i}\langle\Psi|H|\Psi\rangle + \frac{\partial}{\partial R_i}(\text{constraints}) \\
\mu_i\ddot{\chi}_i &= -\frac{\partial}{\partial \chi_i^*}\langle\Psi|H|\Psi\rangle + \frac{\partial}{\partial \chi_i^*}(\text{constraints})
\end{aligned}
\tag{3.12}
$$

which can solved numerically using the velocity Verlet algorithm described in Section 3.1. To ensure CPMD still follows BOMD, the assigned μ must be small enough to prevent energy transfer from the nuclei to the wavefunction. With CPMD, only a single electronic structure is required while the classical formulation takes care of the rest. Despite the reduction in computational costs that CPMD provides, the timescales it can access still fall short of classical MD by five orders of magnitude[4].

To accelerate atomic events so they can be observed within CPMD's short timescale, the metadynamics method was developed[51]. Metadynamics adds small energy potentials to fill a system's free energy surface (FES) within the collective variable (CV) space of interest. The CV space chosen depends on the objective of the study; examples of CVs are bond length, bond angle, bond distance or coordination numbers. This technique requires an extended CPMD Lagrangian which is defined as[52]:

$$
\mathcal{L}_{MTD} = \mathcal{L}_{CPMD} + \frac{1}{2}\sum_{CV} m_{CV}\dot{\mathrm{s}}_{CV} - \frac{1}{2}\sum_{CV} \mathrm{K}_{CV}\left[\mathrm{S}_{CV}\left(\vec{R}_{CV}\right) - \mathrm{s}_{CV}\right]^2 + \nu_{CV}(t,\mathbf{s}) \tag{3.13}
$$

where $\mathbf{s}$ is the vector of CVs, m_{CV} is the fictitious mass assigned to the collective variables, K_{CV} is the force constant, $\mathrm{K}_{CV}\left[\mathrm{S}_{CV}\left(\vec{R}_{CV}\right) - \mathbf{s}_{CV}\right]$ is the potential energy acting on the CVs and $\nu_{CV}(t,\mathbf{s})$ are the energy potentials added. Similar to the selection of μ in CPMD, the selection of m_{CV} in metadynamics must be sufficiently large so that the dynamics of CVs are separate from the dynamics of both nuclear and fictitious electronic motion. By adding energy potentials $\nu_{CV}(t,\mathbf{s})$, the system is forced to cross energy barriers to reach nearby local and global energy minima. These energy potentials are tracked and used to reconstruct the FES after the simulation[53]. An example of how metadynamics explores and allows for the reconstruction of a FES is shown in Figure 3.1. The training and testing data used in

Chapters 4 and 5 comes from a CPMD metadynamics simulation of 5-hydroxymethylfurfural in water and dimethylsulphoxide. With CPMD and the metadynamics method, it is possible to simulate reactions accurately within a reasonable time frame, but there is still room for improvement. Chapters 4 and 5 will explore ways a machine learning model could be used as a proxy model with CPMD metadynamics to achieve the same modeling results at a fraction of the computational cost.

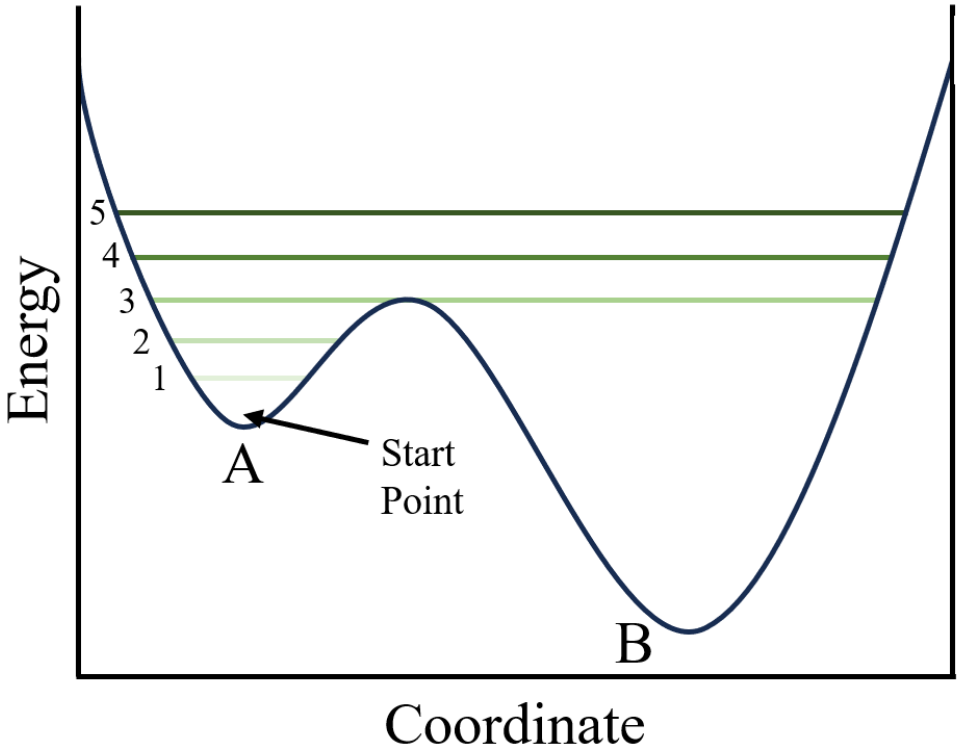


Figure 3.1: Example of how metadynamics explores a FES for a system that goes from state A to a more stable state B. The potential energy surface is filled starting from "well" A as shown by lines 1 and 2. As the "well" gets filled up it spills into "well" B as shown with line 3. Then the FES is filled uniformly (lines 4 and 5).

Chapter 4

LSTM autoencoder application for the time-series prediction of spatial features from CPMD with metadynamics simulation data

4.1 Introduction

Applying machine learning to *ab initio* molecular dynamics (AIMD) simulations of solvent-based reaction systems, particularly those with significant solvent reorganization, involves a two-step process. First, it requires making a time-series prediction of the spatial configuration of the system. Subsequently, this spatial configuration is used to predict the system's free energy for that time step. This chapter aims to train a set of models that can act as a proxy model for Car-Parrinello molecular dynamics (CPMD) metadynamics simulation given a short initial simulation. Thus, a time-series prediction of the simulation is necessary. The outputs from CPMD simulations performed with metadynamics can serve as training data for a set of models that can address this two-step process. CPMD, as explained in Chapter

3, is a type of AIMD simulation that approximates the movement of the electronic structure based on a classical formulation instead of using a quantum formulation to recalculate the electronic structure for each time step, thereby reducing the computational expense of the simulation[50]. Despite this, CPMD's accessible time scale remains limited to picoseconds[4]. However, when paired with the metadynamics technique that accelerates the dynamics within the system, a larger time scale is now accessible. Metadynamics influences the system by periodically adding small energy potentials to fill up the free energy surface (FES) within the coordinate domain of a set of human-selected collective variables (CVs)[51]. These CVs are specifically selected based on the system being studied. Moreover, metadynamics allows the reconstruction of an FES post-simulation[53]. This reconstruction aids in studying reaction systems as it provides the free energy of the reactant, transition and product states. By pre-processing the trajectory and FES data from the CPMD-metadynamics simulation, this dataset can serve as training and testing data for machine learning (ML) models to achieve the previously stated objective.

4.2 Methods

The dataset designated for training comes from a CPMD-metadynamics simulation of the protonation of 5-hydroxymethylfurfural (HMF) in 50 wt% dimethyl sulfoxide (DMSO) and water. The simulation ran for $T = 200,000$ time steps for a simulation cell with side lengths of 11.832 Å, containing one HMF, one proton, 9 DMSO and 42 water molecules. The time step used for the simulation was 0.0967 fs. Two CVs were chosen for metadynamics for this reaction: the coordination number of the proton participating in the reaction (H_α) with bulk oxygen belonging to water (O_{water}) and the coordination number of the protonation site (C_4) with bulk water belonging to water (H_{water}) and (H_α). The CVs selected with respect to the protonation reaction are shown in Figure 4.1. For further details regarding the parameters of the CPMD-metadynamics simulation, refer to Calderón and Mushrif[54].

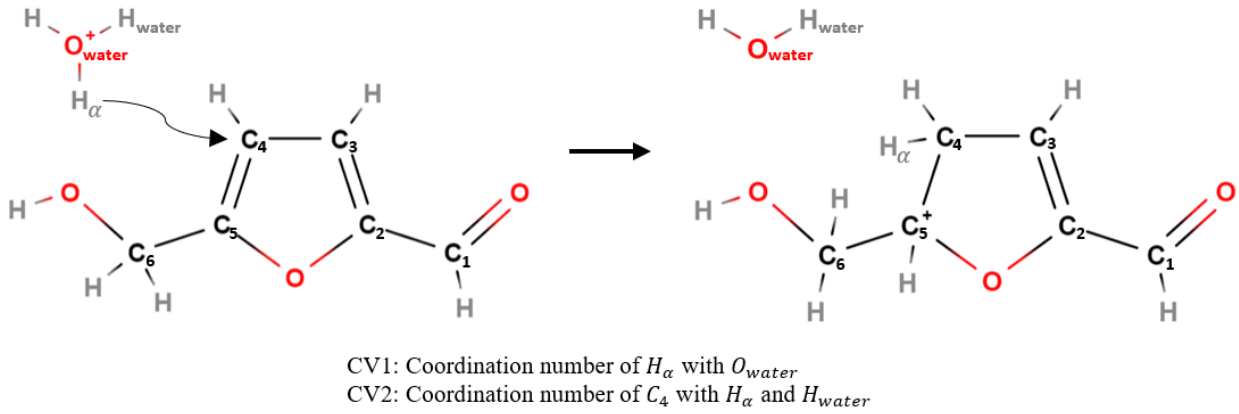
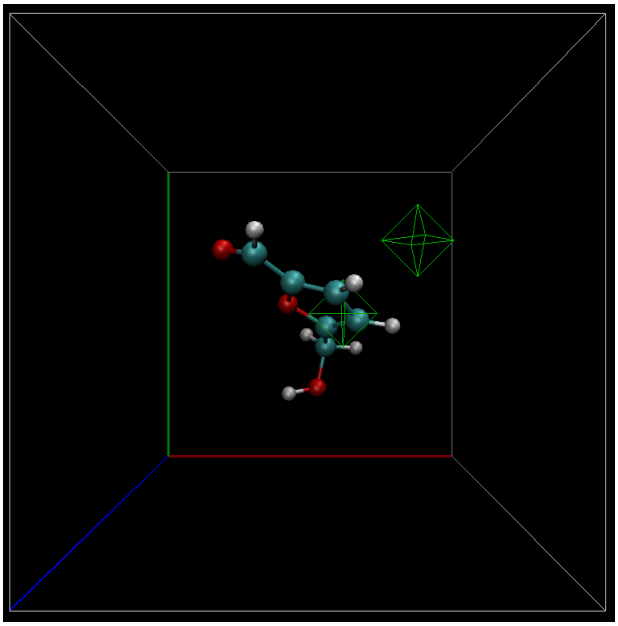


Figure 4.1: Defined collective variables for the protonation of HMF. Adapted from Calderón and Mushrif[54].

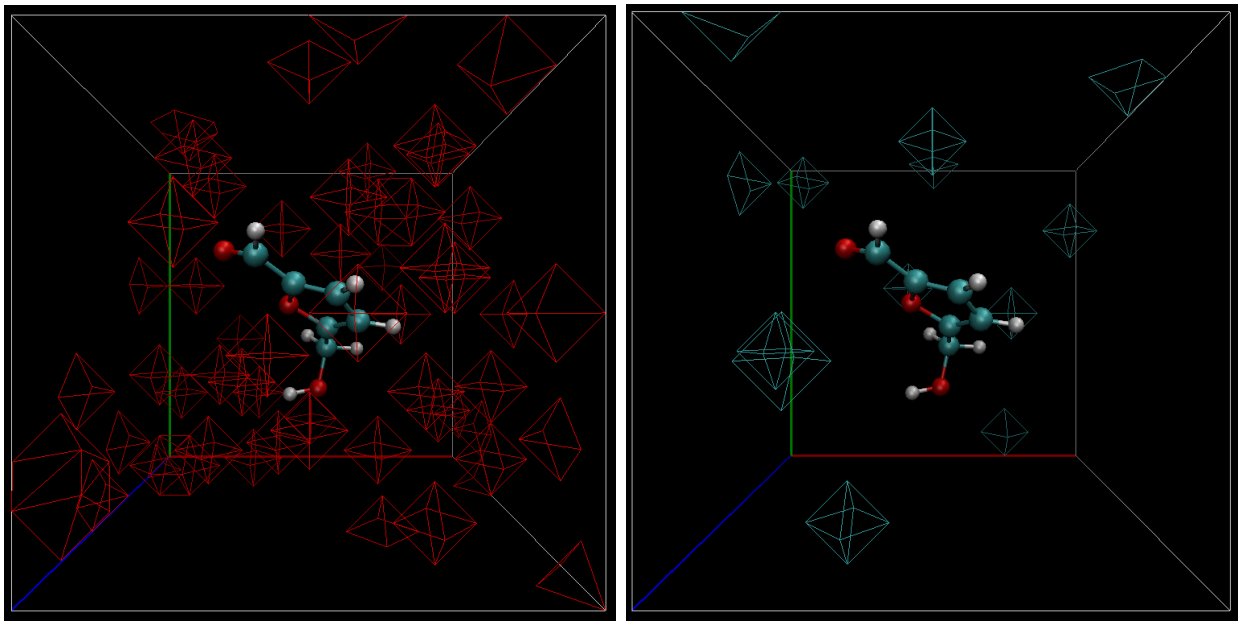
Future analysis will change the reference point of the simulation to the HMF molecule. This could cause issues because the location of the HMF molecule within the simulation cell was close to a corner. Based on the trajectory file, there is a large region near the HMF molecule where no solvent molecules exist because the periodic boundary condition is implied within the output file. Since this isn't indicative of reality, two periodic images of the simulation cell were created in each dimension (26 periodic images) so that the output file explicitly shows the periodic boundary condition. The $T = 200,000$ time steps were segmented into slices of 10 frames each, and each time slice was loaded into the Visual Molecular Dynamics (VMD) software, where each frame in each slice was aligned so that the orientation and position of the central HMF molecule are the same for each time slice. This alignment changes the position and trajectory of each atom in the system so that they move relative to the central HMF molecule instead of the origin of the simulation cell. This modification to the simulation data allows each simulation cell to be directly compared with each other because the HMF molecule will have a consistent position and orientation. A Tool Command Language (TCL) script was used within VMD to generate an occupancy distribution for each species and each time slice. An occupancy distribution shows the fraction of frames that contain the molecule of focus for every discrete point within the system. The discrete points are generated by VMD automatically by finding

the smallest 3D grid needed to contain the entire occupancy distribution and dividing each dimension into 1 Å gridlines. The orientation of the 3D grid VMD creates always remains constant; therefore, each occupancy distribution can be directly compared with one other. The occupancy at the intersection of every gridline is recorded by VMD and saved as a file. This process yields 60,000 total occupancy distributions, $n = 20,000$ samples for each species. Given the substantial volume of data in each file, resulting in billions of data points, voxelization was necessary to condense the information. It can be assumed that solvent molecules further from the reaction will have a negligible effect compared to those close to the studied reaction. Therefore, a cutoff of 12 Å was assigned for each occupancy distribution beyond which solvent molecules were assumed to have negligible impact on the reaction. This cutoff strikes a balance between including solvents and cosolvents in the vicinity while not incorporating too much space that could lead to a voxelization with an overly large feature space or a feature space with a large resolution. A Python code was used to voxelize each dimension into 12 voxels, with the HMF molecule acting as the centre of each voxelized distribution. The values within each voxelized occupancy distribution were normalized to yield a probability distribution function (PDF). A voxelization factor of 12 was chosen to give the voxelized PDF a resolution of 2 Å. With this resolution, there is no concern that two different water molecules will overlap on the same voxel as the intermolecular distance between water molecules is roughly 3 Å. Figure 4.2 shows a sample occupancy map for each of the three species while Appendix A contains sample projections of the final PDFs onto two dimensional planes.

The goal is to utilize past PDFs to make time-series predictions of future PDFs for each species. This information will aid in making predictions regarding the energetics for the studied reaction. Artificial neural networks (ANNs), particularly those containing recurrent layers, are commonly used for time-series predictions. An ANN consists of interconnected layers, each composed of neurons. Each connection between neurons is associated with a weight W, and each neuron has an associated bias b. It is these values that change during



(a)



(b) (c)

Figure 4.2: Projection of a randomly selected PDF for each of the three species onto the XY plane.

training to form the network's memory. Each neuron's activation function determines the output fed to the neurons connected to itself, as pictured in Figure 4.3[55].

ANNs generally contain an input layer, an output layer, and hidden layers between the input

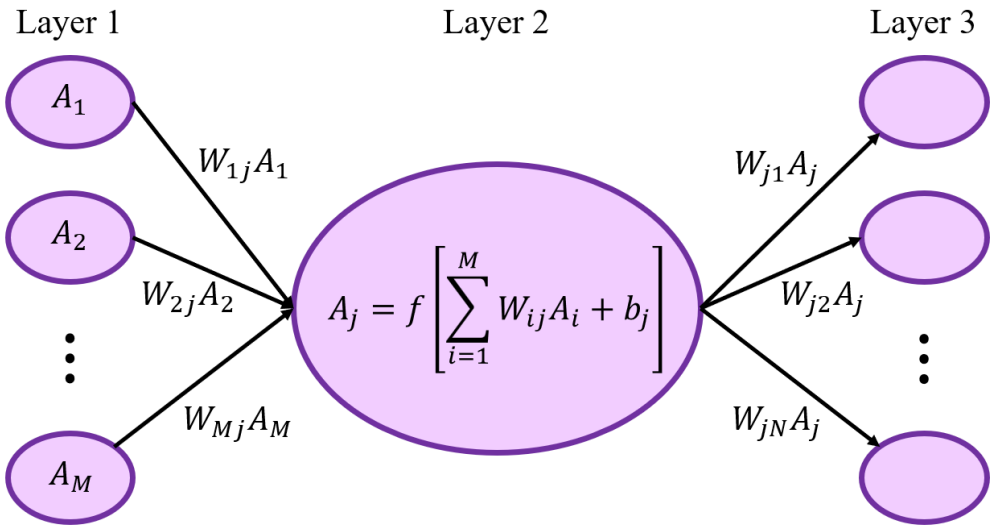


Figure 4.3: Neuron structure within a neural network. Adapted from Maren et al.[55].

and output layers. Special layers, such as recurrent layers, introduce additional memory elements into the network to optimize its performance for certain tasks. As illustrated in Figure 4.4, Recurrent layers contain a sequence of neurons that relay information regarding previous inputs to subsequent layers, making them useful for time-series predictions. However, recurrent layers may face problems during backpropagation. The gradients responsible for updating weights and biases can become extremely small or extremely large. This problem is known as a vanishing gradient and an exploding gradient, respectively[56]. Any recurrent layers that contain many neurons are particularly vulnerable to this weakness, making them more suitable for shorter sequences. The Long-Short Term Memory (LSTM) layer was created to address the exploding or vanishing gradients and handle longer sequences, making it a sensible starting point for this study[57]. Within each neuron, a series of gates and activation functions update a cell state that is passed on from one neuron to the next within the LSTM layer. A visualization of these gates and activation functions is shown in Figure 4.5, with the cell state indicated with a dashed line. Given the previous hidden state $h_{(t-1)}$ and the new input x_t, the forget gate determines what information should be forgotten while the input gate determines what new information should be added. The outputs of these two gates are then applied to the old cell state $C_{(t-1)}$ to update it into the new cell state C_t.

Finally, the output gate determines the new hidden state h_t based on the new cell state C_t that was just updated[58].

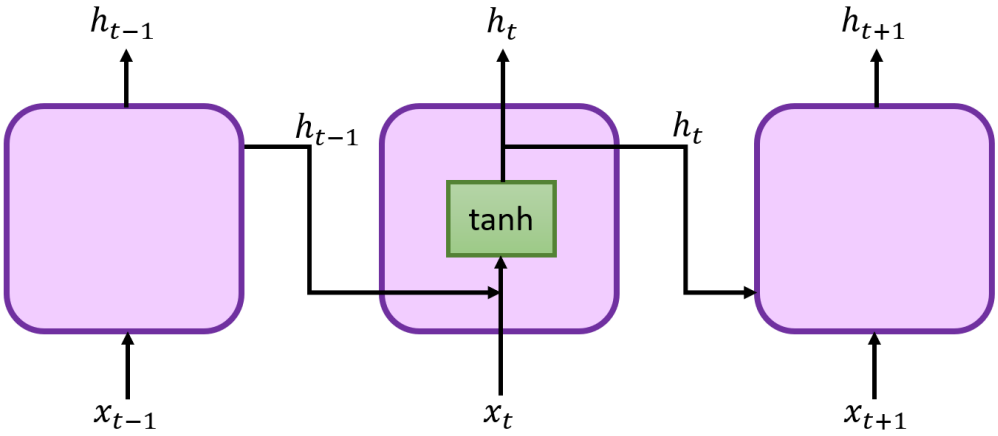


Figure 4.4: Visualization of a recurrent layer.

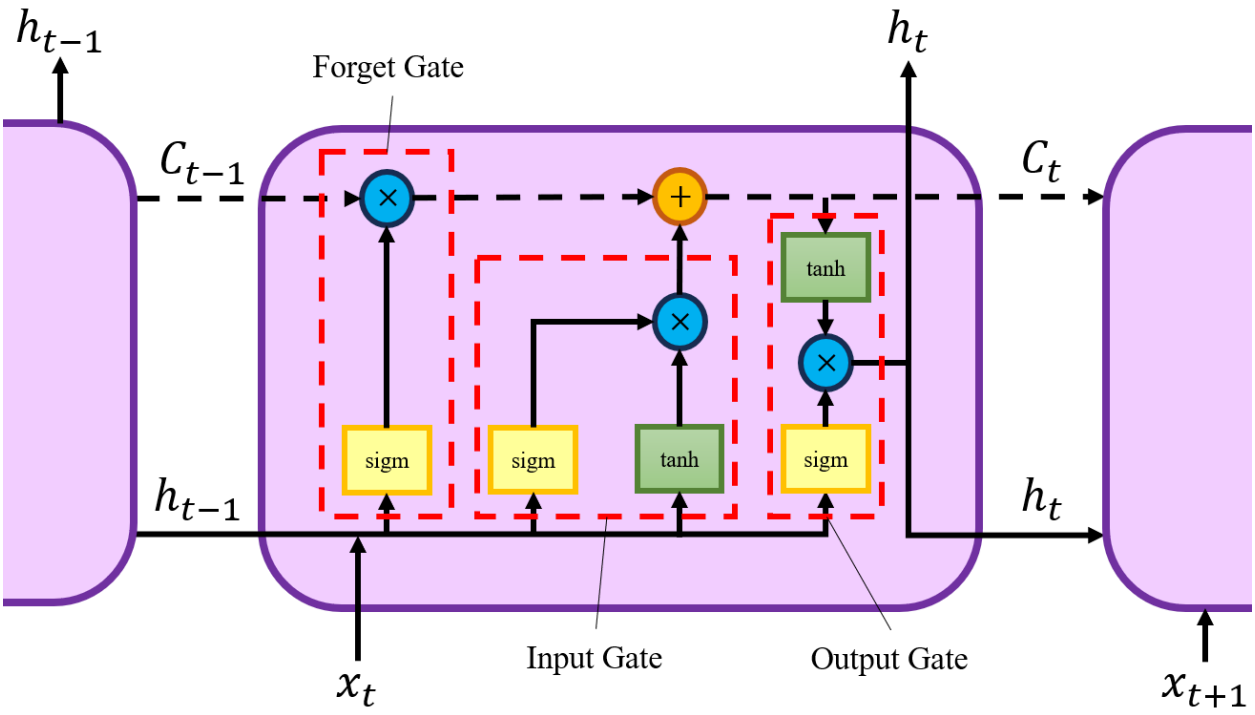


Figure 4.5: Visualization of an LSTM layer. Adapted from Eslamibidgoli et al.[30].

The number of past time steps used to predict the next time step is called the sequence length (l) and is an adjustable hyperparameter for each model. Since an LSTM layer is being used within each model, the sequence length will be able to take on a wider range of

values while keeping the threat of an exploding or vanishing gradient to a minimum. In the training phase of each model, $n = 20,000$ PDFs for each species were arranged into sequences of length l and paired with its designated output, which is the PDF of the species for the next step. Figure 4.6 demonstrates an example of this procedure.

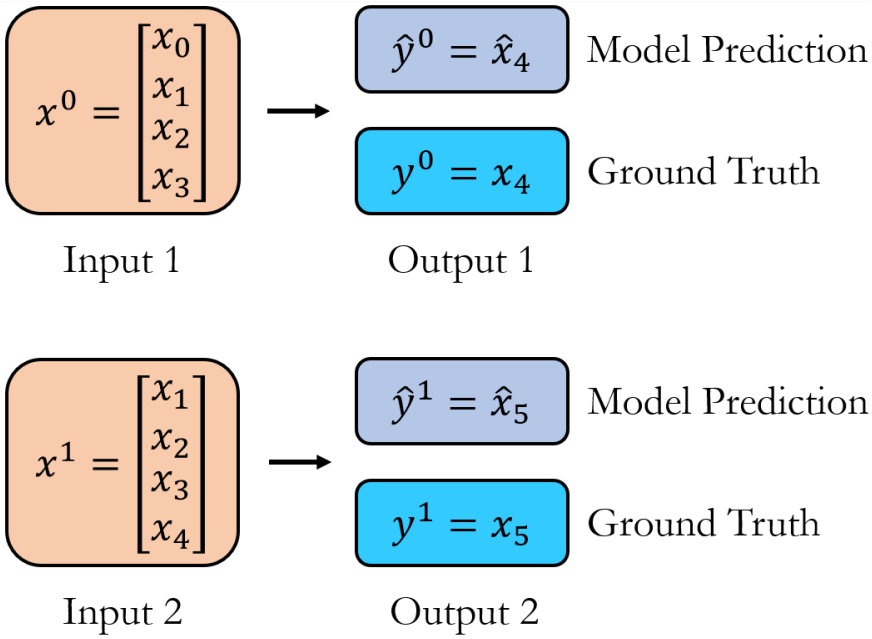


Figure 4.6: Representation of the pairing of input sequences to their outputs.

This creates a total of $n - l$ samples. Of the complete dataset, 80% was allocated for training, while the rest was used as validation data. The loss function used to evaluate the performance of the model was the Kullback-Leibler Divergence (KL divergence). For two PDFs, $P(x)$ and $Q(x)$, the KL divergence is defined as[59]:

$$D_{KL}(P||Q) = \sum_{x \epsilon X} P(x) \ln \left(\frac{P(x)}{Q(x)} \right) \tag{4.1}$$

The KL divergence measures the entropy between two PDFs, $P(x)$ and $Q(x)$, indicating their similarity or dissimilarity. This loss function is ideal for the given situation as it allows us to evaluate how similar the predicted PDFs $\hat{y}^t$ are to the ground truths y^t established by the CPMD-metadynamics simulation. The KL divergence will always return a non-negative

value with lower values signifying greater similarity between the two PDFs.

Using the Adam optimizer, each model was implemented, trained, and tested using version 2.10.0 of the TensorFlow Python package. Each model was tested using various configurations of hyperparameters, with the configuration producing the lowest validation error deemed the best configuration for that architecture. The hyperparameters adjusted were the number of fully connected layers after the LSTM layer, the number of neurons in each dense layer and the sequence length. During training, the validation loss for each epoch was monitored and only when a new validation loss was lower than all preceding validation losses was the model saved, overwriting any older models. This procedure was set in place to prevent models that have been overtrained from being saved. Each model was tested by assuming a simulation had already been performed but aborted halfway. With this initial data, each model was tasked with predicting the PDF for each remaining time step of the simulation. To achieve this, the output of each time step is subsequently integrated into the input sequence used to predict the next time step. Because the number of steps the model needs to predict without any ground truth is unbounded, the prediction horizon H for this model in this testing scenario can be considered infinite. An infinite horizon test is necessary because of the target objective of this study. Suppose we want to apply a machine learning model to a simulation for which we have some initial simulation data to reduce computational costs. In that case, the model will not have ground truths to assist in its predictions beyond the initial data provided. The loss and metrics between the prediction and the ground truth were recorded and averaged over every predicted time step. The models weren't tested with previously unseen data because this would have further increased the difficulty. If a model cannot make accurate predictions for the data it was trained on, then we can assume that the model will perform poorly on previously unseen data as well. Once a model has proved to be accurate, it will be subjected to more difficult tests.

4.3 Results and Discussion

This section outlines four different model architectures and their outcomes when the previously mentioned procedure was carried out using CPMD-metadynamics simulation data for the solvent-based protonation reaction of HMF.

4.3.1 LSTM Autoencoder Architecture

After processing the CPMD-metadynamics dataset, there were still 5184 total input features across all three species participating in the reaction. Therefore, three separate LSTM autoencoders were proposed, one for each species. An autoencoder is a type of ANN that deconstructs inputs into a smaller feature space, often called the bottleneck layer or latent space, and then constructs an output based on the deconstructed features[60,61]. This type of architecture is used for this study as it allows multi-variate time-series data to be taken as inputs[30]. Ideally, the initial encoding portion of the autoencoder will further reduce the input feature size and extract critical features to use them to predict the feature space for the next time step. An additional benefit an autoencoder provides is that the trained encoder model can be saved and used to reduce the feature size of the inputs for the model that will predict the energetics of the reaction in a separate model. The dataset is first arranged into sequences of length l and passed into an LSTM layer with a dropout of $p = 0.1$. Dropout is a technique applied to neural networks where each neuron within the dropout layer has a probability p of temporarily being removed from the network. This technique is a simple way of preventing overfitting and exploring variations of the original network, which can be learned and combined to improve the network's performance[62]. The output of the LSTM is fed into a series of fully connected layers that form the rest of the encoder and decoder, with the final fully connected layer giving the prediction for the PDF of the next time step. The activation function used for the fully connected layers was the leaky rectified linear unit (Leaky ReLU) activation function (Equation 4.3), except for the final fully connected layer,

which used a softmax activation function (Equation 4.4)[63]. Unlike the rectified linear unit (ReLU) function (Equation 4.2), the Leaky ReLU function still returns a value even when the input is negative. When the input is negative, the output value is partially determined by the coefficient α, which is an adjustable hyperparameter. For the purposes of this work, α will always be 0.01. The Leaky ReLU function was chosen for its simplicity and immunity to the dying ReLU problem that ReLU functions occasionally suffer from. In a situation where the inputs for a neuron happen to be negative, a ReLU activation function will only return zero. Since its gradient will also be zero in this scenario, its weights will not change during backward propagation, and the neuron will keep outputting zeros[64]. The softmax activation function converts an array of values into an array of values that sum up to one. The converted values will also only fall between zero and one. It is these two properties that ensure the output of the model is always a PDF.

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{4.2}$$

$$LeakyReLU(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{4.3}$$

$$f(x_i) = \frac{e^{x_i}}{\Sigma_{j=1}^{N} e^{x_j}} \quad (i = 1, 2, \ldots, N) \tag{4.4}$$

Figure 4.7 represents a simplified outline of the architecture of the LSTM autoencoder used for training. Table 4.1 shows the training and testing results for each species and Figure 4.8 compares the predicted and ground truth PDFs for the reactants, water and DMSO at a selected test time step when an LSTM autoencoder architecture is used. The loss function used was the KL divergence. The hyperparameters adjusted for the training of this model are the sequence length, the number of neurons in the LSTM and fully connected layers and the number of the fully connected layers.

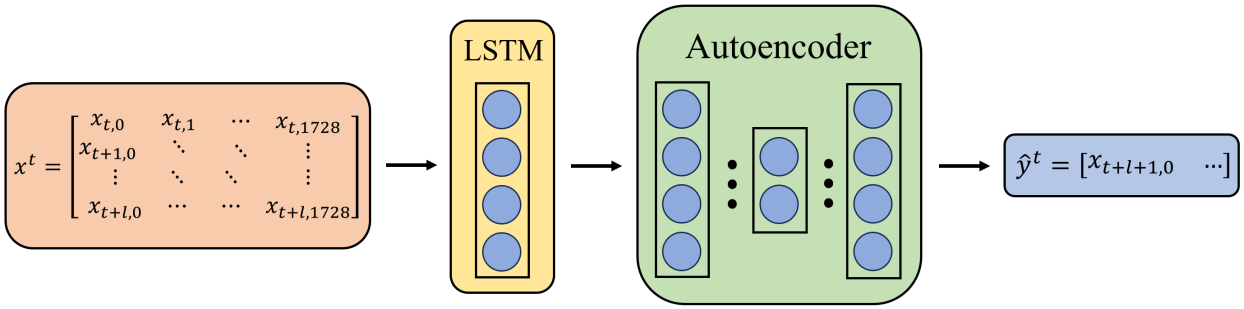
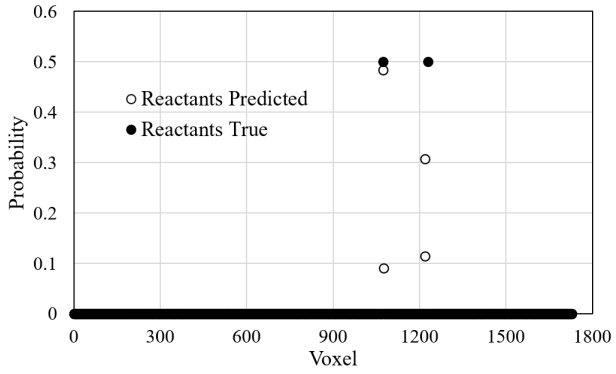


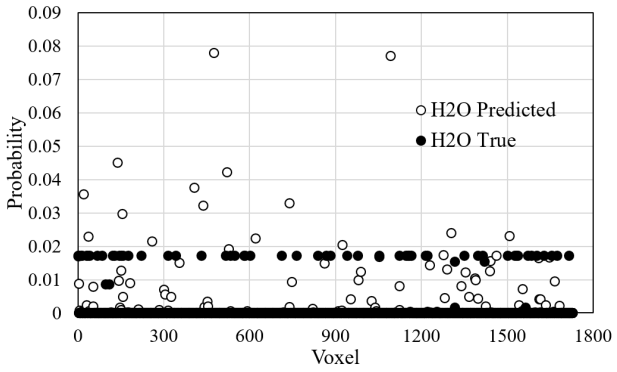
Figure 4.7: Model architecture of the LSTM autoencoder.

Table 4.1: Training and testing results for the 3 LSTM autoencoder model.

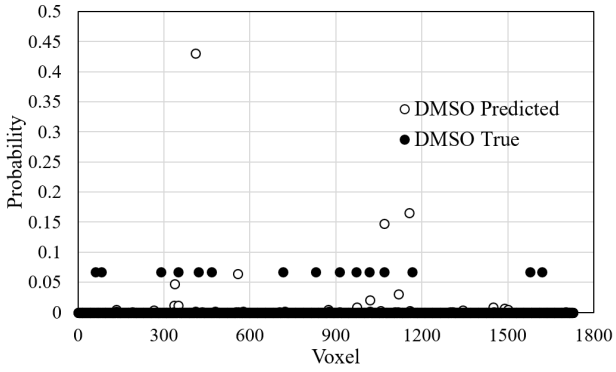
Species	Train KL Divergence	Validation KL Divergence	Avg. Test KL Divergence
Reactants	0.837	0.866	15.431
Water	0.864	1.486	9.947
DMSO	0.848	1.504	11.843



(a)



(b)



(c)

Figure 4.8: Predicted versus actual PDF of the reactants, water and DMSO for a selected test time step when using an LSTM autoencoder architecture.

When two PDFs are very similar, their KL divergence is extremely small on the count of the KL divergence formula containing the natural logarithm of the ratio between two probabilities. This tells us that the values within Table 4.1 are too large to indicate that the models are accurate, especially during testing. Upon further examination, this poor performance could be attributed to the sparsity of input data, as indicated by the large number of ground truth points with a value of 0. Using DMSO as an example, there are a total of 9 DMSO molecules within the simulation cell and assuming they all move one voxel during the 10-time frame period each PDF entails, then there will be 18 out of 1728 voxels (1%) containing a non-zero probability. The uneven distribution of the data makes it difficult for the model to give attention to the voxels with non-zero probability.

4.3.2 3D Convolutional neural network LSTM autoencoder architecture

A convolutional neural network (CNN) is a type of neural network model that uses the convolution operation to identify spatially or temporally correlated data locally within a dataset. The ability to extract local correlations within the data is why CNNs excel at tasks that involve spatiality[65]. If the flattened PDFs of the CPMD-metadynamics simulation are reverted to their 3D state, a CNN might be able to capture the important features within the PDFs and outperform the LSTM autoencoder model. Like the LSTM autoencoder model, the 3D PDFs will be arranged into sequences of a certain sequence length and then act as the input for multiple time-distributed 3D CNN layers. Because the input data is now four-dimensional (4D) with three spatial components and a temporal component, time-distributed 3D CNN layers are needed to perform 3D convolution on each 3D PDF within an input sequence. After four time-distributed 3D convolution layers, the resulting output is flattened and fed into an LSTM layer. A bottleneck layer, a fully connected layer and three 3D deconvolution layers follow the LSTM layer to complete the autoencoder. Figure 4.9

shows a rough outline of the architecture described. Table 4.2 shows the training, validation and test results for each species and Figure 4.10 compares the predicted and the actual PDF for each species for a selected time step when a 3D CNN-LSTM architecture is used. KL divergence remained as the loss function used for this model, and each model predicts the spatial features of one species. The hyperparameters adjusted for the training of this model are the sequence length, the filter size of each convolution layer, the number of convolution layers, the number of filters for each convolution layer, the number of neurons in the LSTM and the size of the bottleneck layer.

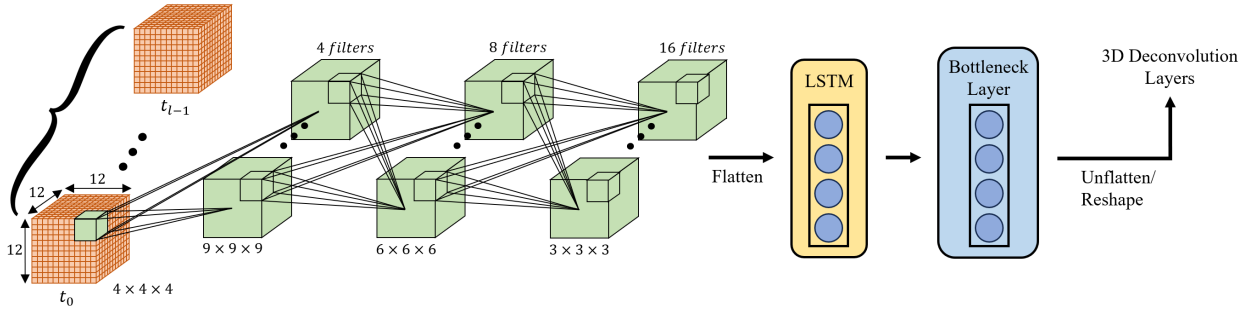


Figure 4.9: An example of a 3D CNN-LSTM model architecture that was tested.

Table 4.2: Training and testing results for the 3D CNN-LSTM autoencoder model.

Species	**Train KL Divergence**	**Validation KL Divergence**	**Avg. Test KL Divergence**
Reactants	0.574	0.575	15.430
Water	0.567	0.606	3.365
DMSO	0.521	0.632	5.580

Overall, both training and testing performance improved from the LSTM autoencoder models. Regardless, the KL divergence values are still too substantial to indicate that this model is accurate. It is hypothesized that predicting both the location and the magnitude of a handful of probabilities within a feature space of 1728 is too demanding. Therefore, the focus of future models was shifted to just predicting the location of the probability within the feature space. The location was prioritized over the magnitude because the position of solvent molecules relative to the reactants is crucial for the study of the impact of solvents

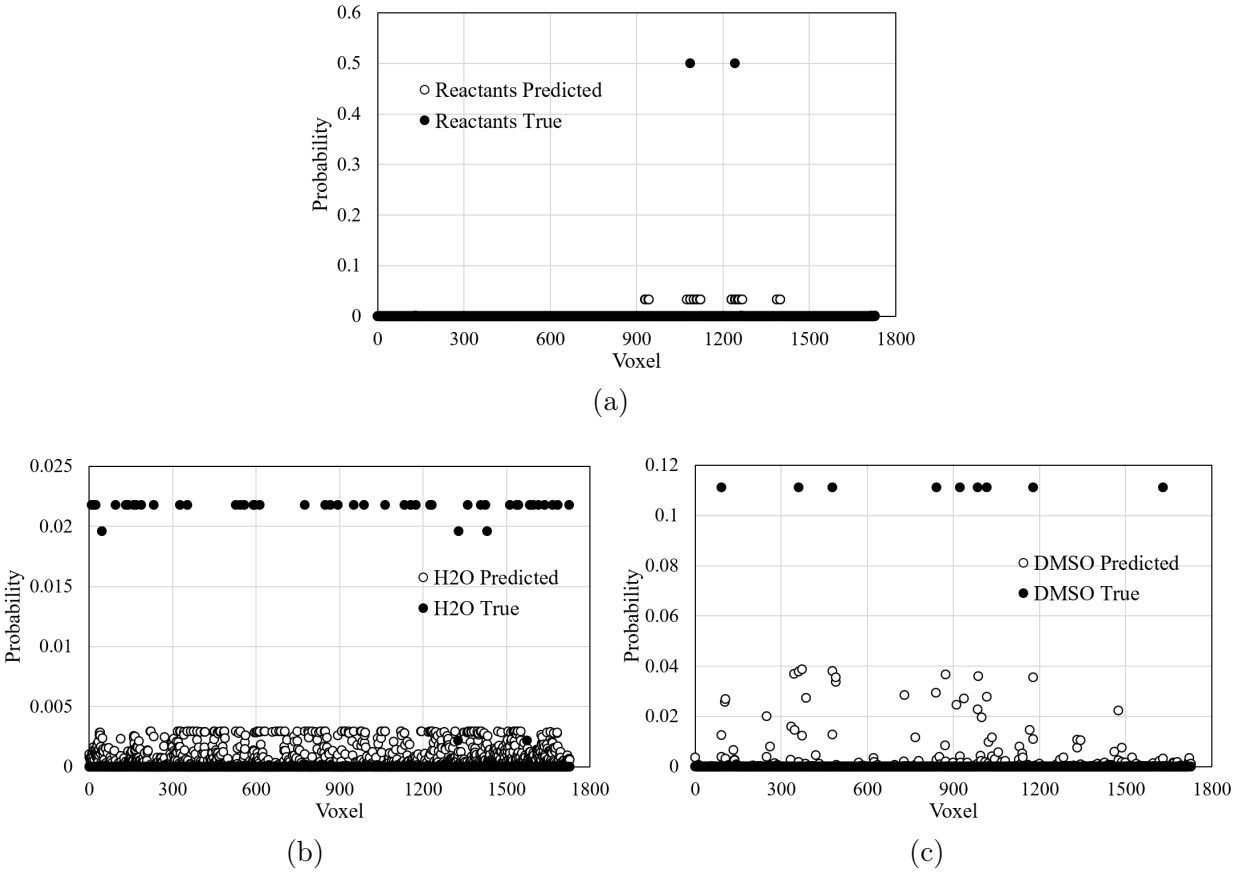


Figure 4.10: Predicted versus actual PDF of the reactants, water and DMSO for a selected test time step when using a 3D CNN-LSTM autoencoder architecture.

on reaction energetics regardless of magnitude, while the reverse is not true.

4.3.3 Binary relevance CNN-LSTM autoencoder classifier architecture

With greater emphasis on predicting the probability locations, the input data must be adjusted to reflect this objective. Any non-zero probability within the dataset was converted into a one while the zeros remained zero, creating a binary data set. This new model will individually assign either class 0 or class 1 to each voxel within every PDF it predicts, making it a binary relevance classifier. It will also use the same CNN-LSTM autoencoder architecture described in Section 4.3.2. Since the input data is now in a binary format rather than

a PDF, the loss function will change to a binary cross-entropy (BCE) function (Equation 4.5)[66]. BCE works well as a loss function but cannot give us an intuitive sense regarding how well the model is performing, much like the KL divergence. Binary classifiers often use binary accuracy as a metric to assess the accuracy of their model, which is simply the ratio between the sum of true positives and true negatives to the total number of predictions made. However, due to the skewed nature of the training data set, this metric could be misleading because even if the model were to assign 0 to every voxel, its binary accuracy would still appear to be very high owing to the scarcity of class 1 voxels within the ground truth as proven in Section 4.3.1.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{4.5}$$

For skewed binary datasets, the F1 score (Equation 4.6) is a more reliable metric to base the assessment of a model on because it heavily penalizes a high ratio of false positives to actual positives, which would be the case if the model assigned a 0 to every voxel. Therefore, the F1 score will be used to evaluate models on their performance rather than binary accuracy. The F1 score is the harmonic mean between precision (Equation 4.7) and recall (Equation 4.8) and returns a value between 0 and 1 with higher values associated with greater accuracy. Table 4.3 shows the training and testing results of the binary relevance classifier model, and Figure 4.11 shows the confusion matrices for a selected test time step for each species. Each species of the reaction system had its own model to keep the three channels of information separate. The hyperparameters adjusted for the training of this model are the sequence length, the filter size of each convolution layer, the number of convolution layers, the number of filters for each convolution layer, the number of neurons in the LSTM and the size of the bottleneck layer.

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.6}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.8}$$

TP = true positive FP = false positive FN = false negative

Table 4.3: Training and testing results for the binary relevance 3D CNN-LSTM classifier model.

Species	**Train BCE**	**Train F1 score**	**Validation BCE**	**Validation F1 score**	**Avg. Test BCE**	**Avg. Test F1 score**
Reactants	0.019	0.00	0.018	0.00	0.018	0.00
Water	0.010	0.82	0.012	0.27	0.460	0.34
DMSO	0.040	1.00	0.052	0.36	0.154	0.10

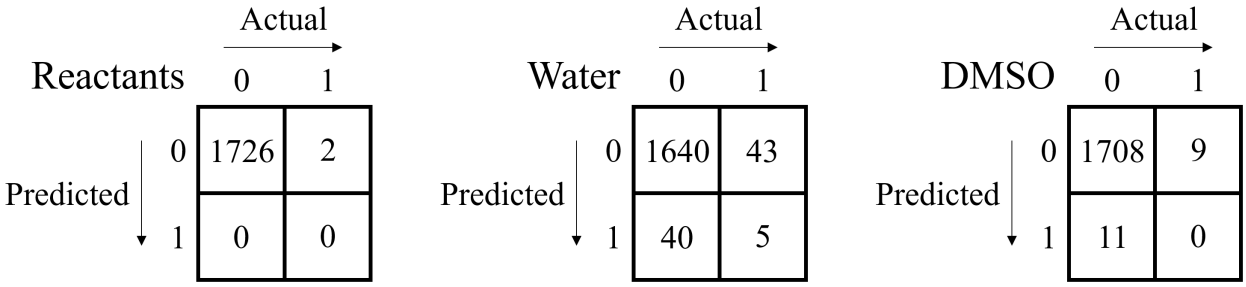


Figure 4.11: Confusion matrices of the predicted vs actual classes of each voxel for a selected test time step for reactants, water and DMSO based on the binary relevance CNN-LSTM autoencoder models.

Since the loss function and metric used for this model differ from previously tested models, a direct comparison cannot be made with previous models. However, based on the low F1 scores for training, validation and testing, we can still conclude that these models also perform poorly. The water and DMSO models produced good results during training; however, that success was not carried over during validation and testing. The F1 scores also vary across species. A plausible hypothesis for this would be that because of the relatively high number of water molecules within the system, it is easier to obtain true positives, even if by chance, than if there are very few molecules, such is the case with the reactants. To improve this model, a strategy to deal with the sparsity in the dataset is required. To prove that

sparsity is the primary issue, a total of 4 artificial datasets were created and used as training data with the binary relevance classifier model without any modifications to architecture or hyperparameters to prove that sparsity is indeed what is causing the poor performance of the models. The first artificial dataset contained a $12 \times 12 \times 12$ tensor of alternating 0s and 1s, creating evenly balanced classes. Every 10 time steps, every voxel would flip to the opposite class. The second artificial dataset contains the same distribution of 0s and 1s as the first artificial dataset but contains a different time pattern (a-a-a-b-b-b-b-b-a-a-a-a-a-a-a-a-b-b-b-b or 3a-5b-8a-4b). The third artificial data set contains 1s in three "hollow cubes" of different sizes to mimic hypothetical solvation shells. This was done in a specific manner so that the distribution of class 0 and class 1 was still roughly even. Every 10 time steps, the 0s and 1s would once again switch. The fourth artificial dataset maintained the "hollow cubes" pattern but this time with the 3a-5b-8a-4b time pattern. The training and testing results for all four artificial datasets are shown in Table 4.4. As shown by the results in Table 4.4, the same model architecture performs perfectly for all four artificial datasets, which supports the hypothesis that the problem with the CPMD-metadynamics data is that it is a heavily imbalanced dataset.

Table 4.4: Training and testing results for the binary relevance 3D CNN-LSTM classifier model on four different artificial datasets.

Species	**Train BCE**	**Train F1 score**	**Validation BCE**	**Validation F1 score**	**Avg. Test BCE**	**Avg. Test F1 score**
1	0.000	1.00	0.000	1.00	0.000	1.00
2	0.001	1.00	0.001	1.00	0.000	1.00
3	0.001	1.00	0.001	1.00	0.001	1.00
4	0.000	1.00	0.000	1.00	0.000	1.00

4.3.4 Binary relevance CNN-LSTM autoencoder classifier architecture with a weighted loss function

Naturally, with the skewed nature of the data, the next course of action would be to introduce a weighted loss function. A weighted loss function will emphasize predicting true positives by heavily penalizing false negatives, thereby increasing the F1 score. The model architecture follows the same architecture outlined in Sections 4.3.2 and 4.3.3, with only the loss function changing. Two different methods of using weighted loss functions were tested. The first method involves calculating a weighting tensor that will apply to each BCE calculation. This weighting tensor is calculated by first identifying the voxels that contain a probability for at least one timestep at some point during the simulation. These voxels are deemed critical and assigned a weight greater than 1. This weight is different for each species and is calculated by dividing the total number of voxels by the number of critical voxels within the input data for that species (tabulated in Table 4.5), while the rest of the voxels were assigned a weight of 1. This information was used to create a weighted matrix, which would be multiplied element-wise inside the loss function with the matrix containing the element-wise BCE between the prediction and the ground truth. The mean of the resulting matrix is the final loss value. The second weighted loss function method involves calculating a different weighting tensor based on the ground truth for each prediction and using that tensor to find the final loss value. Table 4.6 shows the training and testing results for the model that uses the global weighting tensor, and Table 4.7 shows the training and testing results for the model that uses the "on-the-fly" weighting tensor.

Table 4.5: Number of critical voxels and the weights assigned to each species.

Species	**Number of critical voxels**	**Weighting**
Reactants	44	39.27
Water	1308	1.32
DMSO	516	3.35

Table 4.6: Training and testing results for the binary relevance 3D CNN-LSTM classifier model using a global weighted loss function.

Species	Train BCE	Train F1 score	Validation BCE	Validation F1 score	Avg. Test BCE	Avg. Test F1 score
1	0.017	0.97	0.030	0.96	0.697	0.00
2	0.020	0.99	0.323	0.27	0.673	0.32
3	0.007	1.00	0.159	0.36	0.395	0.11

Table 4.7: Training and testing results for the binary relevance 3D CNN-LSTM classifier model using an "on-the-fly" weighted loss function.

Species	Train BCE	Train F1 score	Validation BCE	Validation F1 score	Avg. Test BCE	Avg. Test F1 score
1	0.006	0.67	0.334	0.69	12.829	0.23
2	0.064	0.80	8.751	0.21	12.756	0.39
3	0.021	0.74	5.106	0.36	14.442	0.14

Figures 4.12 and 4.13 show the confusion matrices of a selected test time step for all three species for the global weighting tensor and the "on-the-fly" weighting tensor implementations, respectively. The hyperparameters adjusted for the training of this model are the sequence length, the filter size of each convolution layer, the number of convolution layers, the number of filters for each convolution layer, the number of neurons in the LSTM and the size of the bottleneck layer. The training and validation F1 score for the reactant species increased dramatically from the model that did not use a weighted loss function. Because only two molecules make up the reactant species, there are fewer voxels that will see a reactant molecule at some point during the simulation. Due to the calculation method of the weights, the critical voxels will have a much greater weighting than other voxels and increasing performance. However, there is a delicate balance between the weighting and how many voxels it is applied to. If a heavy weight is applied to too many voxels, then the effectiveness of the weighting goes down. This is the reason behind limiting the weighting for water and DMSO in both weighted BCE function methods. The test F1 score for each species across both methods shows that only the "on-the-fly" weighting calculation method increased the F1 score for the classification of reactant species voxels. While there was an improvement,

neither weighted BCE method facilitated the training of an acceptable ML for any of the three species.

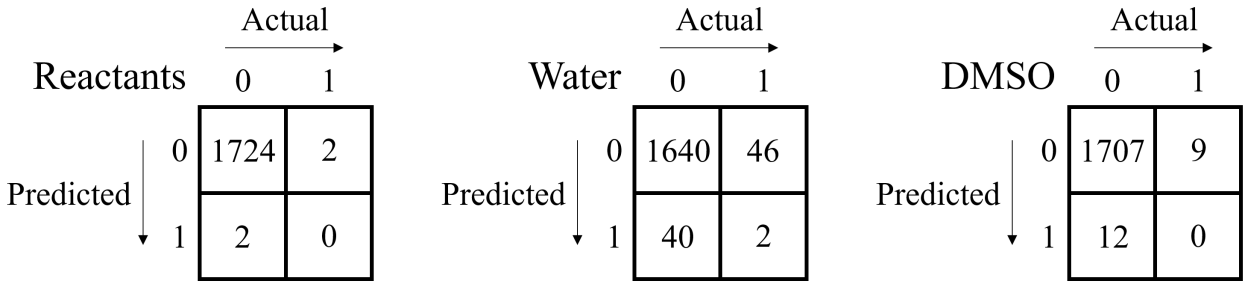


Figure 4.12: Confusion matrices of the predicted vs actual classes of each voxel for a selected test time step for reactants, water and DMSO based on the binary relevance CNN-LSTM autoencoder models with a global weighting tensor.

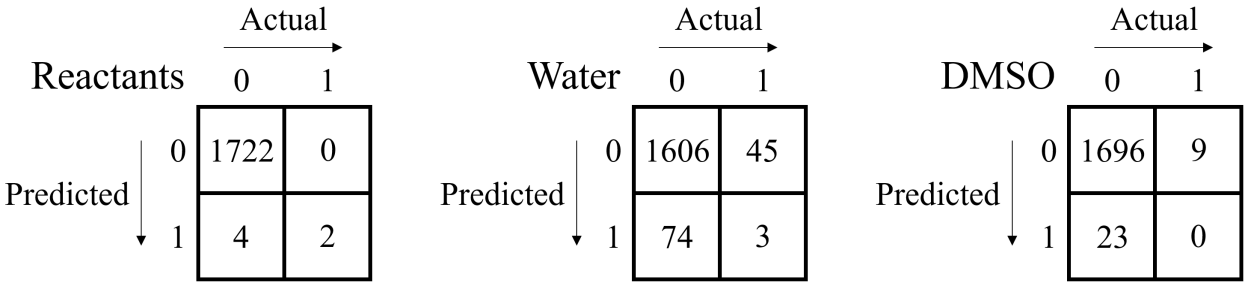


Figure 4.13: Confusion matrices of the predicted vs actual classes of each voxel for a selected test time step for reactants, water and DMSO based on the binary relevance CNN-LSTM autoencoder models with an "on-the-fly" weighting tensor.

4.4 Conclusions

In this chapter, five different models were trained and tested using CPMD-metadynamics trajectory data for the protonation reaction of HMF within a 50 wt% solution of water and DMSO. The LSTM autoencoder model processed sequences of flattened PDFs, encoding the information to predict subsequent PDFs, which gave disappointing results. To enhance the model's ability to capture important spatial features, convolution layers were used in conjunction with an LSTM layer for the next iteration of the model. The test results of the CNN-LSTM model show an improvement, but the loss was still too high to indicate an

accurate model. To simplify the task required of each model, the PDFs were converted into binary labels with a class 0 label indicating no probability and a class 1 label indicating a non-zero probability. Three binary relevance classifier models were evaluated: one with an unweighted loss function, one with a global weighted loss function calculated based on training data, and the other with an "on-the-fly" weighted loss function. Unfortunately, none of the three model architectures accurately predicted spatial features for all three species within the feature space. As proven in Section 4.3.3, the sparsity of the input data is likely making it difficult for these models to perform well. All four models discussed in Chapter 4 use an autoencoder structure to reduce the feature space of the input, so an alternative approach to feature reduction is needed. These alternative approaches will be discussed in Chapter 5.

Chapter 5

Alternate dimensionality reduction methods applied for time-series prediction of spatial features from CPMD with metadynamics simulation data

5.1 Introduction

Chapter 4 primarily focused on autoencoder architectures for making time-series predictions of probability distribution functions (PDFs) for each species within a reaction system. These predictions, if accurately representative of the ground truth, would serve as inputs for a separate model to predict the system's free energy at that specific time step within the collective variable (CV) space. However, the autoencoders proved ineffective, regardless of the architecture type used. An alternative approach involves applying a reversible transformation to the input PDFs before their use in training. The inverse transformation can be applied to

return the actual prediction following the predictions. This chapter outlines several candidates that meet this criterion and their performance and draws conclusions based on each model's outcomes.

5.2 Methods

Section 4.2 outlines the methodology employed to extract trainable data from the raw data files of the Car-Parrinello molecular dynamics (CPMD)-metadynamics simulation. The preprocessing yielded 20,000 flattened PDFs for each species: reactant, solvent and cosolvent. An applicable reversible transformation that can be applied to this dataset is principal component analysis (PCA). This technique projects a dataset onto a lower dimensional space consisting of a set of orthogonal axes called principal components. This process preserves any patterns or trends within the data[67]. PCA begins with the standardization of all values within the dataset and computing its covariance matrix. The eigenvectors of the covariance matrix indicate the principal components' directions, and the covariance matrix's eigenvalues indicate the amount of variance each principal component accounts for. The PCA transformation was executed automatically using the built-in PCA function within version 1.1.2 of the scikit-learn module in Python. The number of principal components used was determined by the minimum number of principal components needed to account for 99% of the total variance within the input dataset. Another transformation, dynamic principal component analysis (DPCA)[68], builds on standard PCA to incorporate both temporal and feature trends. DPCA involves creating a lagged matrix containing lagged features up to L past time steps alongside the non-lagged features. Standard PCA is then performed on the lagged matrix. Both transformations are viable options for addressing the sparsity within the data. Since the new input was transformed by PCA or DPCA, the long short-term memory (LSTM) model was trained to predict the principal components defined by the PCA or DPCA transformation. The output was multiplied by the transpose of the PCA or

DPCA transformation matrix to recover the real prediction.

Another reversible transformation involves eliminating voxels within the dataset that do not contribute useful information when making time-series predictions. If a voxel contained a probability of zero for the entire simulation, they were considered a "dead voxel" and eliminated from that species' dataset entirely. This allowed the LSTM model to receive fewer input features while predicting fewer outputs. These eliminated voxels were tracked and can be used to restore the full PDF if needed. This approach substantially reduced the feature space for all three species, with their feature spaces becoming 44, 1308 and 516 in size for the reactants, solvent and co-solvent species, respectively. This transformation has the largest impact on the reactants; therefore, it is expected that this model will perform the best with the reactant species.

A different approach to reducing input features is to abandon the idea of predicting PDFs and instead focus on predicting the Cartesian coordinates for each molecule within the simulation cell[69]. To simplify the task further, only the coordinates of one atom within each molecule were saved as training data. For the reactants, these atoms were the proton and the fourth carbon in HMF that is the bonding site for the proton for this reaction. For the solvent and cosolvent molecules, these were the oxygen atom in water and the sulphur molecule in dimethyl sulphoxide (DMSO). These atoms were chosen based on their importance to the reaction being studied or their location within their molecule. Once applied to the input data, the feature space for the reactant solvent and cosolvent species was 6, 126 and 27, respectively, representing a 99.7%, 92.7% and 98.4% reduction in feature space. With the modifications to the input dataset, the outputs of the LSTM were Cartesian coordinates instead of a PDF. The simulation data was also down-sampled by a factor of 10 so that the number of total samples within the data set remained $n = 20,000$.

The training and testing methodology discussed in Section 4.2 will remain as the methodology used to train and evaluate the models in Chapter 5.

5.3 Results and Discussion

This section will outline three approaches to feature reduction and their impact on model performance when applied to the CPMD-metadynamics simulation dataset for the condensed phase protonation reaction of 5-hydroxymethylfurfural (HMF).

5.3.1 Principal component analysis and dynamic principal component analysis for feature reduction

Before training any models, the input dataset was first transformed using PCA. To account for 99% of the total variance, the principal components used for the reactant, solvent and cosolvent species were 29, 848 and 390, respectively. Sequences of principal components were used to train the LSTM model. The model architecture consists of an LSTM layer with a dropout of 0.1 followed by fully connected layers that use the Leaky ReLU activation function with an α of 0.01. Table 5.1 shows this model's training and testing results, and Figure 5.1 compares the predicted and ground truth PDFs for a selected test time step across all three species. Once the inverse PCA transformation was applied to the predictions for this model, the result is still PDFs. Therefore, the Kullback-Leibler divergence (KL divergence) was used as the loss function for this model. The hyperparameters adjusted for the training of this model were the sequence length, the number of neurons in each layer and the number of fully connected layers.

Table 5.1: Training and testing results for the LSTM model using PCA-transformed data.

Species	**Train KL Divergence**	**Validation KL Divergence**	**Avg. Test KL Divergence**
Reactants	0.0005	0.0003	12.753 (0.748 - 16.855)
Water	0.0011	0.0012	2.563 (1.771 - 3.113)
DMSO	0.0011	0.0014	9.917 (1.387 - 16.419)

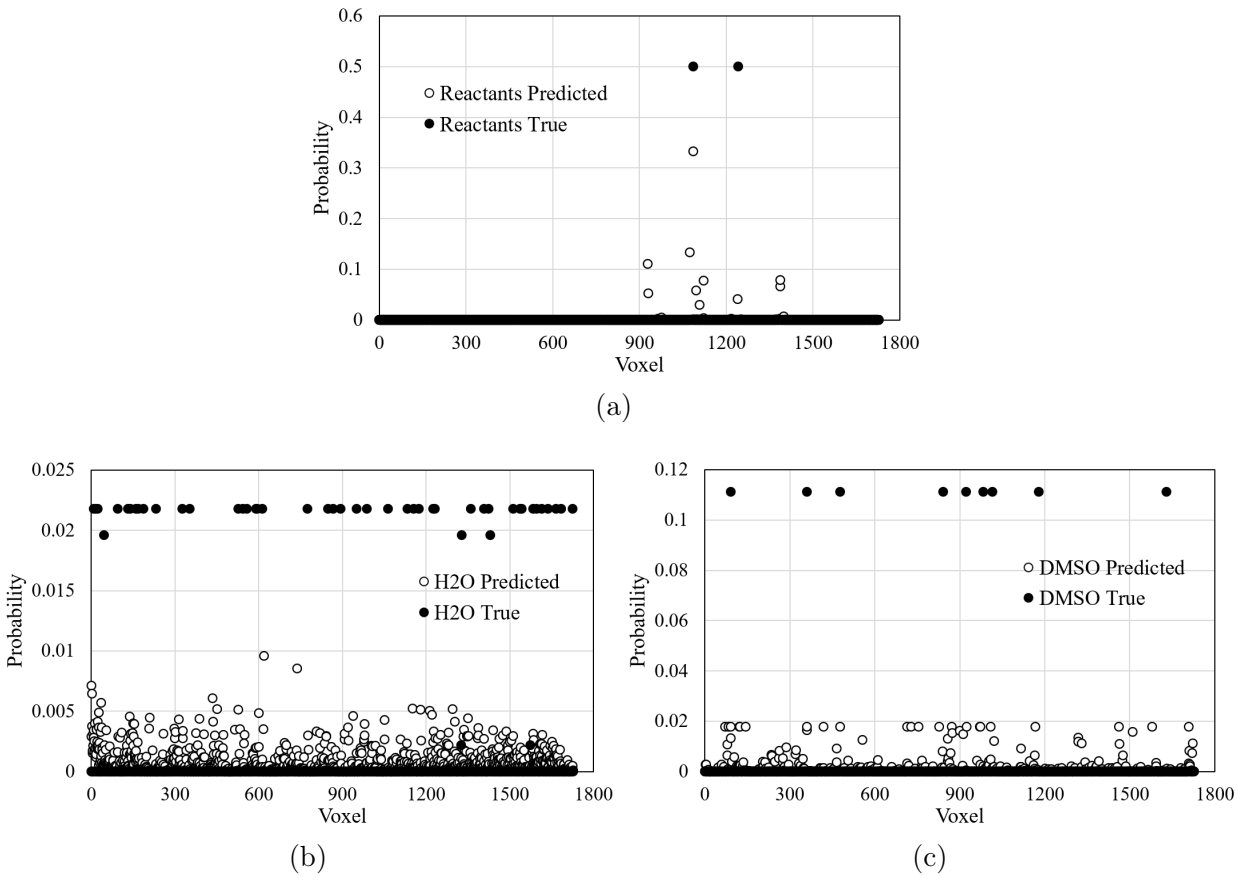


Figure 5.1: Predicted versus actual PDF of the reactants, water and DMSO for a selected test time step when using PCA transformation on input data.

Table 5.1 shows that the KL divergence values for training and validation for this model are now on the magnitude expected of a valid model. Unfortunately, this performance did not translate to testing. A range of KL divergence values is also given for testing in Table 5.1 to convey the wide range of loss values seen during testing. The lowest KL divergence within the range always comes from the model's first few predictions, with the loss values steadily increasing with each time step predicted. The underlying cause is due to the model making predictions based on an infinite horizon. The first prediction is likely the most accurate as the model has been provided with a sequence of ground truths for the first prediction. This prediction will naturally carry some error, and this error will propagate since this prediction will be a part of the input for the prediction of the next time step. This process repeats as prediction error within each prediction will continually compound.

Disregarding the compounding error issue, the KL divergence for the initial prediction for each model is three orders of magnitudes larger than it was during training and validation. Normally, the disparity between training and testing is the result of overfitting[70], data mismatch, over-tuning of hyperparameters[71] or a test set that is too small. Overfitting was prevented during training by only saving models that improved the validation loss, and a data mismatch is impossible as the same data was used for training and testing. Over-tuning of hyperparameters can happen when the hyperparameters are tuned towards the validation data; however, this is impossible since the validation data and testing data match. While the total number of time steps predicted by the model during testing is 10,000, which cannot be considered small, only one time step was predicted with purely ground truth values. It is possible the poor performance in predicting the first step was an outlier, and unfortunately, that error was propagated through the rest of the predictions. To investigate the validity of this hypothesis, each trained model was used to perform a one-step horizon prediction for every input within the training dataset. The range of KL divergences for these predictions for each species is tabulated in Table 5.2.

Table 5.2: Testing results for the LSTM model using PCA-transformed data with a prediction horizon of $H = 1$.

Species	**Minimum KL Divergence**	**Maximum KL Divergence**
Reactants	0.032	10.921
Water	1.485	2.594
DMSO	1.040	3.816

As the results show, the cause of the poor performance cannot be attributed to the first prediction within an infinite horizon prediction scheme being an outlier. The minimum KL divergence for the reactant species is lower than for the solvent and cosolvent, but it is still two orders of magnitude greater than what was seen in training and validation. Therefore, the reasons behind the mismatch between model performance are currently unknown.

DPCA incorporates past information into the PCA procedure to better capture temporal

trends. This characteristic of DPCA may allow it to be more effective than standard PCA when used for time-series prediction. While the transformation method changed, the model's architecture remained the same. Table 5.3 shows this model architecture's training and testing results, while Figure 5.2 compares the predicted and the ground truth PDF of all three species for a given test time step. The lag was adjusted within the range of 5 to 15 alongside the rest of the hyperparameters that accompany the PCA model.

Table 5.3: Training and testing results for the LSTM model using DPCA-transformed data.

Species	**Train KL Divergence**	**Validation KL Divergence**	**Avg. Test KL Divergence**
Reactants	0.002	0.003	14.535 (0.255 - 16.623)
Water	0.009	0.009	9.776 (7.319 - 12.445)
DMSO	0.007	0.015	10.384 (1.430 - 14.465)

Table 5.3 and Figure 5.2 show that the DPCA models perform worse than the PCA models; therefore, no further investigation was done regarding DPCA. Because the PCA transformation is applied on a larger matrix in DPCA, the number of principal components needed to account for 99% of the variance is larger. On average, it was found that when the lag is set to five, DPCA resulted in roughly three times the number of principal components compared to standard PCA. If the lag is increased, so will the number of principal components, resulting in more input features and making it harder to predict all principal components accurately. This increase in principal components is hypothesized to be the reason behind why standard PCA performs better than DPCA for the purposes of this study. Because the PCA method gave the best overall results out of any model predicting a PDF as its output, its projections can be found in Appendix B.

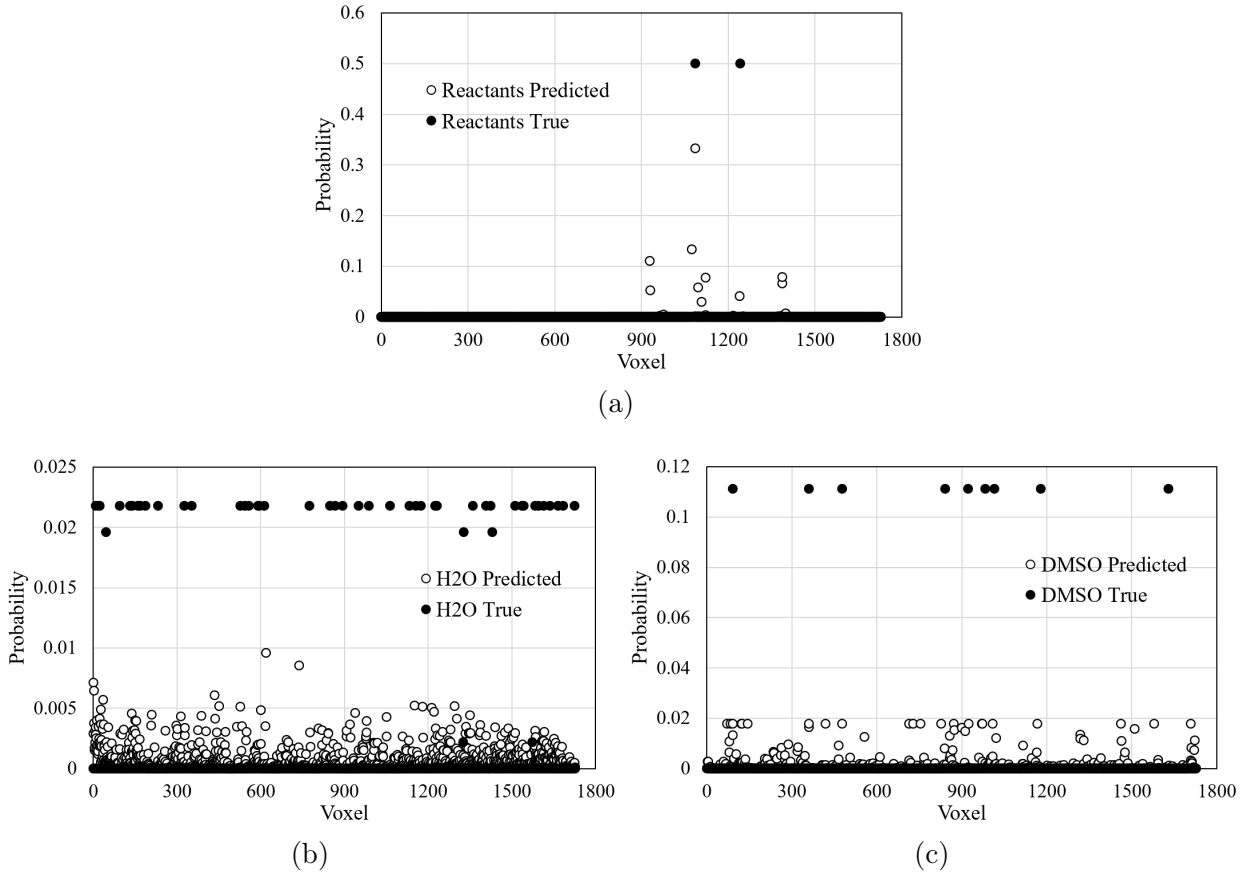


Figure 5.2: Predicted versus actual PDF of the reactants, water and DMSO for a selected test time step when using DPCA transformation on input data.

5.3.2 Feature reduction through elimination of negligible features used in conjunction with an LSTM

Looking at the PDFs generated from the simulation, one might notice that some voxels within the input data always show zero probability. These "dead voxels" can be calculated and predicted without a model because these voxels will always be associated with a zero probability. The location and number of "dead voxels" differ between species but can be easily calculated by taking the element-wise summation of all sample PDFs for a certain species and removing the voxels that have a sum of zero. Leaving these "dead voxels" inside the input data increases the feature space and gives the model more opportunities to make wrong predictions. The location of these "dead voxels" was recorded so that the original PDF could

be recovered. Because the voxels removed from each PDF only contain zeroes, the resulting inputs are still PDFs; therefore, the KL divergence is still viable as a loss function. However, as discussed at the end of Section 4.3.2, predicting the location of the probabilities within the PDFs is the most critical part. To set a more achievable target, the model was used to only predict the locations of the probabilities as a binary relevance classifier. Any non-zero probability within the input data set was labelled class 1, while all other voxels were labelled class 0. The loss function used was the binary cross-entropy (BCE) function, and the metric was the F1 score. Table 5.4 shows this model's training and testing results, and Figure 5.3 shows the confusion matrix between the predicted and actual classification of voxels for all three species. The hyperparameters adjusted for the training of this model are the lag of the lagged matrix, the sequence length, the number of neurons in each layer and the number of fully connected layers. The values in Table 5.4 do not include the "dead voxels" eliminated from the dataset at the beginning. Since they can be considered true positives as they are known to always be zero, the F1 scores within Table 5.4 are lower than if the F1 score was calculated with the full PDF. This was done so the known true positives would not inflate the F1 score.

Table 5.4: Training and testing results for the LSTM model using data free of "dead voxels".

Species	**Train BCE**	**Train F1 score**	**Validation BCE**	**Validation F1 score**	**Avg. Test BCE**	**Avg. Test F1 score**
Reactants	0.039	0.93	0.039	0.94	0.405 (0.013 - 0.475)	0.13 (0.00 - 1.00)
Water	0.049	0.90	0.052	0.90	0.731 (0.083 - 0.890)	0.12 (0.00 - 0.85)
DMSO	0.057	0.86	0.059	0.86	0.519 (0.030 - 0.721)	0.09 (0.00 - 0.95)

The results in Table 5.4 show that the model performs well during training and validation; however, the testing results show a wide range of values. No doubt this is because of the

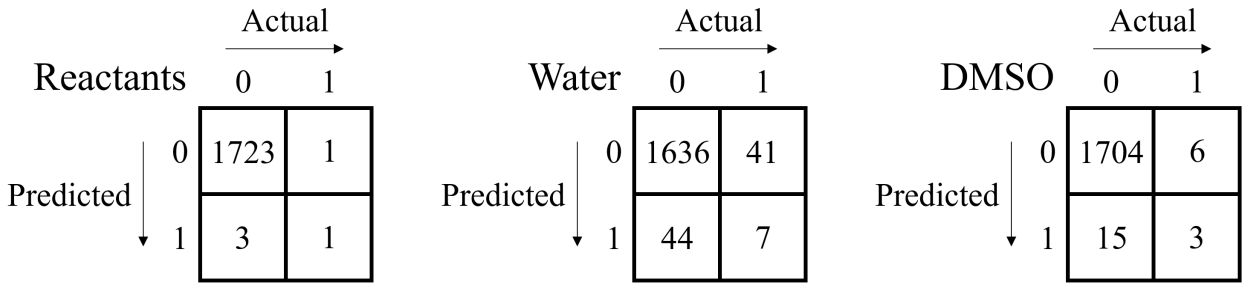


Figure 5.3: Confusion matrices of the predicted vs actual classes of each voxel for a selected test time step for reactants, water and DMSO based on the "dead voxel" feature reduction method.

infinite horizon prediction scheme. As discussed in Section 5.3.1, the error of one prediction gets carried onto the input for the next input and through this interaction, the errors within the input begin to compound with each other. During training, the prediction horizon H is always one, so the model learns to make one-step-ahead predictions very well, as shown in Table 5.4. This training method is called teacher forcing and translates poorly when applied to infinite horizon predictions. Teacher forcing is a training strategy for recurrent neural network (RNN) type neural networks that uses the ground truth as the input instead of the output of a previous time step. This is the default training method for an RNN or LSTM because it accelerates and stabilizes training, but the model can become reliant on having the ground truth as an input, which is exactly what happened to these models. Table 5.5 shows the average F1 scores from testing with different prediction horizons to determine the extent to which the trained models can make accurate predictions.

Table 5.5: Average F1 scores of the trained models when tested to make 10,000 time-series predictions with different prediction horizons H.

Species	**Avg. F1 score,** $H = 1$	**Avg. F1 score,** $H = 5$	**Avg. F1 score,** $H = 10$	**Avg. F1 score,** $H = 15$	**Avg. F1 score,** $H = 20$
Reactants	0.97	0.93	0.90	0.87	0.85
Water	0.90	0.85	0.79	0.76	0.72
DMSO	0.87	0.80	0.71	0.65	0.60

Table 5.5 shows that with increasing H, the performance of the models quickly deterio-

rates. The biggest weakness in these models is that ground truth will not be available in a real-life application to assist the model in its predictions. Another major weakness is that ground truth was used to determine which voxels were "dead voxels" and which voxels were important before the data was used to train the model. Combined, these weaknesses prevent this model from being applied to real-life situations. However, the model still proves that given the ground truth, time-series predictions can be made with acceptable accuracy.

5.3.3 Using Cartesian coordinates with an LSTM to predict the future locations of molecules within the system

If the goal is to reduce the number of features to a minimum, then one such solution is to use a different spatial representation than what has been considered so far, which is the PDF. The trajectory file output from the CPMD-metadynamics simulation gives the three Cartesian coordinates for each atom within the system for all $T = 200,000$ time steps. It is trivial to extract the coordinates of certain atoms into a dataset spanning all time steps. It would be inefficient to predict the coordinates of each atom within the system therefore, certain atoms were selected. For the reactants, this was the fourth carbon on the HMF molecule and the proton. For the solvent and cosolvent, it was the oxygen atom in water and the sulphur atom in DMSO, respectively. The total feature space across all species was now 159 compared to the 5184 features from the PDF approach. Similar to what was done with the PDFs, the dataset was then down-sampled by a factor of 10 for a total of $n = 20,000$ data points. This was necessary because each time frame represented an extremely small step in time, meaning the coordinates were roughly the same from one time step to the next. These tiny differences would be challenging to learn, whereas down-sampling the input data and magnifying the changes within the system slightly gives the model a higher chance of learning these trends. Sequences of these features were created and used as the training data for an LSTM network consisting of an LSTM layer and several fully connected layers with

Leaky Rectified Linear Unit (Leaky ReLU) activation functions. Because the input data is no longer in the form of PDFs, a new loss function was selected for this model. Mean average percentage error (MAPE) was chosen over root mean squared deviation (RMSD) as the loss function because the magnitude of values within the dataset is generally small. We aim to penalize small deviations with MAPE because these deviations can be significant relative to the magnitude of the dataset. Table 5.6 shows the training and testing results for this model. The hyperparameters adjusted for this model were the sequence length, the number of layers and the number of neurons in each layer. Similar to the model discussed in Section 5.3.2, the models showed excellent accuracy during training and validation with a wide range of losses during training due to the infinite horizon prediction testing method. Table 5.7 shows that the LSTM models that use the Cartesian coordinates as input data give more sustained accuracy with increasing prediction horizon compared to the models described in Section 5.3.2, particularly for the solvent and cosolvent.

Table 5.6: Mean average percentage error for training, validation and testing for LSTM using Cartesian coordinates dataset.

Species	**Train MAPE**	**Validation MAPE**	**Avg. Test MAPE**
Reactants	2.5%	2.1%	33.5% (2.2% - 42.8%)
Water	2.0%	1.9%	22.5% (1.7% - 35.9%)
DMSO	2.5%	2.2%	33.3% (2.3% - 46.2%)

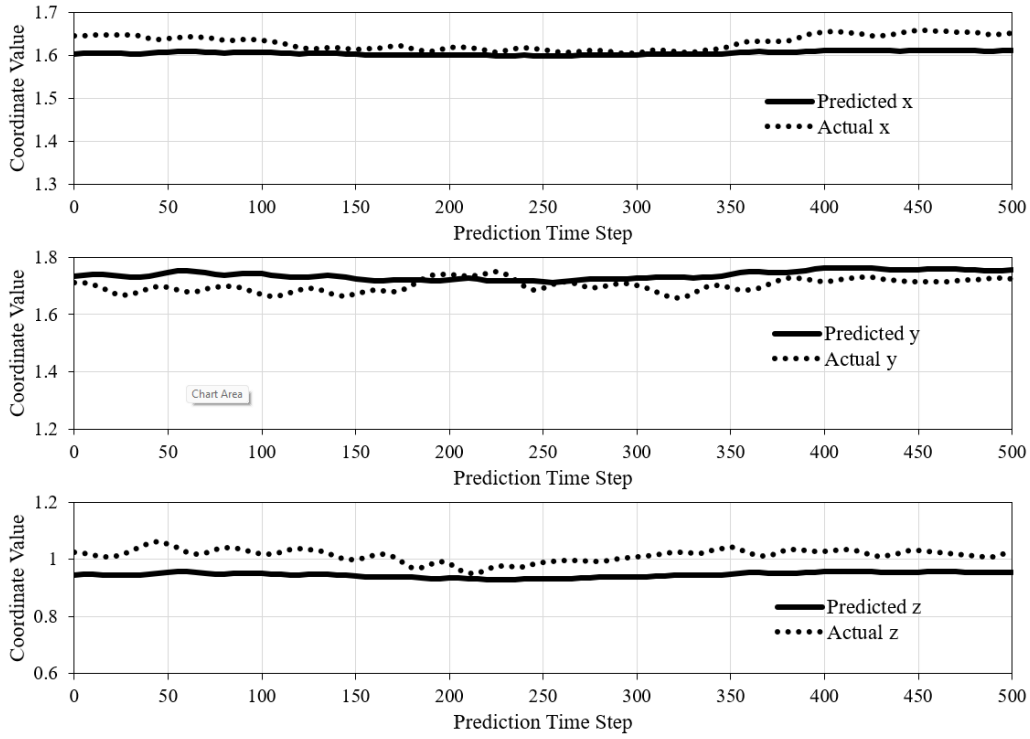
Table 5.7: Average MAPE of the trained models when tested to make 10,000 time-series predictions with different prediction horizons H.

Species	**Avg. MAPE,** $H = 1$	**Avg. MAPE,** $H = 50$	**Avg. MAPE,** $H = 100$	**Avg. MAPE,** $H = 200$	**Avg. MAPE,** $H = 300$
Reactants	2.0%	5.1%	8.9%	16.6%	22.0%
Water	1.6%	2.4%	3.4%	5.3%	7.1%
DMSO	1.7%	3.4%	5.4%	9.0%	11.8%

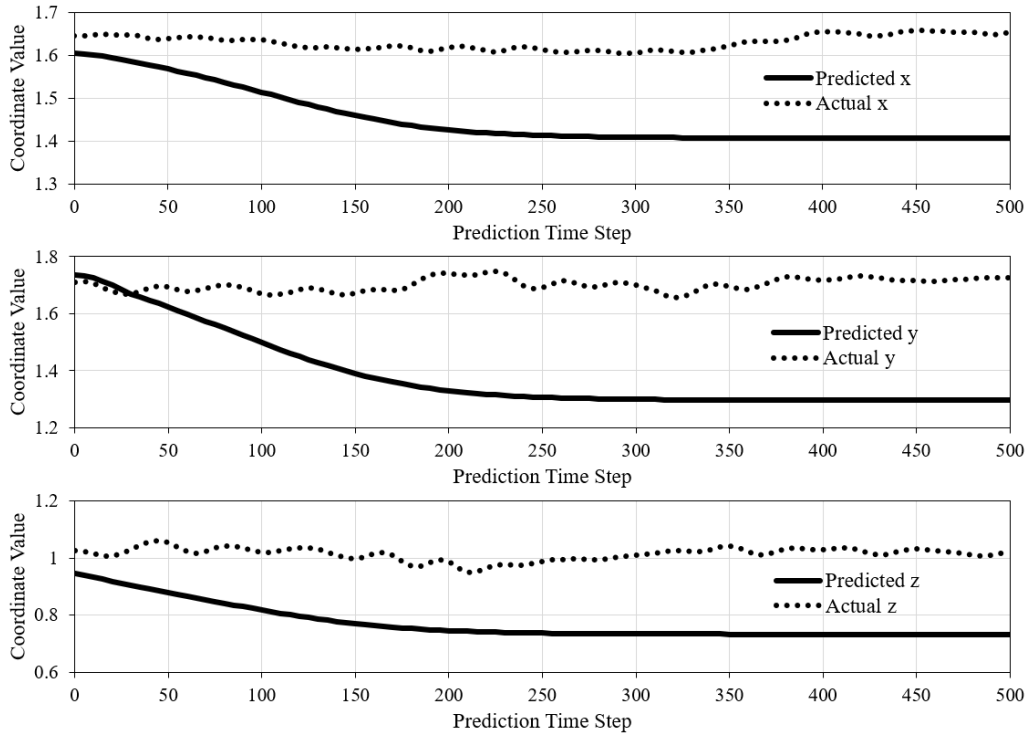
It also has the advantage of being more generalizable since these models do not require

ground truth information to further adjust the input data in some way before testing. Figure 5.4 compares the predicted Cartesian coordinates of the protonating hydrogen ion in the protonation reaction for two different prediction horizons. It is observed that within 25 prediction steps, the predicted Cartesian coordinates start deviating heavily from targets with no ground truth to set them back on course and eventually, the predictions plateau. This pattern was seen not just for the hydrogen ion but for all atoms predicted. Figure 5.4a indicates that the trained models perform well when given a test environment that emulates their training environment but falters when given longer prediction horizons, as shown by Figure 5.4b.

Both the wide range of test MAPE values observed in Table 5.6 and the comparison made in Figure 5.4 indicate that teacher forcing during training made the models too dependent on ground truths. Scheduled sampling is introduced into the training procedure to address this issue. Scheduled sampling is a training method that chooses if the ground truth or the predicted output is used for the next prediction at random. A teacher forcing ratio ϵ_E is used to determine whether the model uses the previous prediction or the ground truth as part of the next input for epoch E. The teacher forcing ratio can be conceptualized as the chance of a prediction step being teacher forced. If a randomly generated value is less than the teacher forcing ratio, the ground truth will be used as part of the next input, otherwise, the predicted output is used instead. When $\epsilon_E = 1$, teacher forcing will always enabled, whereas when $\epsilon_E = 0$, the model will train with solely self-generated outputs. The teacher forcing ratio is often determined by a function rather than a static value. Examples of such functions are linear, exponential decay and inverse sigmoid[72].



(a)

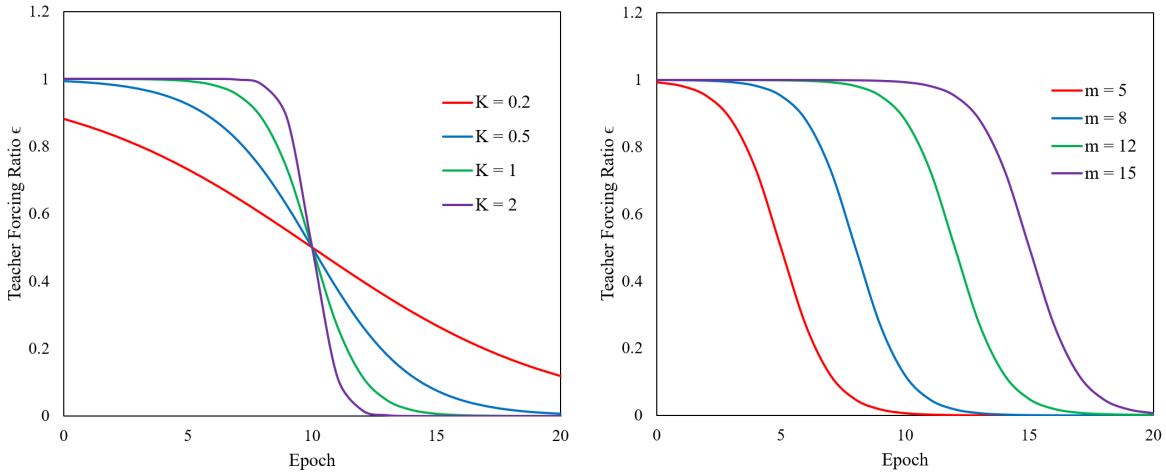


(b)

Figure 5.4: Predicted vs. actual Cartesian coordinates of protonating hydrogen ion (a) when the prediction horizon is 1 and (b) when the prediction horizon is infinite.

The inverse sigmoid function (Equation 5.1) was chosen because it gradually reduces ϵ_E initially to ensure training stability. Once deeper into training, ϵ_E starts to drop significantly until $\epsilon_E = 0$. The parameters K and m are hyperparameters adjusted to manipulate the shape of the inverse sigmoid curve depending on the training context. The variable K is a positive value that controls the steepness of the inflection point seen within any sigmoid function while m controls which epoch E the inflection point ($\epsilon_E = 0.5$) occurs at. Figure 5.5 demonstrates the effects of K and m on the inverse sigmoid function.

$$\epsilon_E = \frac{1}{1 + e^{K(E-m)}} \tag{5.1}$$



(a) (b)

Figure 5.5: The inverse sigmoid curve with (a) varying K values and (b) varying m values.

The training and validation methodology was also adjusted to properly implement scheduled sampling. Previously, each input within a batch predicts only one step and their mean loss is used for backpropagation; however, if only one prediction is made, then the decision of whether teacher forcing should be used or not is meaningless because a new batch of inputs will be used to calculate the next loss value. In other words, each input sequence must have a $H > 1$ for the teacher forcing decision to have an impact on training. This change is reflected in the new training loop, shown in Figure 5.6, where each input sequence of length

l makes multiple one-step-ahead predictions in a row. After each one-step-ahead prediction, a randomly generated value between 0 and 1 would be compared with the teacher forcing ratio ϵ_E for that epoch to determine which output (predicted or ground truth) is used for the next prediction.

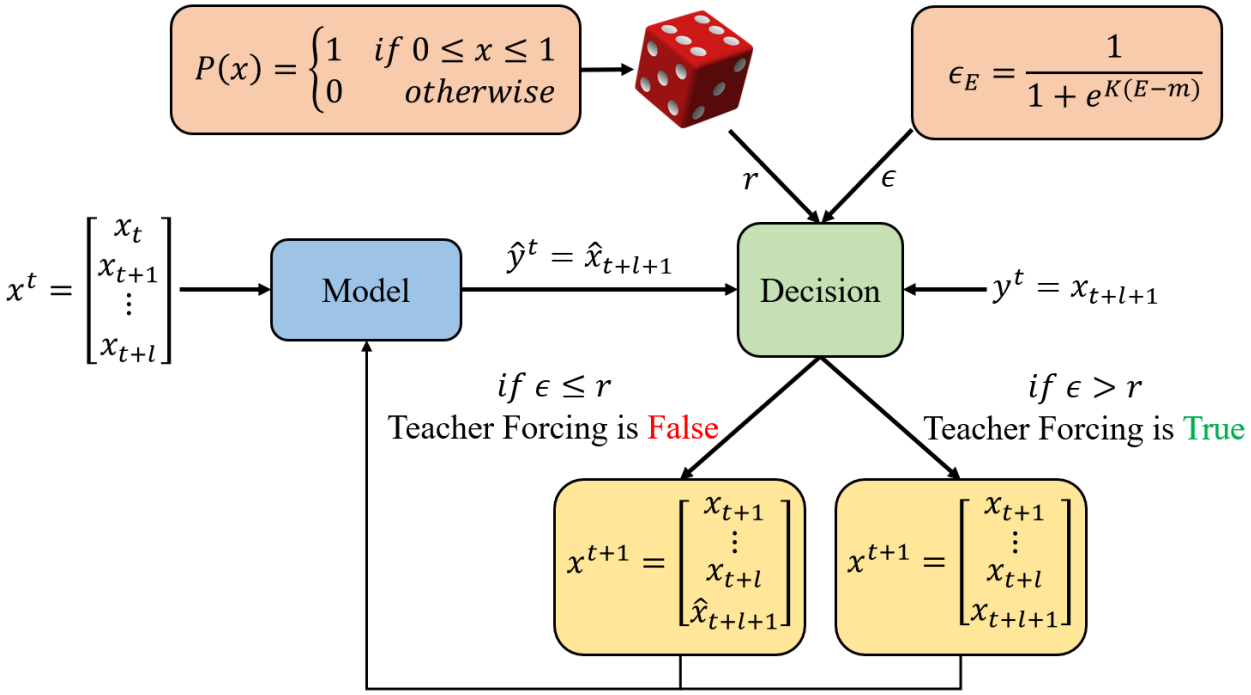


Figure 5.6: The new training loop used with the scheduled sampling technique. The number of times each training step goes through this loop is adjustable through a hyperparameter.

The prediction horizon H was always at least $l+1$. The motivation behind this constraint was to ensure that at least one prediction within every set of predictions would have a chance of containing only predicted outputs, which most similarly emulates an infinite prediction horizon scenario. To code and execute this custom training loop alongside a scheduled sampling teacher forcing scheme, version 1.12.1 of Pytorch was used. The hyperparameters adjusted for each model were the number of times the teacher forcing decision was for a single starting input, the K and m parameters within the inverse sigmoid function and the number of neurons in each layer. In theory, applying scheduled sampling should have improved each model's ability to make long-term predictions. However, Tables 5.8 and 5.9 show that using scheduled sampling only helped the HMF model improve, while the models

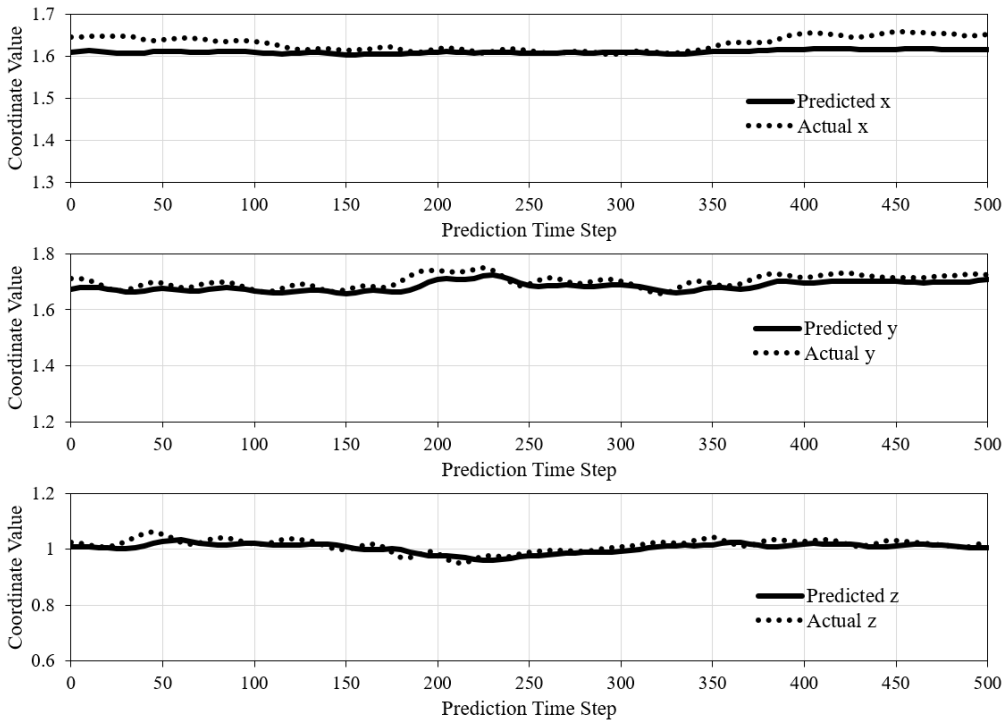
for water and DMSO performed worse. The improvement in MAPE for the reactants model is also highlighted in Figure 5.7a. Figure 5.7b indicates that despite employing the scheduled sampling method, the predictions still reached a plateau, albeit over a longer duration, suggesting that scheduled sampling is still insufficient.

Table 5.8: Mean average percentage error for training, validation and testing for LSTM trained using scheduled sampling with cartesian coordinates dataset.

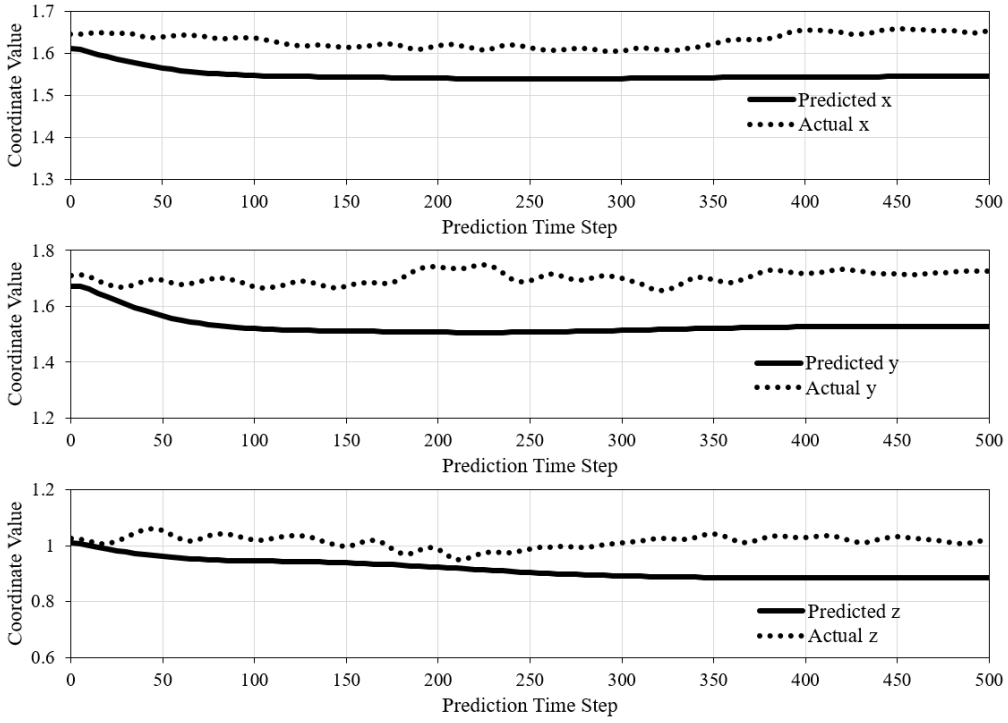
Species	**Train MAPE**	**Validation MAPE**	**Avg. Test MAPE**
Reactants	1.4%	1.7%	20.7% (1.3% - 31.7%)
Water	1.4%	2.5%	33.5% (3.8% - 44.8%)
DMSO	1.5%	2.7%	45.6% (2.5% - 55.1%)

Table 5.9: Average MAPE of the trained models when tested to make 10,000 time-series predictions with different prediction horizons H.

Species	**Avg. MAPE,** $H = 1$	**Avg. MAPE,** $H = 50$	**Avg. MAPE,** $H = 100$	**Avg. MAPE,** $H = 200$	**Avg. MAPE,** $H = 300$
Reactants	1.8%	6.0%	9.2%	13.1%	15.0%
Water	3.8%	5.1%	7.3%	10.5%	12.7%
DMSO	3.6%	8.9%	14.4%	23.4%	28.9%



(a)



(b)

Figure 5.7: Predicted vs. actual Cartesian coordinates of protonating hydrogen ion for the scheduled sampling model (a) when the prediction horizon is 1 and (b) when the prediction horizon is infinite.

5.4 Conclusions

Various feature reduction methods were applied to the sample data with varying levels of success in terms of model performance. The PCA transformation significantly improved training loss for each model; however, this improvement was not reflected in the validation loss. This result pointed to several possibilities, including overfitting, data mismatch, over-tuning of hyperparameters or a test set that is too small. All these possibilities were ruled out due to the testing method's nature, precautions made during training or further testing. When a DPCA transformation was applied, it decreased model performance since using DPCA resulted in more principal components. The following method tested involved identifying redundant features within the input data called "dead voxels" to shave down the feature space. Lastly, a new data set containing the Cartesian coordinates of each species within the simulation cell was used as a dataset. The models trained from this dataset gave promising results but again suffered from dependence on ground truth, albeit to a lesser extent than previously trained models. The scheduled sampling method was used to remedy this issue, but to no avail.

Unfortunately, no model discussed in this chapter could perform well under the proposed testing method despite some promising training results. The effects of teacher forcing were evident within the trained models even when scheduled sampling was applied, indicating that a different approach is required. Some strategies and approaches that could be taken toward training a model capable of predicting a CPMD simulation of a solvent-based reaction are discussed in Chapter 6.

Chapter 6

Conclusion and Future Work

6.1 Summary

This thesis outlined the process of training machine learning (ML) models that can act as a proxy model for Car-Parrinello molecular dynamics (CPMD) metadynamics simulations of condensed-phase biomass reactions. Chapter 2 reviews a handful of relevant literature, showcasing past work regarding the cooperation between ML and molecular dynamics (MD). These works portray the academic community's strong motivation to improve MD methods' computational efficiency to simulate larger systems and access longer time scales. Doing so would only serve to enhance our comprehension of computational chemistry. This work follows in these footsteps and attempts to push beyond what studies have achieved. Chapter 3 contains a high-level overview of the principles and concepts behind various molecular modeling methods. The intricacy of these methods, particularly *ab initio* molecular dynamics (AIMD) methods, is behind why ML is being leveraged to push MD beyond its current limits, which this study also attempts to do.

Chapter 4 uses data from an explicit solvation CPMD metadynamics simulation of HMF undergoing protonation in a 50/50 solution of water and dimethyl sulphoxide (DMSO) to train autoencoder models to predict the probability distribution function (PDF) of each

species within the system. The molecules within the system were split into three species: the reactants, water and DMSO. The output data from the simulation was converted into PDFs for each species, voxelized and constructed into sequences. These sequences served as training data as well as testing data. Testing data was the same as training data to ensure that the model at least had the proficiency to make predictions based on data it had already seen. Each model was tested by assuming that a CPMD metadynamics simulation had been run for 100,000 steps, and the proxy model was needed to predict the PDFs for the next 100,000 time steps. Because each PDF is constructed based on positional data from a time slice of 10 frames, the model would predict 10,000 PDFs that encapsulate the next 100,000 time steps. The first model architecture tested was a Long Short-Term Memory (LSTM) autoencoder architecture, with each species getting its own separate model. Each method described in this chapter refers to a set of three trained models, each trained to make predictions for one species. The second model architecture used a 3D CNN-LSTM autoencoder architecture to better capture spatial relationships within the PDFs. Both models failed to capture both the magnitude and the location of probabilities with each PDF. The attempted solution to this problem was a binary relevance 3D CNN-LSTM autoencoder that classifies voxels as either a class 1 (contains a non-zero probability) or a class 0 (no probability). This architecture was trained using three different loss functions: an unweighted binary cross-entropy (BCE) function, a weighted BCE function using a global weighting tensor and a weighted BCE function using an "on-the-fly" weighting tensor. Using a weighted loss function marginally improved testing results compared to when an unweighted loss function was used. Nonetheless, none of the models could accurately predict the location of probabilities over a 10,000 time step prediction horizon.

Chapter 5 tested five different ML models that don't use an autoencoder architecture. Instead, they employed feature reduction methods to make it easier for an LSTM neural network to learn the input feature space. The first techniques discussed were Principal Component Analysis (PCA) and Dynamic Principal Component Analysis (DPCA). Both

techniques involve projecting a dataset onto a lower dimensional feature space made up of mathematically determined principal components. The only difference is that DPCA applies the PCA transformation on a lagged matrix of the dataset to better capture temporal trends. It was found that the ML models did a good job training on PCA-transformed data however, this performance did not translate into the testing environment. DPCA, on the other hand, delivered poor results in both training and testing, most likely because DPCA inherently creates a larger feature space than PCA since it also captures time-dependent patterns. The third set of models was trained on PDFs stripped of their "dead voxels." "Dead voxels" refers to voxels within the dataset of each species that always give a probability of zero. Since the probability of these voxels never changes, there is no point in including them in the feature space. The disadvantage of this approach is that it requires knowledge of ground truth before the model is trained, which won't be the case in an application scenario. Nonetheless, this model could indicate what feature space size is learnable by the ML model. Despite promising training results and a significant reduction in input features, particularly for the reactants species, this set of models fails in a testing scenario. This observation can also be applied to the next set of models where the ML models were trained on Cartesian coordinates rather than PDFs. This pattern occurred because teacher forcing during training made these models dependent on ground truth, which wasn't available in the testing environment. Scheduled sampling was used to adjust the training loop for the ML models that use Cartesian coordinates as inputs to dampen the effect of teacher forcing but was met with very little success. In conclusion, no model architectures or dataset formulations resulted in a model that could reliably make accurate long-term predictions when no ground truth data is available.

6.2 Future Work

Given that the ML architectures explored within this work were unsuccessful at achieving the original objective, there are multiple directions for future work. These directions come in two varieties: adjustments to models already discussed within this work and architectures that approach the problem differently.

Throughout Chapter 5, the idea of feature reduction comes up regularly however, this idea was not as emphasized in Chapter 4. A possible feature reduction method that could be applied to the voxelized PDFs used as input data in Chapter 4 is to apply coarser voxelization or limit the range of atoms considered within the voxelization. Since the simulation cell is 3D, a slightly coarser voxelization along each axis would be compounded with each other to reduce feature size exponentially. Other possible modifications that apply to all models discussed include adding relational data into the dataset, bidirectional LSTMs (BLSTMs) [73], and self-tuning hyperparameter optimization [74]. Unfortunately, even if these adjustments successfully improved training results, they would not be able to address the larger underlying issue, which is how to decrease the model's reliance on ground truth data.

Since this study found that scheduled sampling was ineffective at reducing the effect of teacher forcing even when the training horizon was extended to emulate the testing environment, new strategies are required. Two such possibilities include increasing the role that molecular dynamics has on the ML model and attention mechanisms. In Section 5.3.3 the set of models trained using scheduled sampling was found to accurately predict each atom's Cartesian coordinates with a limited prediction horizon. Suppose the ML models had a limited prediction horizon, and CPMD uses the last ML-predicted coordinates to calculate several new time steps before feeding them back into the ML models. In that case, the overreliance on ground truth will no longer be an issue. This would increase the computational costs of this new hybrid CPMD-ML model but could deliver accurate predictions while still being less expensive than only using CPMD. A different approach would be to use

attention mechanisms to improve the ML models' ability to focus on important aspects of the input data to make accurate predictions. This is because attention mechanisms process inputs in an iterative manner while paying greater attention to information deemed more important[75]. It has already been shown that attention-based recurrent neural networks can make accurate multivariate time-series predictions[76]. Given the nature of the training data, this is promising, but only when it is put into action will we be able to see if this type of model can handle large feature spaces and limited ground truth availability in testing environments.

Once accurate time-series predictions can be made regarding the spatial features, a separate model can then be trained to make predictions of the collective variables and free energy of the system. This is so that a free energy surface (FES) can be reconstructed through the proxy models. The reconstructed FES will allow the energetics of the reaction in question to be studied. Unlike the ML model trained on spatial inputs, this new free energy model won't need to make time series-predictions. Instead, it will take each predicted spatial arrangement of the reactants and solvents and use that information to predict the CVs and free energy for that predicted step.

Bibliography

[1] David J. C. Constable, Conchita Jimenez-Gonzalez, and Richard K. Henderson. Perspective on Solvent Use in the Pharmaceutical Industry. *Organic Process Research & Development*, 11(1):133–137, January 2007. ISSN 1083-6160, 1520-586X. doi: 10.1021/op060170h. URL https://pubs.acs.org/doi/10.1021/op060170h.

[2] Jithin John Varghese and Samir H. Mushrif. Origins of complex solvent effects on chemical reactivity and computational tools to investigate them: a review. *Reaction Chemistry & Engineering*, 4(2):165–206, 2019. ISSN 2058-9883. doi: 10.1039/C8RE00226F. URL https://pubs.rsc.org/en/content/articlelanding/2019/re/c8re00226f.

[3] Alex K. Chew, Shengli Jiang, Weiqi Zhang, Victor M. Zavala, and Reid C. Van Lehn. Fast predictions of liquid-phase acid-catalyzed reaction rates using molecular dynamics simulations and convolutional neural networks. *Chemical Science*, 11(46):12464–12476, 2020. ISSN 2041-6520, 2041-6539. doi: 10.1039/D0SC03261A. URL http://xlink.rsc.org/?DOI=D0SC03261A.

[4] Samir H. Mushrif. *Molecular Modeling Methods*. Unpublished Notes for CME 694: Computational Chemistry & Molecular Modeling, University of Alberta.

[5] Jens Kleinjung and Franca Fraternali. Design and application of implicit solvent models in biomolecular simulations. *Current Opinion in Structural Biology*, 25:126–134, April 2014. ISSN 0959440X. doi: 10.1016/j.sbi.2014.04.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0959440X14000438.

[6] Alexey Onufriev. Chapter 7 - Implicit Solvent Models in Molecular Dynamics Simulations: A Brief Overview. In *Annual Reports in Computational Chemistry*, volume 4, pages 125–137. Elsevier, 2008. ISBN 9780444532503. doi: 10.1016/S1574-1400(08)00007-8. URL https://linkinghub.elsevier.com/retrieve/pii/S1574140008000078.

[7] Alex K. Chew, Theodore W. Walker, Zhizhang Shen, Benginur Demir, Liam Witteman, Jack Euclide, George W. Huber, James A. Dumesic, and Reid C. Van Lehn. Effect of Mixed-Solvent Environments on the Selectivity of Acid-Catalyzed Dehydration Reactions. *ACS Catalysis*, 10(3):1679–1691, February 2020. ISSN 2155-5435, 2155-5435. doi: 10.1021/acscatal.9b03460. URL https://pubs.acs.org/doi/10.1021/acscatal.9b03460.

[8] Jin Zhang, Haiyang Zhang, Tao Wu, Qi Wang, and David Van Der Spoel. Comparison of Implicit and Explicit Solvent Models for the Calculation of Solvation Free Energy in Organic Solvents. *Journal of Chemical Theory and Computation*, 13(3):1034–1043, March 2017. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.7b00169. URL https://pubs.acs.org/doi/10.1021/acs.jctc.7b00169.

[9] Nilesh Varadan Orupattur, Samir H. Mushrif, and Vinay Prasad. Catalytic materials and chemistry development using a synergistic combination of machine learning and ab initio methods. *Computational Materials Science*, 174:109474, March 2020. ISSN 09270256. doi: 10.1016/j.commatsci.2019.109474. URL https://linkinghub.elsevier.com/retrieve/pii/S0927025619307736.

[10] Zahra Shamsi, Kevin J. Cheng, and Diwakar Shukla. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. *The Journal of Physical Chemistry B*, 122(35):8386–8395, September 2018. ISSN 1520-6106, 1520-5207. doi: 10.1021/acs.jpcb.8b06521. URL https://pubs.acs.org/doi/10.1021/acs.jpcb.8b06521.

[11] Theodore W. Walker, Alex K. Chew, Huixiang Li, Benginur Demir, Z. Conrad Zhang, George W. Huber, Reid C. Van Lehn, and James A. Dumesic. Universal kinetic solvent effects in acid-catalyzed reactions of biomass-derived oxygenates. *Energy & Environmental Science*, 11(3):617–628, 2018. ISSN 1754-5692, 1754-5706. doi: 10.1039/C7EE03432F. URL http://xlink.rsc.org/?DOI=C7EE03432F.

[12] Venkatesh Botu and Rampi Ramprasad. Adaptive machine learning framework to accelerate *ab initio* molecular dynamics. *International Journal of Quantum Chemistry*, 115(16):1074–1083, August 2015. ISSN 0020-7608, 1097-461X. doi: 10.1002/qua.24836. URL https://onlinelibrary.wiley.com/doi/10.1002/qua.24836.

[13] Zhenwei Li, James R. Kermode, and Alessandro De Vita. Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces. *Physical Review Letters*, 114(9):096405, March 2015. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.114.096405. URL https://link.aps.org/doi/10.1103/PhysRevLett.114.096405.

[14] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the Kohn-Sham equations with machine learning. *Nature Communications*, 8(1):872, October 2017. ISSN 2041-1723. doi: 10.1038/s41467-017-00839-3. URL https://www.nature.com/articles/s41467-017-00839-3.

[15] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Physical Review Letters*, 120(14):143001, April 2018. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.120.143001. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.143001.

[16] Yaoyi Chen, Andreas Krämer, Nicholas E. Charron, Brooke E. Husic, Cecilia Clementi, and Frank Noé. Machine learning implicit solvation for molecular dynamics. *The Journal*

of Chemical Physics, 155(8):084101, aug 2021. ISSN 0021-9606, 1089-7690. doi: 10.1063/5.0059915. URL https://pubs.aip.org/aip/jcp/article/1018119.

[17] Michael Gastegger, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of solvent effects on molecular spectra and reactions. *Chemical Science*, 12(34):11473–11483, 2021. ISSN 2041-6520, 2041-6539. doi: 10.1039/D1SC02742E. URL http://xlink.rsc.org/?DOI=D1SC02742E.

[18] Hanwen Zhang, Veronika Juraskova, and Fernanda Duarte. Modeling Chemical Processes in Explicit Solvents with Machine Learning Potentials. preprint, Chemistry, July 2023. URL https://chemrxiv.org/engage/chemrxiv/article-details/64a8085fba3e99daefab8f89.

[19] Juben N. Chheda, George W. Huber, and James A. Dumesic. Liquid-Phase Catalytic Processing of Biomass-Derived Oxygenated Hydrocarbons to Fuels and Chemicals. *Angewandte Chemie International Edition*, 46(38):7164–7183, September 2007. ISSN 1433-7851, 1521-3773. doi: 10.1002/anie.200604274. URL https://onlinelibrary.wiley.com/doi/10.1002/anie.200604274.

[20] Samir H. Mushrif, Stavros Caratzoulas, and Dionisios G. Vlachos. Understanding solvent effects in the selective conversion of fructose to 5-hydroxymethyl-furfural: a molecular dynamics investigation. *Physical Chemistry Chemical Physics*, 14(8):2637, 2012. ISSN 1463-9076, 1463-9084. doi: 10.1039/c2cp22694d. URL http://xlink.rsc.org/?DOI=c2cp22694d.

[21] Stephen J. Smith and Brian T. Sutcliffe. The Development of Computational Chemistry in the United Kingdom. In Kenny B. Lipkowitz and Donald B. Boyd, editors, *Reviews in Computational Chemistry*, volume 10, pages 271–316. Wiley, 1 edition, January 1996. ISBN 9780471186489 9780470125878. doi: 10.1002/9780470125878.ch5. URL https://onlinelibrary.wiley.com/doi/10.1002/9780470125878.ch5.

[22] Jonathan P. McMullen and Klavs F. Jensen. Integrated microreactors for reaction automation: new approaches to reaction development. *Annual Review of Analytical Chemistry*, 3(1):19–42, June 2010. ISSN 1936-1327, 1936-1335. doi: 10.1146/annurev.anchem.111808.073718. URL https://www.annualreviews.org/doi/10.1146/annurev.anchem.111808.073718.

[23] Zhenpeng Zhou, Xiaocheng Li, and Richard N. Zare. Optimizing Chemical Reactions with Deep Reinforcement Learning. *ACS Central Science*, 3(12):1337–1344, December 2017. ISSN 2374-7943, 2374-7951. doi: 10.1021/acscentsci.7b00492. URL https://pubs.acs.org/doi/10.1021/acscentsci.7b00492.

[24] Max A. Mellmer, Chotitath Sanpitakseree, Benginur Demir, Peng Bai, Kaiwen Ma, Matthew Neurock, and James A. Dumesic. Solvent-enabled control of reactivity for liquid-phase reactions of biomass-derived compounds. *Nature Catalysis*, 1(3):199–207, February 2018. ISSN 2520-1158. doi: 10.1038/s41929-018-0027-3. URL https://www.nature.com/articles/s41929-018-0027-3.

[25] Wen Torng and Russ B Altman. High precision protein functional site detection using 3D convolutional neural networks. *Bioinformatics*, 35(9):1503–1512, May 2019. ISSN 1367-4803, 1367-4811. doi: 10.1093/bioinformatics/bty813. URL https://academic.oup.com/bioinformatics/article/35/9/1503/5104336.

[26] J Jiménez, S Doerr, G Martínez-Rosell, A S Rose, and G De Fabritiis. DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics*, 33(19):3036–3042, October 2017. ISSN 1367-4803, 1367-4811. doi: 10.1093/bioinformatics/btx350. URL https://academic.oup.com/bioinformatics/article/33/19/3036/3859178.

[27] Florian Häse, Ignacio Fdez. Galván, Alán Aspuru-Guzik, Roland Lindh, and Morgane Vacher. How machine learning can assist the interpretation of *ab initio* molecular dy-

namics simulations and conceptual understanding of chemistry. *Chemical Science*, 10 (8):2298–2307, 2019. ISSN 2041-6520, 2041-6539. doi: 10.1039/C8SC04516J. URL http://xlink.rsc.org/?DOI=C8SC04516J.

[28] Morgane Vacher, Anders Brakestad, Hans O. Karlsson, Ignacio Fdez. Galván, and Roland Lindh. Dynamical Insights into the Decomposition of 1,2-Dioxetane. *Journal of Chemical Theory and Computation*, 13(6):2448–2457, June 2017. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.7b00198. URL https://pubs.acs.org/doi/10.1021/acs.jctc.7b00198.

[29] Peter M. Williams. Bayesian Regularization and Pruning Using a Laplace Prior. *Neural Computation*, 7(1):117–143, January 1995. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1995.7.1.117. URL https://direct.mit.edu/neco/article/7/1/117-143/5830.

[30] Mohammad Javad Eslamibidgoli, Mehrdad Mokhtari, and Michael H. Eikerling. Recurrent Neural Network-based Model for Accelerated Trajectory Analysis in AIMD Simulations. *arXiv*, 1909.10124, 2019. doi: 10.48550/ARXIV.1909.10124. URL https://arxiv.org/abs/1909.10124.

[31] Anjana Thimmaiah Puliyanda. *Machine learning-based monitoring of complex reactive systems*. Doctor of Philosophy in Process Control, University of Alberta, Edmonton, 2022. URL https://era.library.ualberta.ca/items/1c56f686-fb63-43e6-a589-2418bd518301.

[32] Benjamin W. J. Chen, Xinglong Zhang, and Jia Zhang. Accelerating explicit solvent models of heterogeneous catalysts with machine learning interatomic potentials. *Chemical Science*, 14(31):8338–8354, 2023. ISSN 2041-6520, 2041-6539. doi: 10.1039/D3SC02482B. URL http://xlink.rsc.org/?DOI=D3SC02482B.

[33] Tetiana Zubatiuk and Olexandr Isayev. Development of Multimodal Machine Learning Potentials: Toward a Physics-Aware Artificial Intelligence. *Accounts of Chemical Research*, 54(7):1575–1585, April 2021. ISSN 0001-4842, 1520-4898. doi: 10.1021/acs.accounts.0c00868. URL https://pubs.acs.org/doi/10.1021/acs.accounts.0c00868.

[34] Siddhartha Laghuvarapu, Yashaswi Pathak, and U. Deva Priyakumar. BAND NN: A Deep Learning Framework for Energy Prediction and Geometry Optimization of Organic Small Molecules. *Journal of Computational Chemistry*, 41(8):790–799, March 2020. ISSN 0192-8651, 1096-987X. doi: 10.1002/jcc.26128. URL https://onlinelibrary.wiley.com/doi/10.1002/jcc.26128.

[35] Zachary D. Pozun, Katja Hansen, Daniel Sheppard, Matthias Rupp, Klaus-Robert Müller, and Graeme Henkelman. Optimizing transition states via kernel-based machine learning. *The Journal of Chemical Physics*, 136(17):174101, May 2012. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.4707167. URL https://pubs.aip.org/jcp/article/136/17/174101/191704/Optimizing-transition-states-via-kernel-based.

[36] Alexander V. Shapeev. Moment Tensor Potentials: A Class of Systematically Improvable Interatomic Potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, January 2016. ISSN 1540-3459, 1540-3467. doi: 10.1137/15M1054183. URL http://epubs.siam.org/doi/10.1137/15M1054183.

[37] Burr Settles. *Active Learning*, volume 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Springer International Publishing, 2012. ISBN 978-3-031-01560-1. URL https://www.springer.com/series/16915.

[38] Martin Karplus and J. Andrew McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9(9):646–652, September 2002. ISSN 10728368.

doi: 10.1038/nsb0902-646. URL https://www.nature.com/doifinder/10.1038/nsb0902-646.

[39] J. Andrew McCammon, Bruce R. Gelin, and Martin Karplus. Dynamics of folded proteins. *Nature*, 267(5612):585–590, June 1977. ISSN 0028-0836, 1476-4687. doi: 10.1038/267585a0. URL https://www.nature.com/articles/267585a0.

[40] Edgar F. Meyer, Stanley M. Swanson, and Jocylin A. Williams. Molecular modelling and drug design. *Pharmacology & Therapeutics*, 85(3):113–121, March 2000. ISSN 01637258. doi: 10.1016/S0163-7258(99)00069-8. URL https://linkinghub.elsevier.com/retrieve/pii/S0163725899000698.

[41] Zhenhua Yao, Chang-Chun Zhu, Min Cheng, and Junhua Liu. Mechanical properties of carbon nanotube by molecular dynamics simulation. *Computational Materials Science*, 22(3-4):180–184, December 2001. ISSN 09270256. doi: 10.1016/S0927-0256(01)00187-2. URL https://linkinghub.elsevier.com/retrieve/pii/S0927025601001872.

[42] Jacob R. Gissinger, Benjamin D. Jensen, and Kristopher E. Wise. Modeling chemical reactions in classical molecular dynamics simulations. *Polymer*, 128:211–217, October 2017. ISSN 00323861. doi: 10.1016/j.polymer.2017.09.038. URL https://linkinghub.elsevier.com/retrieve/pii/S0032386117309114.

[43] C. David Sherrill. Introduction to Molecular Mechanics. URL http://vergil.chemistry.gatech.edu/courses/chem6485/pdf/molmech-lecture.pdf.

[44] Jose Carlos Velasco Calderon. *Investigating acid-catalyzed biomass reactions and solvent effects in humins formation, using multiscale molecular modeling*. Doctor of Philosophy in Chemical Engineering, University of Alberta, Edmonton, Alberta, 2023. URL https://era.library.ualberta.ca/items/f58a8108-bb63-4f5e-9fe3-245f8bf2a7b2.

[45] Frank Jensen. *Introduction to Computational Chemistry*. John Wiley & Sons, Chich-

ester, England ; Hoboken, NJ, 2nd ed. edition, 2007. ISBN 9780470058046. OCLC: 85851059.

[46] E. Schrödinger. An Undulatory Theory of the Mechanics of Atoms and Molecules. *Physical Review*, 28(6):1049–1070, December 1926. ISSN 0031-899X. doi: 10.1103/PhysRev.28.1049. URL https://link.aps.org/doi/10.1103/PhysRev.28.1049.

[47] P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Physical Review*, 136(3B): B864–B871, November 1964. ISSN 0031-899X. doi: 10.1103/PhysRev.136.B864. URL https://link.aps.org/doi/10.1103/PhysRev.136.B864.

[48] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140(4A):A1133–A1138, November 1965. ISSN 0031-899X. doi: 10.1103/PhysRev.140.A1133. URL https://link.aps.org/doi/10.1103/PhysRev.140.A1133.

[49] Arul Mozhi Devan Padmanathan. *The Influence of High Temperature Pyrolysis Melt and Lignin on Cellulose Pyrolysis Chemistry: A First-principles based Investigation*. Doctor of Philosophy in Chemical Engineering, University of Alberta, Edmonton, 2023. URL https://era.library.ualberta.ca/items/8c217839-9584-49a0-857a-042fc0e57308.

[50] R. Car and M. Parrinello. Unified Approach for Molecular Dynamics and Density-Functional Theory. *Physical Review Letters*, 55(22):2471–2474, November 1985. ISSN 0031-9007. doi: 10.1103/PhysRevLett.55.2471. URL https://link.aps.org/doi/10.1103/PhysRevLett.55.2471.

[51] Alessandro Laio and Francesco L Gervasio. Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. *Reports on Progress in Physics*, 71(12):126601, December 2008. ISSN 0034-4885, 1361-

6633. doi: 10.1088/0034-4885/71/12/126601. URL https://iopscience.iop.org/article/10.1088/0034-4885/71/12/126601.

[52] Marcella Iannuzzi, Alessandro Laio, and Michele Parrinello. Efficient Exploration of Reactive Potential Energy Surfaces Using Car-Parrinello Molecular Dynamics. *Physical Review Letters*, 90(23):238302, June 2003. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.90.238302. URL https://link.aps.org/doi/10.1103/PhysRevLett.90.238302.

[53] Giovanni Bussi, Alessandro Laio, and Michele Parrinello. Equilibrium Free Energies from Nonequilibrium Metadynamics. *Physical Review Letters*, 96(9):090601, March 2006. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.96.090601. URL https://link.aps.org/doi/10.1103/PhysRevLett.96.090601.

[54] José Carlos Velasco Calderón and Samir H. Mushrif. Energetics of acid catalyzed biomass reactions: how and why does the solvent model matter? *Reaction Chemistry & Engineering*, page 10.1039.D3RE00340J, 2024. ISSN 2058-9883. doi: 10.1039/D3RE00340J. URL http://xlink.rsc.org/?DOI=D3RE00340J.

[55] Alianna J. Maren, Craig T. Harston, and Robert M. Pap, editors. *Handbook of Neural Computing Applications*. Academic Press, Inc., San Diego, California, 1990. ISBN 9780124712607 9780125460903. URL https://www.sciencedirect.com/book/9780125460903/handbook-of-neural-computing-applications.

[56] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. ISSN 1045-9227, 1941-0093. doi: 10.1109/72.279181. URL https://ieeexplore.ieee.org/document/279181/.

[57] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computa-*

tion, 9(8):1735–1780, November 1997. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL https://direct.mit.edu/neco/article/9/8/1735-1780/6109.

[58] F.A. Gers. Learning to Forget: Continual Prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, volume 1999, pages 850–855, Edinburgh, UK, 1999. IEE. ISBN 9780852967218. doi: 10.1049/cp:19991218. URL https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218.

[59] Ponkrshnan Thiagarajan and Susanta Ghosh. A Jensen-Shannon Divergence Based Loss Function for Bayesian Neural Networks. *arXiv preprint arXiv:2209.11366*, 2022. doi: 10.48550/ARXIV.2209.11366. URL https://arxiv.org/abs/2209.11366.

[60] Shujian Yu and José C. Príncipe. Understanding autoencoders with information theoretic concepts. *Neural Networks*, 117:104–123, September 2019. ISSN 08936080. doi: 10.1016/j.neunet.2019.05.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0893608019301352.

[61] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv*, 1609.08144, 2016. doi: 10.48550/ARXIV.1609.08144. URL https://arxiv.org/abs/1609.08144.

[62] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, June 2014. ISSN 1532-4435.

[63] Meiqi Wang, Siyuan Lu, Danyang Zhu, Jun Lin, and Zhongfeng Wang. A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 223–226, Chengdu, October 2018. IEEE. ISBN 9781538682401. doi: 10.1109/APCCAS.2018.8605654. URL https://ieeexplore.ieee.org/document/8605654/.

[64] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying RELU and Initialization: Theory and Numerical Examples. *Communications in Computational Physics*, 28(5):1671–1706, June 2020. ISSN 1815-2406, 1991-7120. doi: 10.4208/cicp.OA-2020-0165. URL http://global-sci.org/intro/article_detail/cicp/18393.html.

[65] Yann LeCun and Yoshua Bengio. Convolutional Networks for Images, Speech, and Time-Series. In *The handbook of brain theory and neural networks*. MIT Press, Cambridge, Massachusetts, 1995.

[66] Cai Guo, Xinan Chen, Yanhua Chen, and Chuying Yu. Multi-Stage Attentive Network for Motion Deblurring via Binary Cross-Entropy Loss. *Entropy*, 24(10):1414, October 2022. ISSN 1099-4300. doi: 10.3390/e24101414. URL https://www.mdpi.com/1099-4300/24/10/1414.

[67] Sidharth Prasad Mishra, Uttam Sarkar, Subhash Taraphder, Sanjoy Datta, Devi Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. Multivariate Statistical Data Analysis-Principal Component Analysis (PCA). *International Journal of Livestock Research*, 7(5), 2017. ISSN 2277-1964. doi: 10.5455/ijlr.20170415115235. URL http://www.ejmanager.com/fulltextpdf.php?mno=261590.

[68] Wenfu Ku, Robert H. Storer, and Christos Georgakis. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 30(1):179–196, November 1995. ISSN 01697439. doi: 10.

1016/0169-7439(95)00076-3. URL https://linkinghub.elsevier.com/retrieve/pii/0169743995000763.

[69] Jörg Behler and Michele Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Physical Review Letters*, 98(14):146401, April 2007. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.98.146401. URL https://link.aps.org/doi/10.1103/PhysRevLett.98.146401.

[70] Xue Ying. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168:022022, February 2019. ISSN 1742-6588, 1742-6596. doi: 10.1088/1742-6596/1168/2/022022. URL https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022.

[71] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, 13 (2):e1484, March 2023. ISSN 1942-4787, 1942-4795. doi: 10.1002/widm.1484. URL https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1484.

[72] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/e995f98d56967d946471af29d7bf99f1-Paper.pdf.

[73] Hoon Kang, Seunghyeok Yang, Jianying Huang, and Jeill Oh. Time Series Prediction of Wastewater Flow Rate by Bidirectional LSTM Deep Learning. *International Journal of Control, Automation and Systems*, 18(12):3023–3030, December 2020.

ISSN 2005-4092. doi: 10.1007/s12555-019-0984-6. URL https://doi.org/10.1007/s12555-019-0984-6.

[74] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, July 2015. ISSN 1749-4699. doi: 10.1088/1749-4699/8/1/014008. URL https://iopscience.iop.org/article/10.1088/1749-4699/8/1/014008.

[75] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. *arXiv*, 1506.07503, 2015. doi: 10.48550/ARXIV.1506.07503. URL https://arxiv.org/abs/1506.07503.

[76] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv*, 1704.02971, 2017. doi: 10.48550/ARXIV.1704.02971. URL https://arxiv.org/abs/1704.02971.

Appendix A

A.1 Introduction

When the dataset used in this work is converted into probability distribution functions (PDF), the result is four dimensions: a three dimensional PDF and a temporal component. Since the nature of the data is four dimensional, it is difficult to visualize clearly. This appendix attempts to give readers a better sense of the data by showing the three dimensional PDFs as two dimensional projections on three different planes for a certain timestep. This appendix gives the projections for each species at three different states of the ground truth data: the reactant state, transition state and product state.

A.2 Ground Truth Reactants Projections

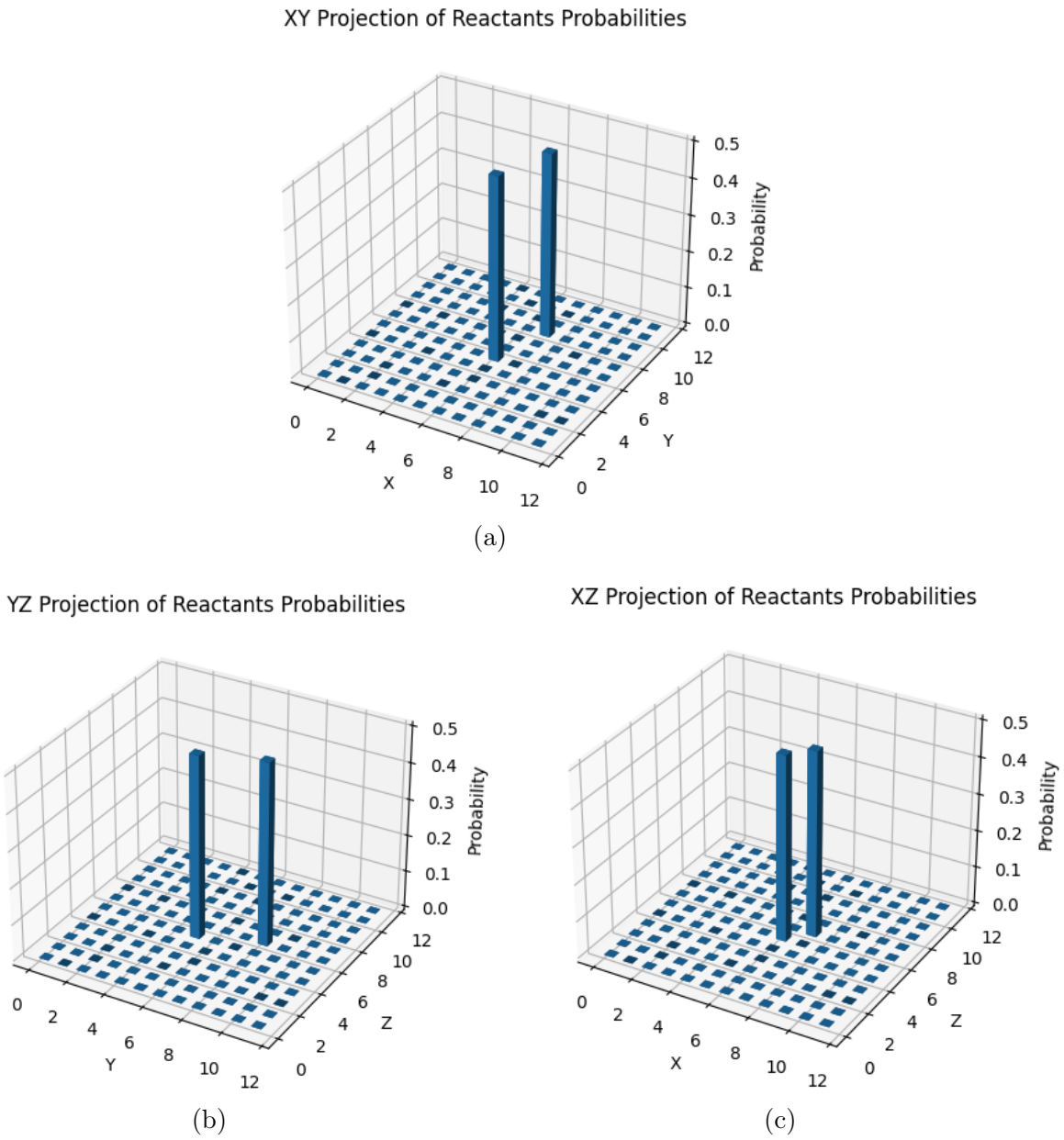


Figure A.1: Projections of the reactants species for the ground truth PDF when the system is at its reactants state.

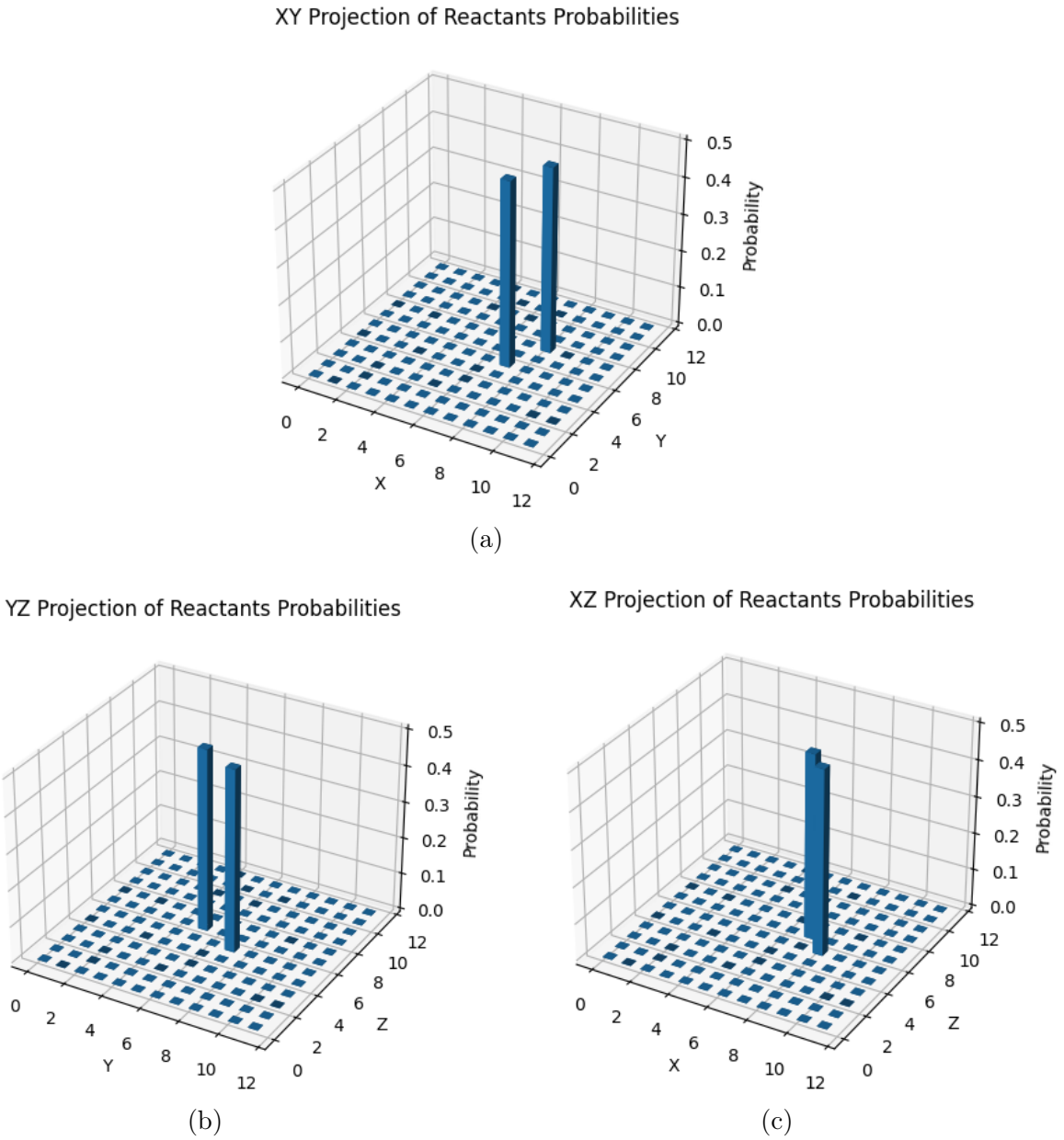


Figure A.2: Projections of the reactants species for the ground truth PDF when the system is at its transition state.

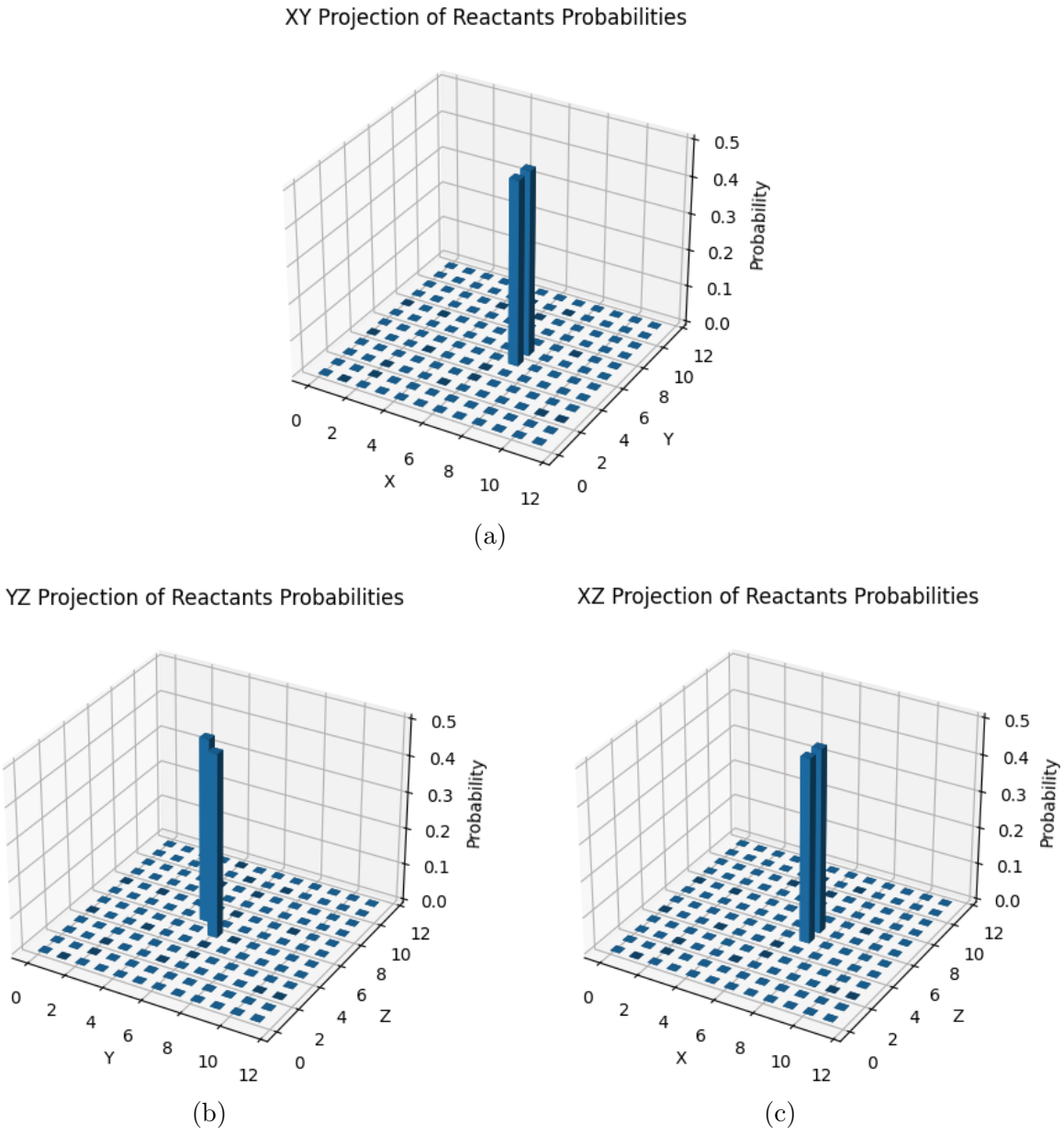
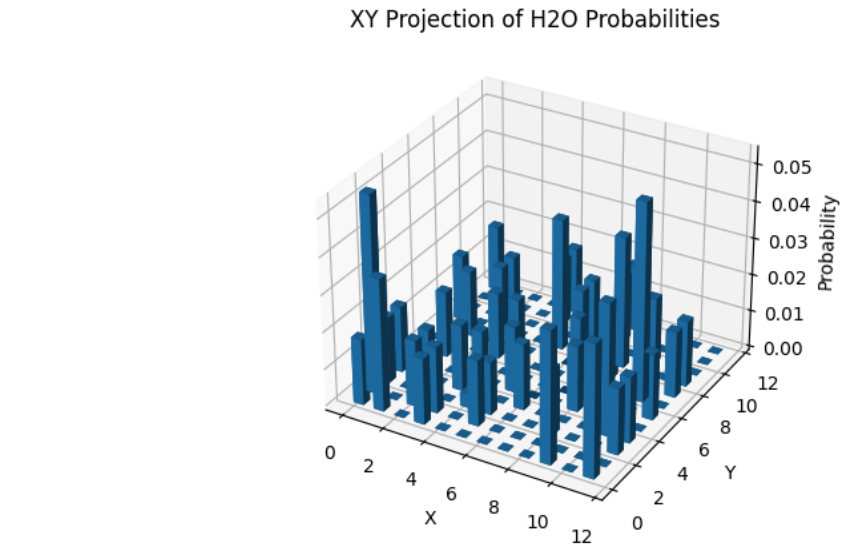
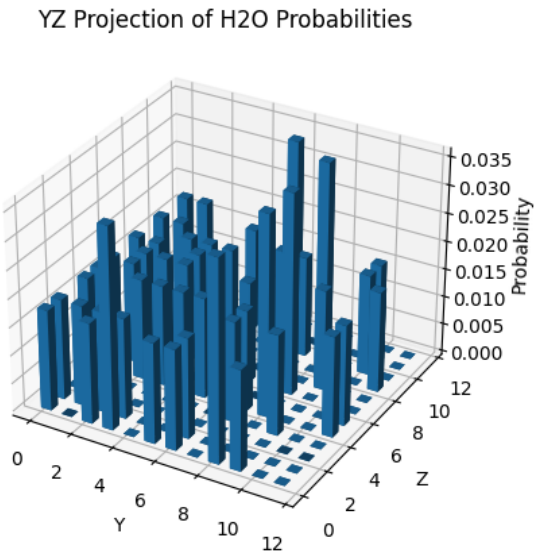


Figure A.3: Projections of the reactants species for the ground truth PDF when the system is at its product state.

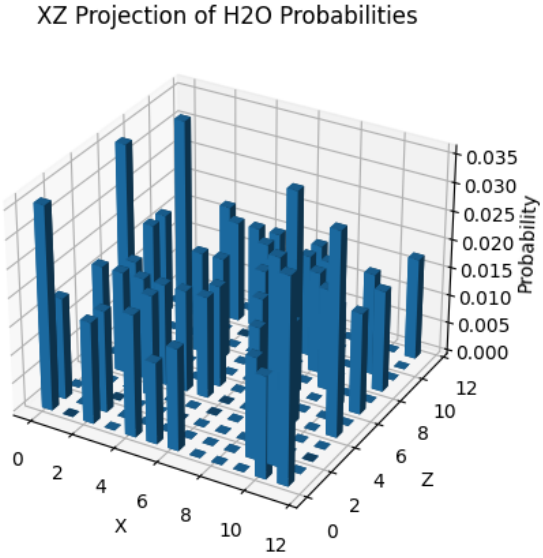
A.3 Ground Truth Water Projections



(a)



(b)



(c)

Figure A.4: Projections of water molecules for the ground truth PDF when the system is at its reactants state.

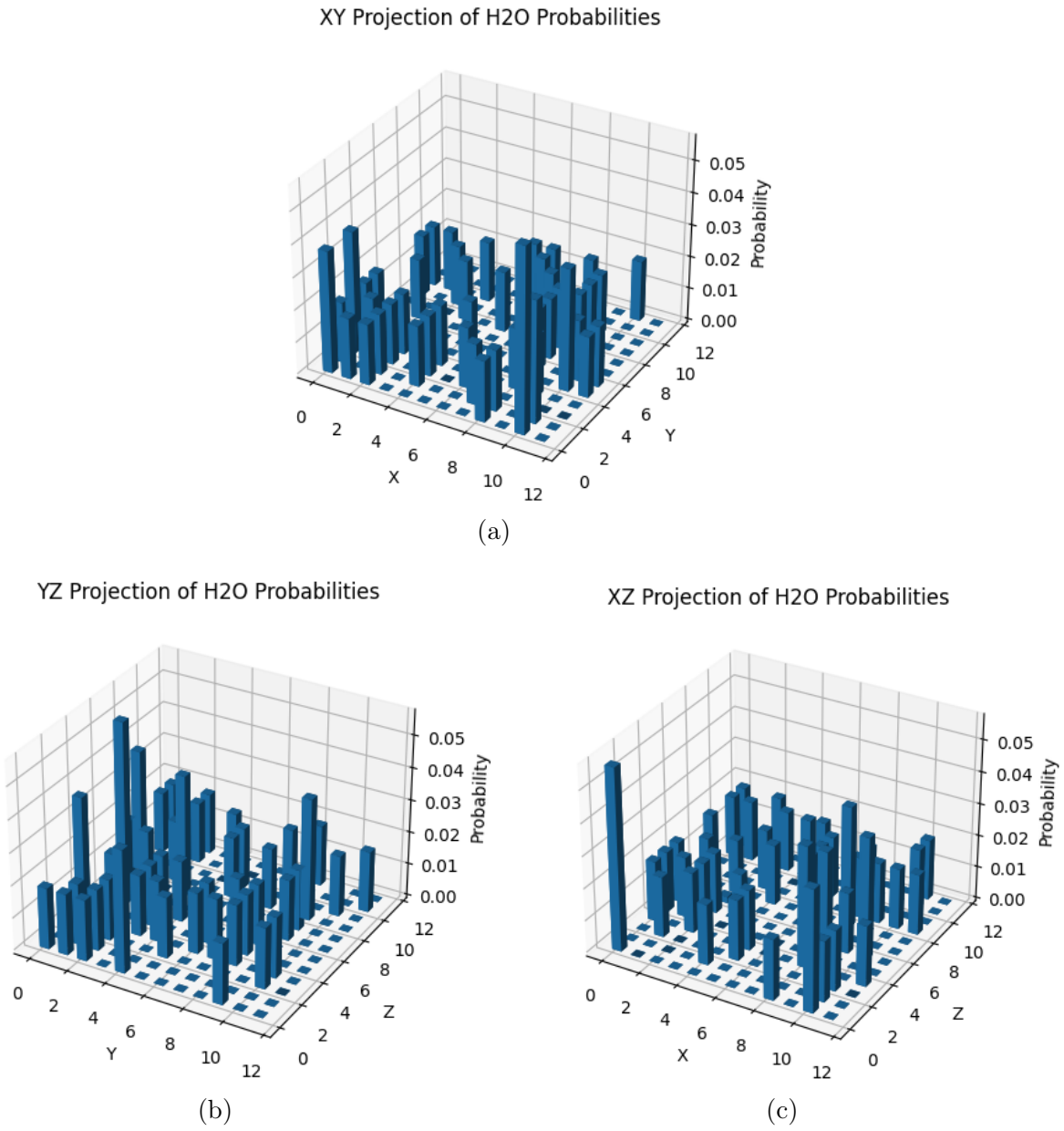


Figure A.5: Projections of water molecules for the ground truth PDF when the system is at its transition state.

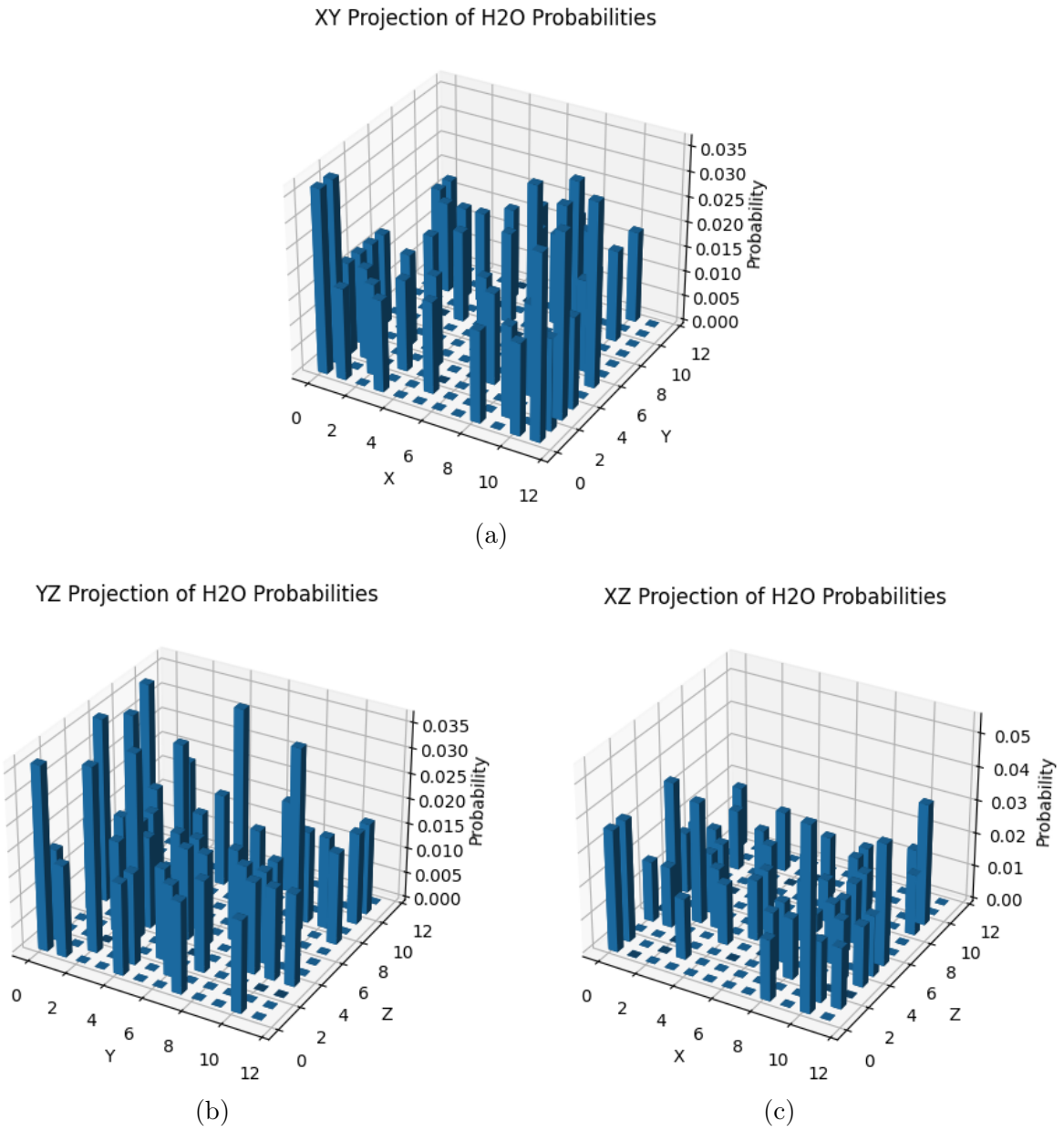
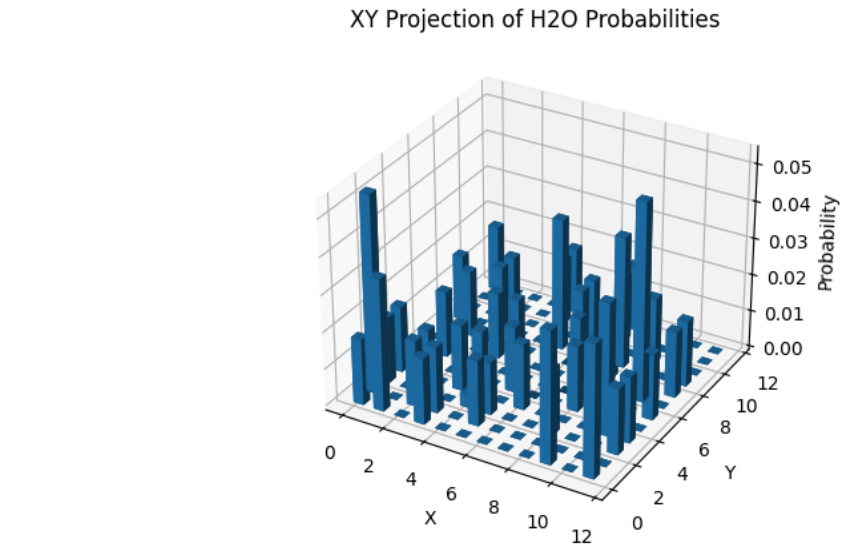
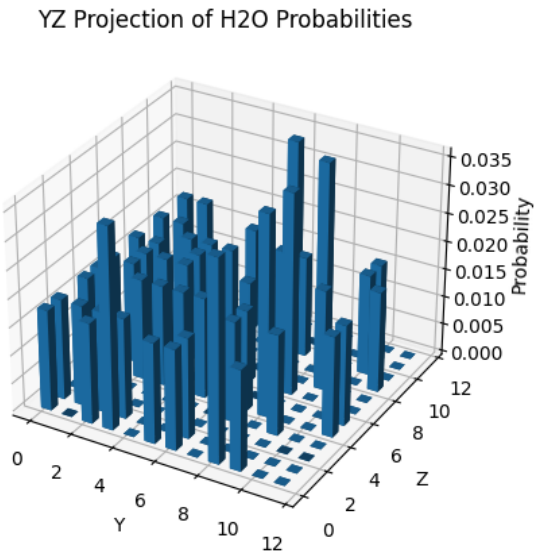


Figure A.6: Projections of water molecules for the ground truth PDF when the system is at its product state.

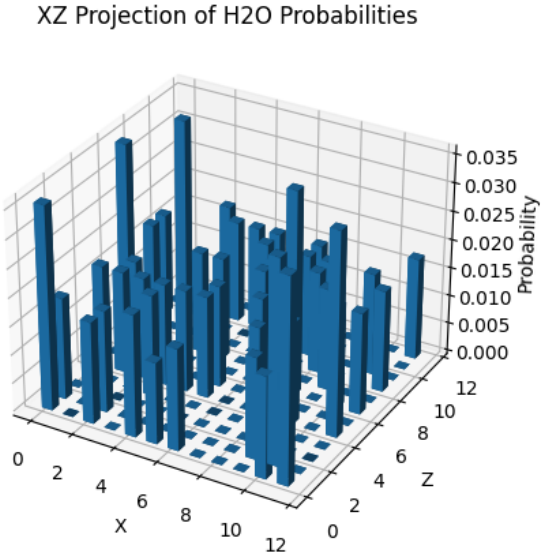
A.4 Ground Truth DMSO Projections



(a)

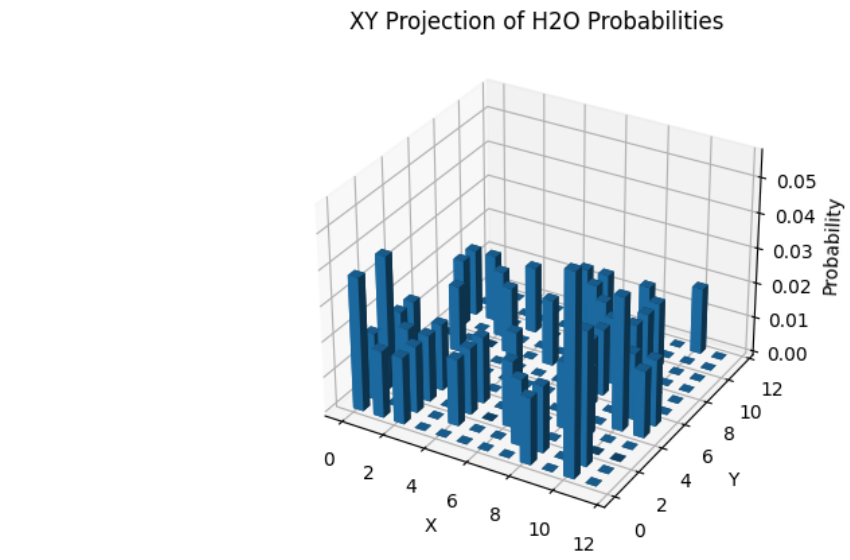


(b)

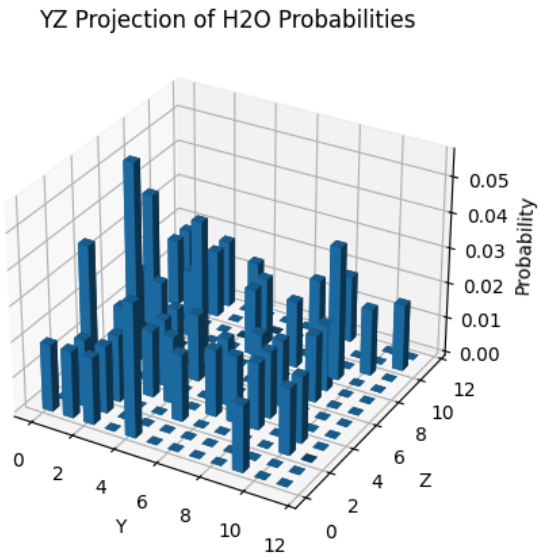


(c)

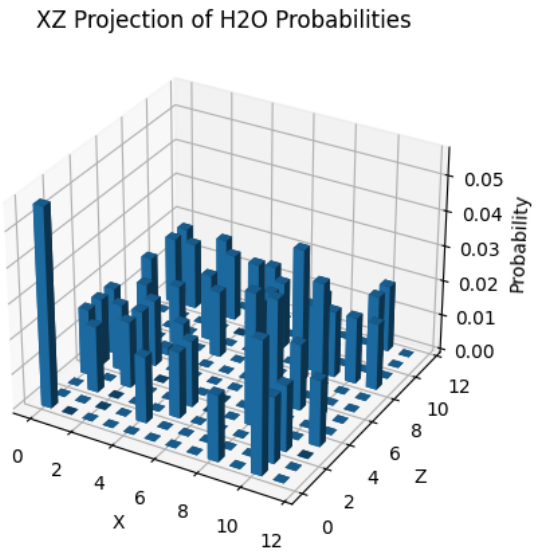
Figure A.7: Projections of DMSO molecules for the ground truth PDF when the system is at its reactants state.



(a)



(b)



(c)

Figure A.8: Projections of DMSO molecules for the ground truth PDF when the system is at its transition state.

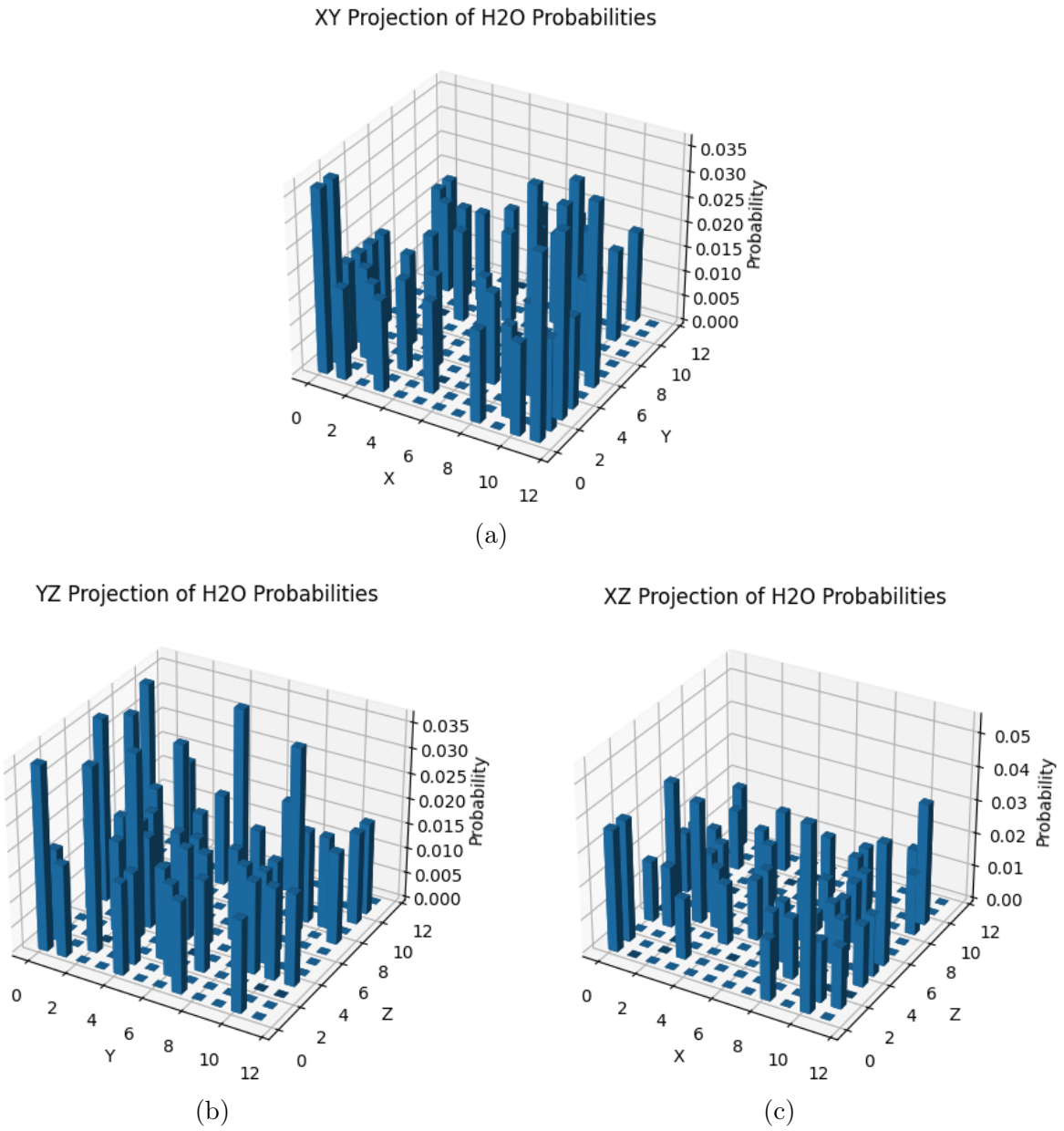


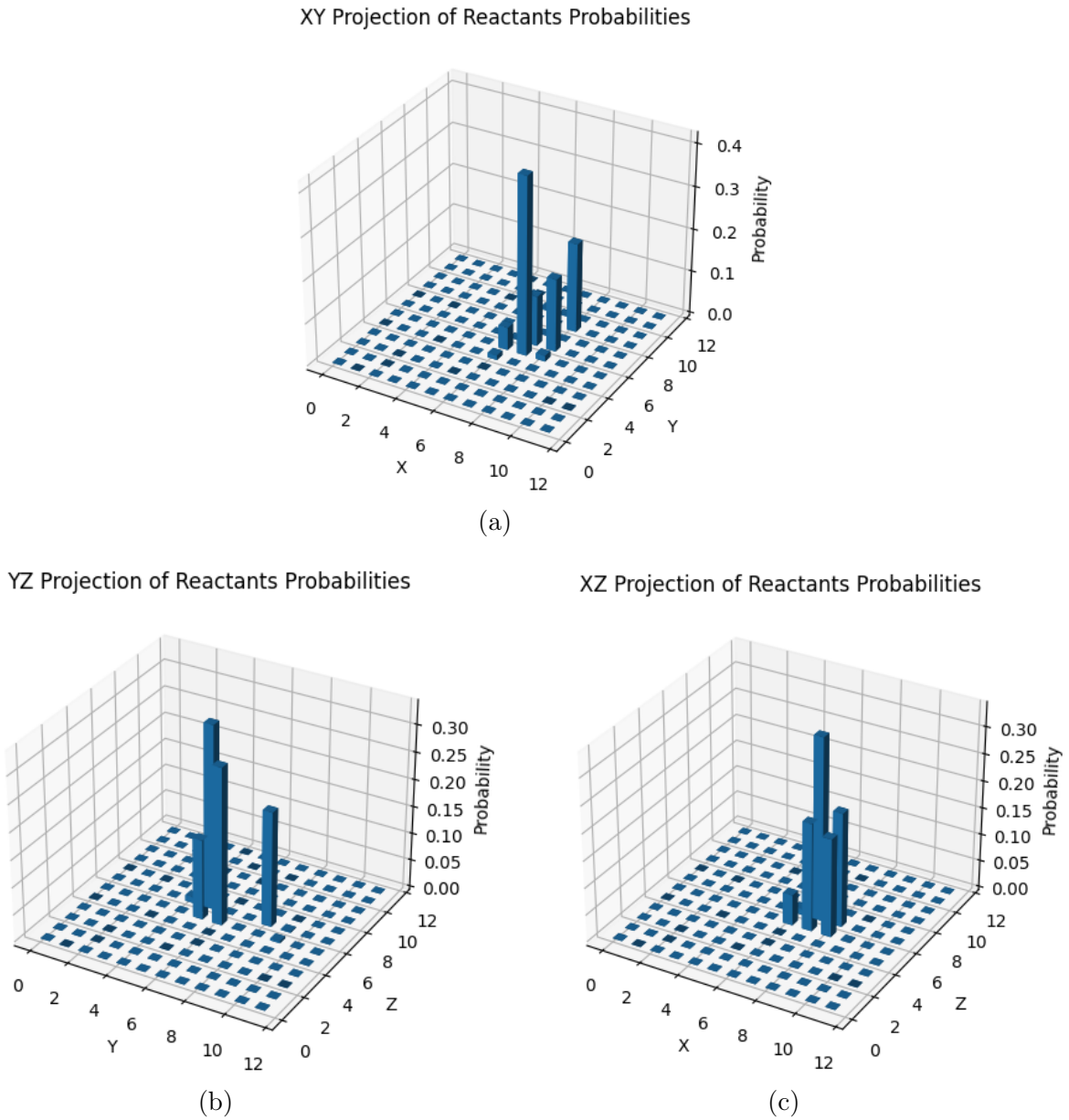
Figure A.9: Projections of DMSO molecules for the ground truth PDF when the system is at its product state.

Appendix B

B.1 Introduction

This appendix gives the two dimensional projections of the PDFs for both the first and 5000th predicted time step by the PCA model discussed in Section 5.3.1.

B.2 Predicted Reactants Projections

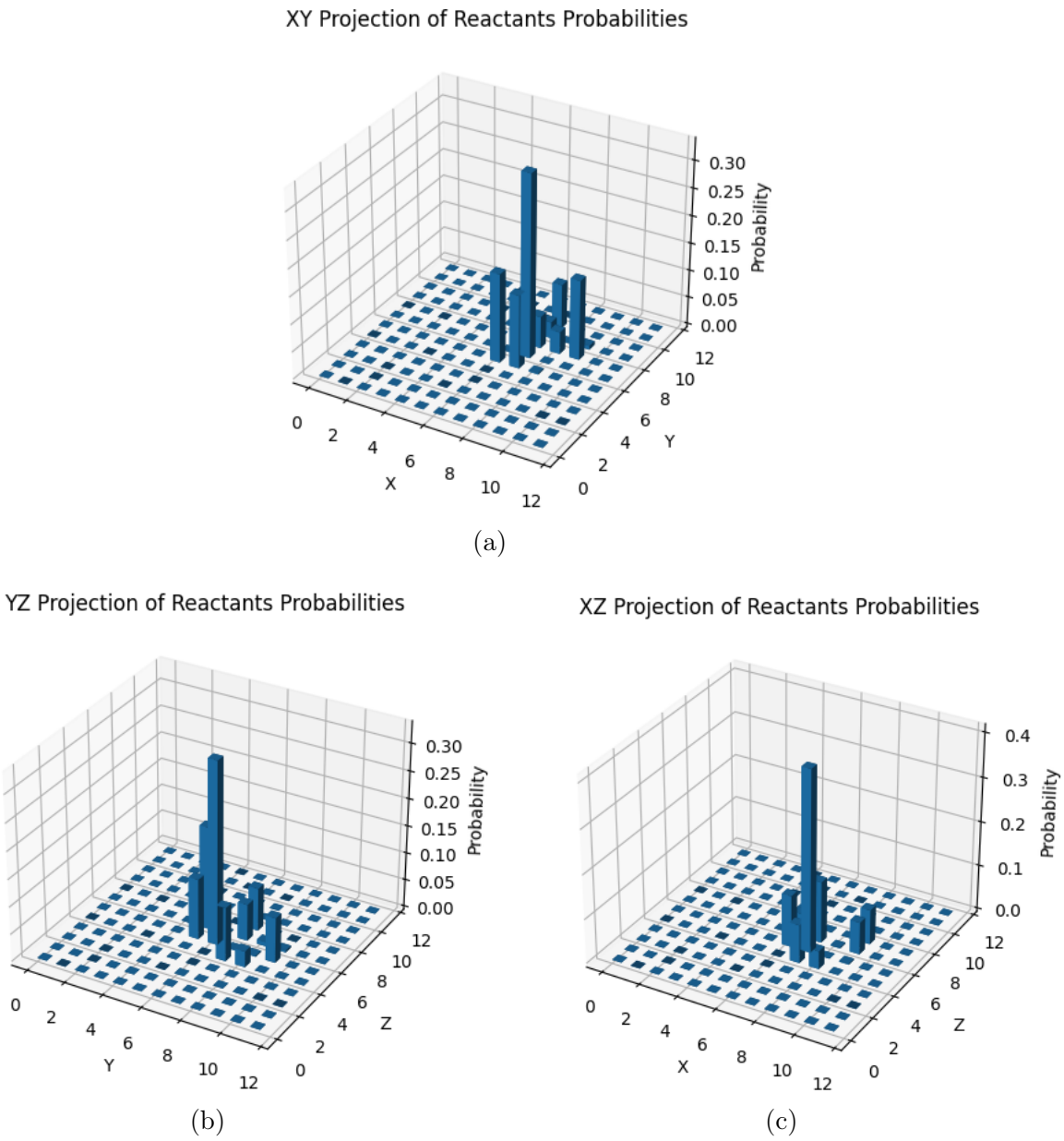


(a)

(b)

(c)

Figure B.1: Projections of the reactants species for the first predicted time step by the PCA model.



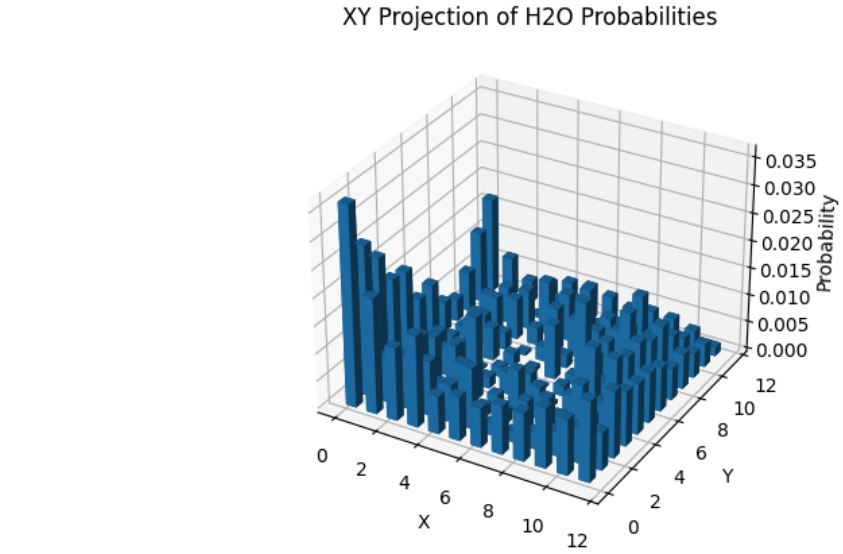
(a)

(b)

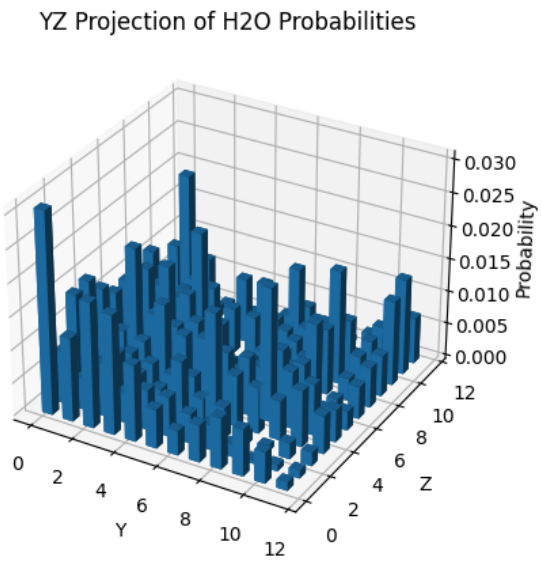
(c)

Figure B.2: Projections of the reactants species for the 5000th predicted time step by the PCA model.

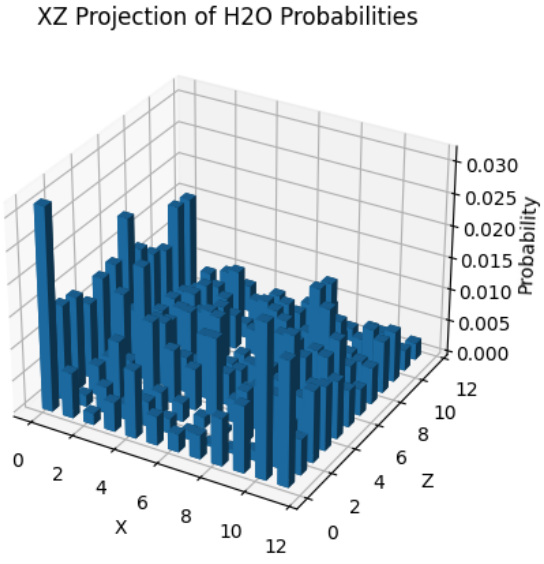
B.3 Predicted Water Projections



(a)



(b)



(c)

Figure B.3: Projections of water molecules for the first predicted time step by the PCA model.

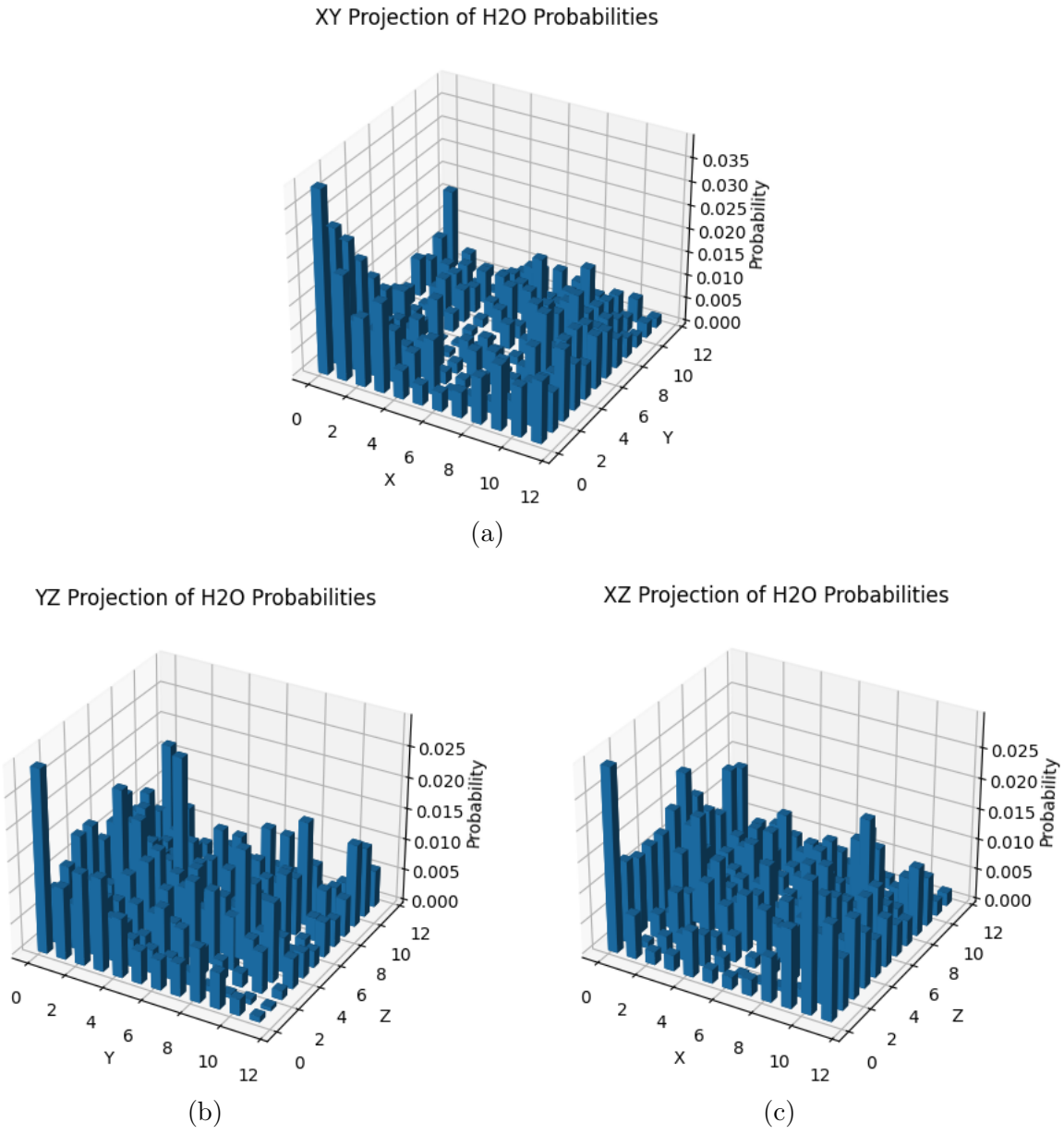
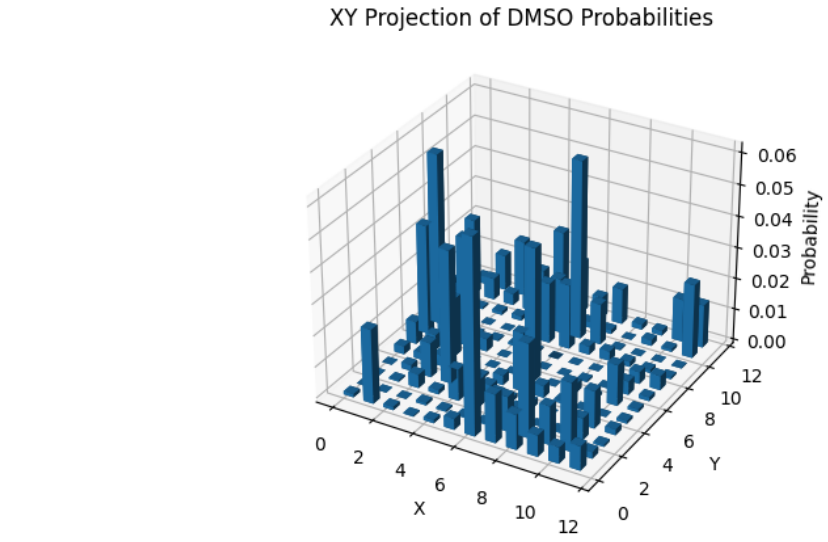
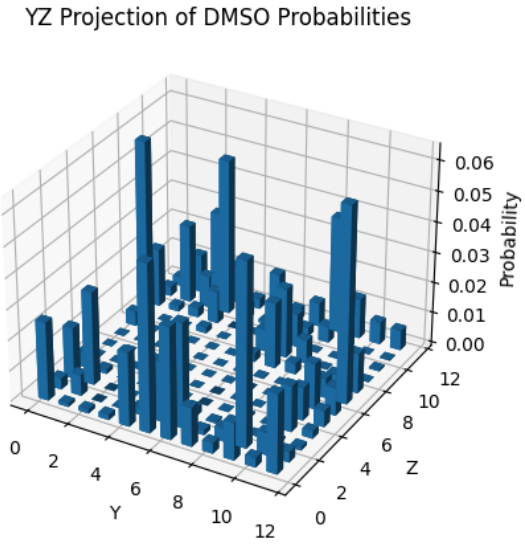


Figure B.4: Projections of water molecules for the 5000th predicted time step by the PCA model.

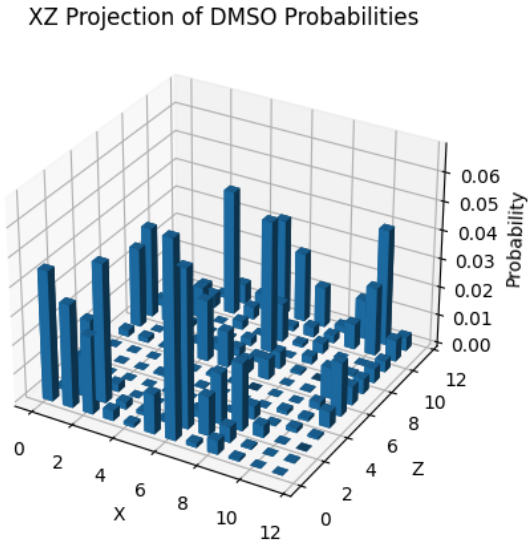
B.4 Predicted DMSO Projections



(a)



(b)



(c)

Figure B.5: Projections of DMSO molecules for the first predicted time step by the PCA model.

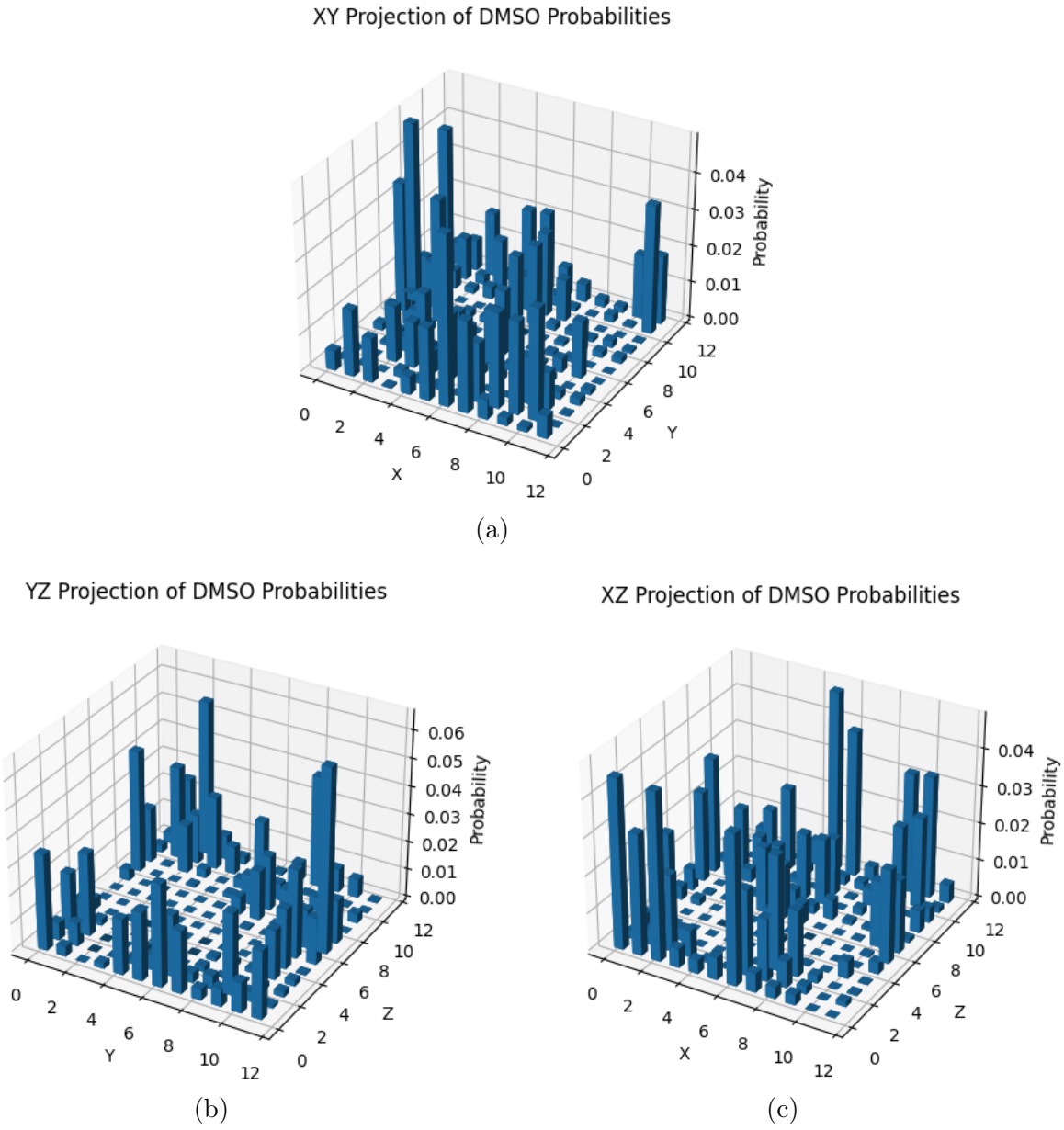


Figure B.6: Projections of DMSO molecules for the 5000th predicted time step by the PCA model.