Leveraging attention for prediction of reaction energetics

by

Mohammad Aarooj Yashin Ali

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

Master of Science

in

Chemical Engineering

Department of Chemical and Materials Engineering
University of Alberta

Abstract

Ab initio molecular dynamics (AIMD) simulations offer unmatched mechanistic insight into intricate chemical processes, especially in solvated environments. Though it comes with high computational cost. In order to significantly cut down on the time and cost needed to investigate reaction energetics and mechanisms, this thesis presents the development and validation of a hybrid deep learning architecture intended to act as a computationally effective proxy for Car-Parrinello molecular dynamics (CPMD) metadynamics simulations.

The suggested architecture approaches this problem by creating context-aware feature representations for input sequences followed by study of local temporal dynamics in these sequences by combining a Transformer encoder with Long Short-Term Memory (LSTM) layers in a synergistic manner. In order to create a context-aware feature representation, the Transformer's self-attention mechanism first records the global, long-range temporal correlations within a series of atomic coordinates. To guarantee physical coherence, LSTM layers subsequently process this representation in a sequential manner, modeling the local, time-ordered evolution of the system's dynamics. Using a historical data from a CPMD simulation of a reaction system, this hybrid model is trained to autoregressively predict future atomic configurations of the same system.

Two different chemical systems were used to test the model's effectiveness. The first, a gas-phase intramolecular hydride transfer reaction, showed how well the model could accurately identify the reactant, product, and transition states while reproducing the qualitative topology of the free energy surface (FES). The global reaction energy (ΔG_{rxn}) showed a considerable quantitative error, indicating a trade-off between kinetic and thermodynamic prediction accuracy, whereas the activation energy ($\Delta G^{\ddagger}$) was predicted with decent accuracy. The scalability and robustness of the model were tested in the second case study, which involved protonating 5-hydroxymethylfurfural (HMF) in a DMSO and water solvent. Since this reaction system is complex compared to the gas phase system, the model is scaled-up in terms of the hyperparameters used. The main mechanistic steps of the reaction were effectively captured.

The results demonstrate that the hybrid Transformer-LSTM model can serve as an effective tool for generating hypotheses, allowing for the quick screening of reaction conditions and the clarification of qualitative mechanisms at a fraction of the price of conventional AIMD. Although the model cannot replace simulations with small amount of data, but it still allows to reduce computational cost in computing the configurations of transition phases involved over the course of complete simulation. Future research will concentrate on expanding the architecture to speed up classical MD simulations, improve the model to recognize patterns in sequences with lesser amount of data and putting transfer learning into practice to produce more generalizable models for a variety of chemical environments.

Preface

This thesis is an original work by Aarooj. No part of this thesis has been previously published.

“If you think you understand quantum mechanics, you don’t understand quantum mechanics.”
~ Richard Feynman

Acknowledgements

I would like to express my gratitude to my academic supervisors, Dr. Vinay Prasad and Dr. Samir H. Mushrif, for providing me with this remarkable chance to enhance my knowledge and expand my capabilities. Your patience, encouragement, and skill were vital to my success. Your fervor for your students and your profession has inspired me and will persist in doing so long after my departure from the University of Alberta.

During my two years at the University of Alberta, my mom; Nazlin Haque Ali, my dad; Naushad Ali and my brother; Ehaan have consistently supported my studies and personal development. My pursuit of a Master's degree has been supported by an extraordinary array of individuals both within and beyond the university. I extend my gratitude to Kevin, Karthik, Anjana, Eduardo, Kuldeep, Sushmita, Devavrat, Shahdab, Mahesh, Aqib, Sergio, and Jose for assisting me with any uncertainties or obstacles encountered during the production of my thesis, and for making my time in Edmonton interesting and pleasurable. Each of these exceptional individuals possesses a distinct array of strengths, from which I endeavored to acquire as much as knowledge possible.

I wish to express my gratitude to the teachers of the Chemical and Materials Engineering department who have guided me during my Master's degrees, and to the Digital Research Alliance of Canada for the computing tools they provide for my research.

Sincerely,
Aarooj Yashin
Edmonton 2025

Table of Contents

List of Tables

List of Figures

List of Symbols

- k_{S,j_i}: The apparent rate constant for reaction i in solvent S with a mass fraction of j.
- k_{H_2O}: The apparent rate constant for reaction i in pure water.
- $\Delta G^{\ddagger}$: Activation Free Energy; the energy barrier for a transformation, calculated as $F_{TS} - F_{Reactant}$.
- ΔG_{rxn}: Reaction Free Energy; the overall thermodynamic favorability of a process, calculated as $F_{Product} - F_{Reactant}$.
- F_{TS}: Free energy of the transition state (saddle point).
- $F_{Reactant}$: Free energy of the initial (reactant) state.
- $F_{Product}$: Free energy of the final (product) state.
- H: The Hamiltonian operator.
- $\hbar$: The Planck constant.
- Ψ_e: The wavefunction for electrons.
- R_e: Electronic coordinates.
- R_n: Nuclear coordinates.
- ρ: Electron density.
- χ_{KS}: Kohn-Sham orbitals.
- $v(R)$: An external potential.
- $v_{KS}(R)$: The potential for Kohn-Sham orbitals.
- E_{MM}: Total potential energy in classical molecular dynamics.
- $E_{stretch}$: Energy from bond stretching.
- E_{bend}: Energy from angle bending.
- $E_{torsion}$: Energy from bond twisting.
- E_{vdW}: Energy from van der Waals interactions.

- $E_{electrostatic}$: Energy from electrostatic interactions.
- $E_{coupled}$: Energy from coupled interactions of forces.
- $K_{stretch}, K_{bend}, K_{torsion}$: Force constants for stretching, bending, and torsion.
- ω: Angle of rotation in torsion energy calculations.
- n: Periodicity in torsion energy calculations.
- E_{minL-J}: Depth of the potential minimum for van der Waals interactions.
- R_0: Distance at zero potential for van der Waals interactions.
- Q: Atomic charge.
- ϵ_{dielec}: The dielectric constant.
- $R(R_1, R_2, \ldots, R_N)$: A system of N atoms.
- $R_0, dR_0/dt$: Initial positions and velocities of atoms.
- $E(R), dE/dR$: Potential energy and forces of the system.
- μ: Fictional mass for wavefunctions in Car-Parrinello Molecular Dynamics (CPMD).
- χ_j: The orbital of the jth electron.
- s: The vector of Collective Variables (CVs).
- m_{CV}: Fictitious mass assigned to the CVs.
- K_{CV}: Force constant for the potential energy acting on the CVs.
- $\nu_{CV}(t, s)$: Energy potentials added during metadynamics.
- CV1, CV2: The first and second collective variables used in a simulation.
- CN: Coordination Number.
- p, q: Exponents used in the CN calculation.
- d_0: Reference distance used in the CN calculation.
- H_α: The proton participating in the reaction.
- O_{water}: Bulk oxygen belonging to a water molecule.
- C_4: The protonation site on a molecule.
- H_{water}: Bulk hydrogen belonging to a water molecule.

- p, d, q: Parameters for an ARIMA model (Autoregressive, Integrated, Moving Average).
- ϕ_i, θ_j: Autoregressive and moving average parameters in the ARIMA model.
- ϵ_t: White noise error term at time t.
- x_t: Input vector at time step t.
- h_t: Hidden state vector at time step t.
- y_t: Output vector at time step t.
- W_{hh}, W_{xh}, W_{hy}: Weight matrices for hidden-to-hidden, input-to-hidden, and hidden-to-output connections in an RNN.
- b_h, b_y: Bias vectors for the hidden and output layers in an RNN.
- C_t: The cell state vector in an LSTM at time t.
- f_t, i_t, o_t: Forget, input, and output gates in an LSTM.
- W_f, W_i, W_C, W_o: Learnable weight matrices for the gates and cell state in an LSTM.
- b_f, b_i, b_C, b_o: Learnable bias vectors for the gates and cell state in an LSTM.
- σ: The sigmoid activation function.
- tanh: The hyperbolic tangent activation function.
- $\odot$: The element-wise (Hadamard) product operation.
- Q, K, V: Query, Key, and Value vectors in the Transformer attention mechanism.
- d_k: The dimension of the key vectors in the Transformer.
- d_{model}: The dimension of the embeddings in the Transformer.
- $F(s)$: The free energy surface as a function of the collective variables s.
- $V_G(s)$: The total accumulated bias potential.
- N_{hills}: The total number of Gaussian hills deposited during the simulation.
- W_k: The height of the k-th Gaussian hill.
- $\sigma_{i,k}$: The width of the k-th Gaussian hill in the i-th CV dimension.
- $s_{i,k}$: The center of the k-th Gaussian hill in the i-th CV dimension.

Chapter 1 - Introduction

At present, numerous chemical and pharmacological processes transpire in the presence of a solvent; nonetheless, the impact of the solvent environment remains inadequately comprehended at a fundamental level [1]. Besides stabilizing the transition state (TS) and modifying reaction kinetics, a solvent and its reorganization can also (i) influence reaction mechanisms and pathways, (ii) affect the relative stability of the reactants, transition state, and products, (iii) modify reaction thermodynamics, (iv) enhance diffusion in porous catalysts, and (v) increase selectivity [2]. These effects must be taken into account while assessing candidate materials and catalysts for optimization. Although empirical tests can identify the optimal solvent(s) and their respective concentrations for a reaction via trial and error, this approach is both time-consuming and costly, and it fails to facilitate a mechanistic comprehension of why a specific solvent is optimal for a particular reaction [3]. Employing molecular dynamics methods Is a compelling choice for such investigations.

Molecular dynamics is a computational method that simulates the behavior of chemical systems at the atomic scale. This degree of precision enables researchers to attain a more profound comprehension of chemical processes via molecular dynamics. The behavior of a system is represented using conventional molecular dynamics (MD) or ab initio molecular dynamics (AIMD). Classical molecular dynamics employs Newton's equations of motion to determine the temporal development of atomic locations. This is frequently computationally efficient, enabling the modeling of larger systems and the exploration of extended timescales. Nonetheless, it inadequately represents chemically intricate systems, such as reaction systems; hence, AIMD is essential for modeling these systems [4]. AIMD methods use quantum mechanics to simulate molecular dynamics and are frequently utilized to investigate reactions compared to classical approaches. AIMD methods for solvent-based systems can model solvent molecules either implicitly or explicitly. Implicit solvation models suggest the existence of solvent molecules by estimating the average force applied to the explicitly modeled molecules, whereas explicit methods represent each molecule within the system [5]. Although implicit models can effectively approximate reality with minimal resources, they also have certain disadvantages [6]. Implicit models have been demonstrated to inadequately represent hydrogen bonding interactions and entropic influences on free energy [7,8]. Explicit solvation modeling may properly simulate solvation systems; however, it is computationally demanding and hence constrained in its timeline.

A significant factor contributing to the computational expense of AIMD is the necessity to recalculate the electrical structure of the system after each time step. Car-Parinello molecular dynamics (CPMD) resolves this issue by tracking the evolution of the electronic structure using a classical description [4]. The metadynamics method can be employed with CPMD to expedite reaction events, enabling their observation within a reduced time frame. In metadynamics, minor energy potentials are introduced to the system to populate the energy wells inside a user-specified coordinate space. The supplementary energy potentials are subsequently deducted to elucidate the potential energy surface of the system inside the coordinate space [9]. The potential energy surface of a chemical system illustrates the activation energy, facilitating the examination of solvent effects on reactions. Despite employing the CPMD formulation and the metadynamics technique, a comprehensive simulation of a reaction system with 232 atoms requires approximately 100,000 CPU-hours. Conducting a study to identify the optimal solvents and their concentrations for a certain set of processes could require several years.

To mitigate the computational expense of AIMD methods, researchers have begun incorporating machine learning (ML) into established molecular dynamics (MD) techniques. It has been demonstrated that machine learning techniques can enhance classical molecular dynamics simulations by more efficiently [10] sampling protein conformations and predicting reaction rates [11]. Nonetheless, the influence of machine learning extends beyond traditional molecular dynamics. Machine learning methods have been employed to expedite AIMD simulations by forecasting the energies and forces of atomic configurations [12,13], develop more precise density functionals [14], and simulate atomic interactions [15]. In the realm of solvents, machine learning has been employed to enhance the precision of implicit solvation simulations [16] and to simulate reactions inside explicit solvation frameworks [17,18]. In explicit solvation contexts, machine learning has mostly been utilized to anticipate the potential energy surface or reaction barriers, rather than forecasting the evolution of atomic configurations. This study endeavors to provide a proxy machine learning algorithm for CPMD metadynamics simulations of explicitly solvated biomass processes. This approach would reduce the time and computational expense of analyzing such systems by forecasting both the atomic configurations and the potential energy surface.

The process being investigated in the condensed phase is the protonation of 5-hydroxymethylfurfural (HMF) in water and dimethyl sulfoxide (DMSO). The reaction pathway of HMF into humins is noteworthy due to the minimal economic value of humins, whereas HMF ultimately yields lucrative fuel additives and polymers [19]. The presence of DMSO as a cosolvent protects HMF molecules, hence impeding their degradation into humins [20]. Machine learning-assisted ab initio molecular dynamics simulations could yield mechanistic insights at a significantly reduced computational cost and time, since a proxy machine learning model would serve as a shortcut to outcomes currently achievable using conventional ab initio molecular dynamics approaches. The enhanced velocity and diminished computing expense of AIMD simulations would render the screening for cosolvents that augment the selectivity of HMF a less arduous endeavor.

We investigate the intramolecular hydride transfer by taking Lewis acid catalyzed glucose to fructose isomerization as an example. The C2–C1 hydride transfer is the rate limiting step in this reaction.

For condensed phase the reaction considered is of **5-hydroxymethylfurfural (5-HMF)** under acidic and basic conditions. In the presence of **protons (H^+)**, 5-HMF undergoes protonation and rearrangement to form **levulinic acid** and **formic acid**. Under alkaline conditions (**OH^-**), 5-HMF converts into intermediate furan derivatives, which can be further protonated to yield **2,5-dihydroxyhex-2-enone (DHH)**. This compound readily undergoes **polymerization**, leading to the formation of **humins**, which are insoluble carbonaceous byproducts

Chapter 2 - Literature Review

This chapter discusses the recent research & developments on machine learning techniques to tackle the high computational cost involved with molecular dynamics methods.

In the past, the fields of computing and chemistry have come into contact. Basic calculations related to chemical reactions and molecular structures were initially performed on crude analog computers [21], but as both domains advanced, so did their relationship. Complex chemical systems can now be analyzed through extremely detailed simulations thanks to recent major advances in computing power. The methods used in computational and experimental chemistry have changed as a result of the combination of machine learning and artificial intelligence (AI).

One of machine learning's main advantages is its ability to save time and reduce costs for the tasks it is trained to perform. Chemists frequently endeavor to optimize reactions by varying a single experimental variable at a time while maintaining the rest constant. This method frequently fails to uncover ideal conditions and is often lengthy and resource-intensive [22]. Therefore, creating a machine learning model to optimize chemical interactions would be quite beneficial. Zhou et al. [23] achieved this objective with their Deep Reaction Optimizer (DRO) model, employing reinforcement learning for training. Reinforcement learning models execute actions based on inputs to optimize a specific "reward." The DRO model modifies reaction conditions to optimize yield in chemical reactions. The model was trained using a combination of Gaussian density functions. This approach is motivated by the premise that the response surface for most responses may be regarded as a continuous function, and that a combination of Gaussian density functions can effectively mimic any continuous function. When evaluated on a set of hitherto unencountered Gaussian mixture density functions in conjunction with many prominent black-box optimization techniques, DRO demonstrated superior optimization efficiency, requiring fewer iterations than its rivals. Initially, it underwent pre-training on simulated reaction data for four reactions to evaluate its efficacy in chemical reactions. For each reaction, the model needed to modify the flow rate, voltage, and pressure according to arbitrary initial circumstances to ascertain the maximum yield. DRO demonstrated superior optimization efficiency, requiring fewer steps than the covariance matrix adaptation-evolution strategy (CMA-ES) and the conventional one variable at a time method employed by chemists for optimization purposes [23]. The DRO model effectively determined the ideal reaction conditions by meticulously adjusting several parameters that directly affected the reaction yield. Nonetheless, it is crucial to acknowledge that the impact of these characteristics remains uniform throughout different reactions. Conversely, modifying the solvent environment in which a reaction transpires can produce disparate effects for distinct processes. Therefore, the DRO model lacks the ability to predict if an alternative solvent or co-solvent combination in a particular ratio would result in an increased yield for a specific reaction.

Classical molecular dynamics simulations can elucidate the interactions of reactants, water, and cosolvents within a condensed-phase reaction system [24]. Walker et al. [11] posited that the reaction rates of acid-catalyzed processes involving biomass-derived compounds could be computationally anticipated based on three parameters: i) the quantity of water molecules within the local solvation shell of the reactant, ii) the average duration of bonds between the reactant and adjacent water molecules, and iii) the proportion of the reactant's surface area accessible to nearby hydroxyl groups. The indicators were quantified and incorporated into a linear regression model to forecast the kinetic solvent parameter σ for a specific process. The kinetic solvent parameter *σ* is articulated in Equation 2.1, where $k_{S,j}^{i}$ denotes the apparent rate constant for reaction *i* in solvent *S* with a mass fraction of *j*, and $k_{H}{}^{i}{}_{2O}$ represents the apparent rate constant for reaction *i* in pure water.

$$\sigma_{S_j}^{i} = \log_{10}\left(\frac{k_{S_j}^{i}}{k_{H_2O}^{i}}\right) \tag{2.1}$$

A positive kinetic solvent parameter signifies that the solvent environment enhances the reaction rate, whereas a negative kinetic solvent parameter denotes the contrary. The model was trained using experimentally derived rate constants for seven biomass reactions in an aqueous environment with one of three cosolvents: 1,4 dioxane (DIO), γ-valerolactone (GVL), and tetrahydrofuran (THF). The regression model findings indicate that these three descriptors may predict the kinetic solvent parameter with acceptable accuracy, especially for DIO cosolvent systems [11]. This indicates that, in specific instances, classical molecular dynamics simulations can forecast solvent effects on reaction rates without the need for detailed modeling of the reaction mechanism. The regression model exhibited reduced accuracy for GVL and THF mixtures, suggesting inherent limits within the model. Theodore et al. determined that either the descriptors derived from classical molecular dynamics are insufficient for quantifying reaction rates in certain systems, or that more intricate descriptors are necessary to encapsulate more complex trends.

Recognizing that identifying progressively intricate descriptors necessitates considerable human labor; thus, a technique that could circumvent such arduous processes would be highly desirable. Convolutional neural networks (CNN) are machine learning models that utilize convolutional layers to extract spatial data. Research has demonstrated that three-dimensional (3D) CNNs trained on protein structures may identify protein functional sites [25] and assess protein-ligand interaction sites [26]. Therefore, the likelihood that a 3D CNN will effectively extract spatial features to identify patterns in intricate descriptors inside a standard MD simulation is significant. Chew et al. [3] sought to develop a 3D CNN proficient in forecasting the reaction rates of acid-catalyzed biomass processes in a solvent and co-solvent system, utilizing classical MD data. Their 3D CNN, designated SolventNet, was trained to forecast the kinetic solvent parameter utilizing 76 experimental reaction rates from seven biomass-derived model reactants, accompanied by their corresponding MD trajectories, and evaluated with 32 experimental reaction rates, including those involving cosolvents absent from the training set. The model's performance was assessed using the root mean squared error (RMSE) of the predicted kinetic solvent parameter. A 4 nm^3 box within the simulation cell was divided into 20 voxels per dimension, and the normalized molecular counts were recorded in each voxel for every simulation time step. The technique was reiterated for the reactant, solvent, and cosolvent, and thereafter stored in three channels. The voxelized occupancy distributions were averaged across 2 ns of simulation time to reduce computational expense without compromising model fidelity. SolventNet demonstrated superior predictive accuracy for experimental reaction rates compared to models utilizing human-selected, MD-derived descriptors and reaction rates for polar aprotic cosolvents not included in the training set, relying on merely 4 ns of MD simulation data. The model provides a degree of interpretability via saliency maps, enabling researchers to pinpoint voxels within each channel that exerted the most substantial influence on the model's prediction [3]. SolventNet demonstrates that machine learning may accelerate solvent screening for reaction optimization through classical molecular dynamics simulations with a degree of interpretability. Although MD simulations may explore extended durations with minimum computational cost, AIMD approaches offer mechanistic insights into processes, which may be helpful to researchers. CPMD utilizing metadynamics, a variant of AIMD, may elucidate the energetics of a reaction, a capability unattainable by traditional MD. Given that a machine learning model can diminish the computational expense of molecular dynamics simulations while accurately forecasting their results, it is logical to infer that this concept may also be applicable to ab initio molecular dynamics trajectories. The subsequent studies illustrate this principle.

Häse et al. [27] employed a Bayesian neural network (BNN) to forecast the dissociation time of unmethylated and tetramethylated 1,2-dioxetane, contrasting their findings with those from AIMD simulations. Besides providing precise predictions of dissociation time, Häse et al. sought to elucidate the trained BNN to comprehend the mechanical link between dioxetanes and their respective dissociation time. The dissociation period of 1,2-dioxetanes into two formaldehyde molecules is noteworthy due to its chemiluminescent characteristics [27]. The chemiexcitation yield is directly related to the dissociation duration; thus, examining molecular variations of 1,2-dioxetane is beneficial for identifying which variation will generate the maximum output [28]. Two distinct Bayesian Neural Networks, designated BNN1 and BNN2, were trained. BNN1 forecasted the dissociation time utilizing the initial nuclear geometry of the system, whereas BNN2 employed both the nuclear geometry and velocities for its prediction of the dissociation time. By explicitly forecasting the dissociation time from the system's initial setup, the BNNs circumvent nearly all the computational expenses generally associated with AIMD. The r^2 values for models BNN1 and BNN2, when comparing anticipated and actual values, were determined to be 0.86 and 0.97, respectively. BNN 2 had a reduced mean absolute deviation, indicating superior generalizability of the model. The models also surpassed AIMD regarding computational efficiency. Each model could generate predictions based on 250 distinct initial conditions within a few seconds, but AIMD would require nearly one year of computation to achieve the same outcome. The weights and biases of the BNNs were initialized utilizing the Laplace distribution [27]. Research indicates that following the training of a Bayesian Neural Network (BNN) initialized with the Laplace distribution, the weight magnitudes within the BNN correlate with the relevance of the relevant features in the final prediction [29]. Analysis of the weights from the trained Bayesian Neural Networks revealed that the planarization of the two formaldehyde molecules and the contraction of their C-O bonds correlated with an accelerated dissociation time [27]. This study demonstrated that machine learning applied to AIMD data can yield precise predictions about AIMD outcomes while minimizing computing expenses and offering mechanistic insights into the examined systems.

Eslamibidgoli et al. [30] developed a gated recurrent unit (GRU) and a long short-term memory (LSTM) model to forecast the AIMD trajectories and potential energy profiles of diverse solids. The training and testing dataset was derived using AIMD trajectory data spanning 5000 fs on rutile $Ir_{0.75}M_{0.25}O_2$(100) surfaces, where M represents one of the following metals: Co, Cr, Ir, Mn, Mo, Nb, Ni, Pb, Pt, Rh, Ru, Sn, Ta, Ti, V, and W. Alongside the trajectory data for each atom within the simulated cell, the potential energy profile and temperature changes were also incorporated as input data for each model. A single iridium atom was randomly selected from the training data to evaluate each model. The model's prediction of the Cartesian coordinates for this atom and the total potential energy profile was evaluated against ground truth data from AIMD using mean absolute percentage error (MAPE). The test MAPE for the coordinates of the iridium atom was below 0.9% for the GRU model and 1.3% for the potential energy profile, whereas the LSTM model exhibited MAPEs of 0.2% and 2.1%, respectively. This indicates that both the GRU and LSTM models exhibited high accuracy in predicting the Cartesian coordinates and the potential energy profile. Subsequent analysis revealed that the spatial arrangement of atoms within the simulation cell, along with the distribution of temperature variations and potential energy, conformed to a Gaussian distribution. Given that GRU and LSTM models typically exhibit superior performance with Gaussian distributions, it is understandable why these models excelled [30]. This study shows that GRU and LSTM models can act as substitute models instead of AIMD simulations for solid-state systems. Although the prediction horizon for current models is limited, future, more sophisticated models may enable longer-term forecasts, hence further reducing computing expenses. This principle can also be extended to condensed phase systems, where molecule interactions vary, presenting an additional difficulty for machine learning models.

Puliyanda [31] sought to train a 3D CNN autoencoder to extract characteristics from the AIMD trajectory of a reactant, thereby determining the degree of solvent rearrangement. AIMD simulations of the reactant and product trajectories for the pyrolytic breakdown of cellobiose at four distinct temperatures (100 K, 500 K, 900 K, and 1200 K) were employed to train the model. The root mean squared deviation (RMSD) of the encoded characteristics between the reactant and product was fitted to a probability distribution via kernel density estimation. This probability distribution function was employed to ascertain the degree of solvent rearrangement. Solvent rearrangement was seen to be more pronounced in the cellobiose system at 100 K and 500 K, while it was less noticeable at 900 K and 1200 K. The 3D CNN autoencoder model was evaluated using reactant trajectories from an acid-catalyzed conversion process of fructose to HMF in a water and DMSO solution. A classifier based on Mahalanobis distance was employed to compare the retrieved properties of the reactant trajectory of the new system with those of Class 1. Class 1 includes the training data that showed the most significant changes in solvent arrangement, specifically within the cellobiose system at 100 K. The model predicted a significant decrease in solvent rearrangement when 5% by weight of DMSO was added to a pure water solution. In contrast, DMSO concentrations beyond 5% wt would lead to a linear escalation in solvent rearrangement. This trend aligns with free energy surface minima predictions for the reactant and product states at varying concentrations of water and DMSO. The introduction of % wt DMSO into the system resulted in an increase in the free energy difference (ΔG) between the hydronium ion in the bulk solvent and the hydronium ion in the initial solvation shell of fructose. This indicates that DMSO favored proximity to the fructose molecule, while water molecules stabilized the hydronium ion, leading to a significant reduction in solvent rearrangement. As the

DMSO concentration exceeded 5% wt, ΔG diminished while solvent reorganization escalated. This indicates that the hydronium ions were compelled to engage with the fructose molecule owing to a deficiency of water molecules in the system and heightened solvent rearrangement [31]. This model demonstrates the feasibility of screening processes for their degree of solvent rearrangement across various solvents. This is beneficial because if solvent reorganization is determined to be minimal for a reaction under specific solvent conditions, the mechanics and energetics of the reaction can be analyzed using an implicit solvent model, thereby conserving computational resources and time. If solvent reorganization has a substantial effect, an alternative methodology must be employed to investigate the reaction cost-effectively. This issue could also be addressed via machine learning. Chen et al. [32] employed an active learning framework utilizing machine-learned interatomic potentials (MLIP) to expedite explicit solvent ab initio molecular dynamics (AIMD) simulations of solvated heterogeneous catalytic systems. MLIPs are a category of machine learning models that associate the potential energy of a system with the coordinates of its constituent atoms [33]. It has been demonstrated that MLIPs can aid in the identification of global minima [34] and transition states [35]. An effective MLIP generally necessitates a training set comprising thousands of Density Functional Theory (DFT) calculations, hence imposing a significant computing obstacle to the methodology. Predicting the type of training data that would yield generalizable MLIPs is equally challenging [32]. To address these challenges, a moment tensor potential (MTP) [36] derived from literature was trained in real-time via active learning [32]. Active learning is distinctive because machine learning models taught by this method autonomously select their training data, which is beneficial when labeled data is scarce [37]. Chen et al. employed their machine learning model to conduct machine learning-accelerated molecular dynamics (MLaMD) simulations during the active learning process. Inaccurate predictions were re-evaluated with DFT and incorporated into the training data set. This guarantees that the training dataset expands in comprehensiveness with low resource cost. As the training data expands, the necessity for DFT computations diminishes, consequently elevating the computational expenses of MLaMD simulations. Periodic DFT computations are conducted to maintain the accuracy of the ML model. This study demonstrated that their MLaMD simulations could (i) precisely model the molecular structure of water adjacent to a metal surface, (ii) forecast adsorption energies of CO* and OH* on Cu(111), and (iii) evaluate the energy barrier for C-H bond cleavage in ethylene glycol over Cu(111) and Pd(111) utilizing MLaMD-based metadynamics. This model has several advantages, including a much reduced number of needed DFT computations, simulation times that are four orders of magnitude lower than AIMD, and the capability to access extended time scales [32].

Chapter 3 - Molecular Dynamics Methods

Molecular dynamics (MD) is a computational modeling technique that facilitates the examination of dynamic molecular systems at the atomic scale [38]. The ability to simulate atomic dynamics grants researchers unprecedented insights into chemical processes, although at a high computing cost. Since the 1970s, molecular dynamics (MD) has been employed to complement physical experiments and enhance our understanding [39]. As computer resources become increasingly accessible, molecular dynamics (MD) has begun to assume a more prominent role in chemistry, materials science, and biology. It has been utilized to facilitate medication development [40] and test the characteristics of novel materials [41] and examine chemical reactions [42].

3.1 Classical Molecular Dynamics

There are two primary categories of molecular dynamics (MD): force field-based MD (also known as classical MD) and ab initio molecular dynamics (AIMD) [4]. Both types of molecular dynamics update the spatial arrangement of atoms according to the system's potential energy, although their methods of calculating this potential energy differ. Classical molecular dynamics (MD) employs molecular mechanics (MM) to ascertain the total potential energy, whereas ab initio molecular dynamics (AIMD) utilizes quantum mechanics (QM) [43]. In molecular mechanics, Newton's equations of motion are utilized by considering each atom as a particle with specific size and mass and each bond as a spring characterized by particular length and stiffness. The energy of the forces within a system is represented by a series of equations known as the force field, hence the term "force field-based molecular dynamics" (MD). The modeled forces can be categorized into bound interactions, including stretch, bend, and torsion This includes energy and non-bonded interactions, which comprise van der Waals and electrostatic energies, as indicated in Equation 3.1 [4].

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

The words $E_{stretch}$, E_{bend}, $E_{torsion}$, E_{vdW}, and $E_{electrostatic}$ denote the energy associated with bond stretching between two atoms, angle bending among three atoms, bond twisting, van der Waals interactions, and electrostatic interactions, respectively. Examples of equations for each of these terms are as follows:

$$E_{stretch}^{12} = K_{stretch}(l^{12} - l^{0})^{2} \tag{3.2}$$

$$E_{bend}^{123} = K_{bend}(\theta^{123} - \theta^{0})^{2} \tag{3.3}$$

$$E_{torsion}^{1234} = K_{torsion}(1 - \cos(n\omega)) \tag{3.4}$$

$$E_{vdW}^{12} = 4E_{min}^{12}\left[\left(\frac{R^{0}}{R^{12}}\right)^{12} - \left(\frac{R^{0}}{R^{12}}\right)^{6}\right] \tag{3.5}$$

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

where $K_{stretch}$, K_{bend}, and $K_{torsion}$ are force constants, ω is the angle of rotation, n is periodicity, $E_{min}^{L\text{-}J}$ 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].

Every energy term in Equation 3.1 solely depends on atomic coordinates. Consequently, the lowest energy state of a certain system is determined by minimizing E_{MM} as a function of atomic coordinates [44]. Minimization algorithms are categorized into two types: local and global. Local minimization techniques identify the energy minima in proximity to the system; nevertheless, these minima may not represent the global minimum of the system. The global minima of the system denote the lowest attainable energy state, representing the optimal outcome achieved by minimization. Two conventional approaches for identifying global energy minima are Monte Carlo and molecular dynamics (MD). The Monte Carlo approach identifies global minima by iterative modifications to a system's design. Conversely, molecular dynamics assigns a velocity to each atom in the system according to the system's temperature and classically computes 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 $E(\vec{R})$ and forces $dE/d\vec{R}$
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 E ($\vec{R}_{t+1}$) and forces dE/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

Numerous force fields exist because the parameter values for the energy terms in Equation 3.1 are obtained from experimental data, which yield varying outcomes based on the data utilized [45]. The equations employed to compute each energy term, along with the incorporation of additional energy factors that account for coupled interactions of forces ($E_{coupled}$), represent distinctions across different force fields. The optimal systems for each force field are contingent upon the experimental data used for their parameterization. Consequently, a trade-off arises for each force field, wherein enhanced precision for one system type necessitates a compromise in generalizability [4].

MM approaches are computationally efficient and can effectively describe extensive systems such as proteins and DNA. Nonetheless, in chemically unusual situations where few or no precise parameters are available, the efficacy of molecular mechanics may be diminished. Applying MM to systems including metals and reaction mechanisms is exceedingly challenging. The bonds in metallic systems are less well-defined than those in organic systems. MM lacks a definitive response to this matter as it considers atoms as isolated entities rather than nuclei with associated electronic configurations. MM also deteriorates when addressing reaction systems, as the transformation from reactant to product alters the bonds and perhaps the atom count, necessitating distinct force field energy functions for both the reactant and product [45].

3.2 *Ab-initio* Molecular Dynamics

As outlined in Section 3.1, the fundamental concept of molecular dynamics (MD) involves determining the potential energy, calculating the forces, updating the velocities, and recalibrating the position for the subsequent timestep. In AIMD, quantum mechanics is employed to determine the electronic structure of a system and its potential energy. The Schrödinger equation (Equation 3.7) [46] delineates the spatial and temporal progression of a particle's wavefunction within a potential field.

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

The use of the Born-Oppenheimer approximation to the Shrödinger equation separates electronic and nuclear motion, enabling their resolution using quantum mechanical and classical methods, respectively. This enables the optimization of the electronic structure according to the nuclear configuration after each time step. This is referred to as Born-Oppenheimer molecular dynamics (BOMD). The Lagrangian for BOMD is presented in Equation 3.8. Nuclei may be analyzed classically due to their substantial mass, which allows them to conform to Newtonian physics. Consequently, the solution to the Schrödinger equation for electrons within a system (Equation 3.9) is needed for determining the potential energy [4].

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

$$H\Psi(R, R_e) = E\Psi(R, R_e) \tag{3.9}$$

In the aforementioned equations, H denotes the Hamiltonian operator, $\hbar$ represents the Planck constant, Ψe signifies the wavefunction for electrons, R_e refers to the electronic coordinates, and R_n indicates the nuclear coordinates. Due to the inability to analytically calculate Equation 3.9 for systems with many electrons, an approximation is necessary for polyatomic systems [44]. Density functional theory (DFT) is a quantum mechanical computational approach that use electron density rather than individual electron wavefunctions to determine a system's energy. Hohenberg and Kohn demonstrated that an external potential v($\vec{R}$) can be ascertained if the electron density is known, and that wavefunctions can be articulated as a functional of electron density [47]. This simplifies the energy computation, as only the density function is required [4]. Kohn and Sham proposed a method to compute the energy of a multi-electron interacting system using DFT, minimizing computational costs [48]. Kohn-Sham DFT presents a set of fictitious noninteracting electrons characterized by Kohn-Sham orbitals χ_{KS} within a potential $v_{KS}(\vec{R})$. The Kohn-Sham orbitals are selected to match the electron density ρ of the interacting system in the actual potential v($\vec{R}$). Equation 3.10 illustrates the calculation of energy based on density. The energy in Equation 3.10 must be minimized about density according to the variational principle [4]. The variational principle asserts that the energy computed using approximation wavefunctions will invariably exceed the true energy of the system [45]. Consequently, the ground state density must be computed iteratively to identify the density that minimizes energy. The Kohn-Sham DFT enables the quantum mechanical resolution of a system's potential energy, facilitating the execution of ab initio molecular dynamics (AIMD).

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

AIMD possesses a considerable advantage over classical MD approaches, as the constraints of classical MD do not apply to AIMD. Chemically unusual systems, metallic systems, and reaction systems may now be precisely represented using AIMD. Nonetheless, the significant computing expense linked to AIMD, coupled with the pronounced trade-off between computational costs and precision, renders AIMD infrequently viable [4].

The substantial computational expense of AIMD arises from the necessity to optimize the electron density after each time step to guarantee the system attains its energy minima. [49] To address this, Car and Parrinello illustrated how the transition of the ideal electronic structure across time steps might be simulated using a classical framework (Car-Parrinello molecular dynamics (CPMD)). This is accomplished by attributing a fictional mass μ to the wavefunctions, thus endowing them with fictitious kinetic energy. The extended Lagrangian for Car-Parrinello molecular dynamics (CPMD) is delineated as follows [50]:

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

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

$$\begin{aligned} &\frac{d}{dt}\left(\frac{\partial \mathcal{L}_{CPMD}}{\partial \dot{R}_i}\right) - \frac{\partial \mathcal{L}_{CPMD}}{\partial R_i} = 0 \\ &\frac{d}{dt}\left(\frac{\partial \mathcal{L}_{CPMD}}{\partial \dot{\chi}_i}\right) - \frac{\partial \mathcal{L}_{CPMD}}{\partial \chi_i} = 0 \\ &m_i \ddot{R}_i = -\frac{\partial}{\partial R_i}\langle \Psi | H | \Psi \rangle + \frac{\partial}{\partial R_i}(\text{constraints}) \\ &\mu_j \ddot{\chi}_j = -\frac{\partial}{\partial \chi_i}\langle \Psi | H | \Psi \rangle + \frac{\partial}{\partial \chi_i}(\text{constraints}) \end{aligned} \tag{3.12}$$

which can solve 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].

The metadynamics approach was created to expedite atomic events for observation within CPMD's brief timescale [51]. Metadynamics introduces minor energy potentials to populate a system's free energy surface (FES) inside the relevant collective variable (CV) space. The selected CV space is contingent upon the study's purpose; examples of CVs include bond length, bond angle, bond distance, or coordination numbers. This method necessitates an augmented CPMD Lagrangian, defined as [52]:

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

where s is the vector of CVs, mCV is the fictitious mass assigned to the collective variables,

KCV is the force constant, $K_{CV}\ [S_{CV}(\vec{R}_{CV}) - s_{CV}]$ is the potential energy acting on the

CVs and v_{CV} (t,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 v_{CV} (t,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.

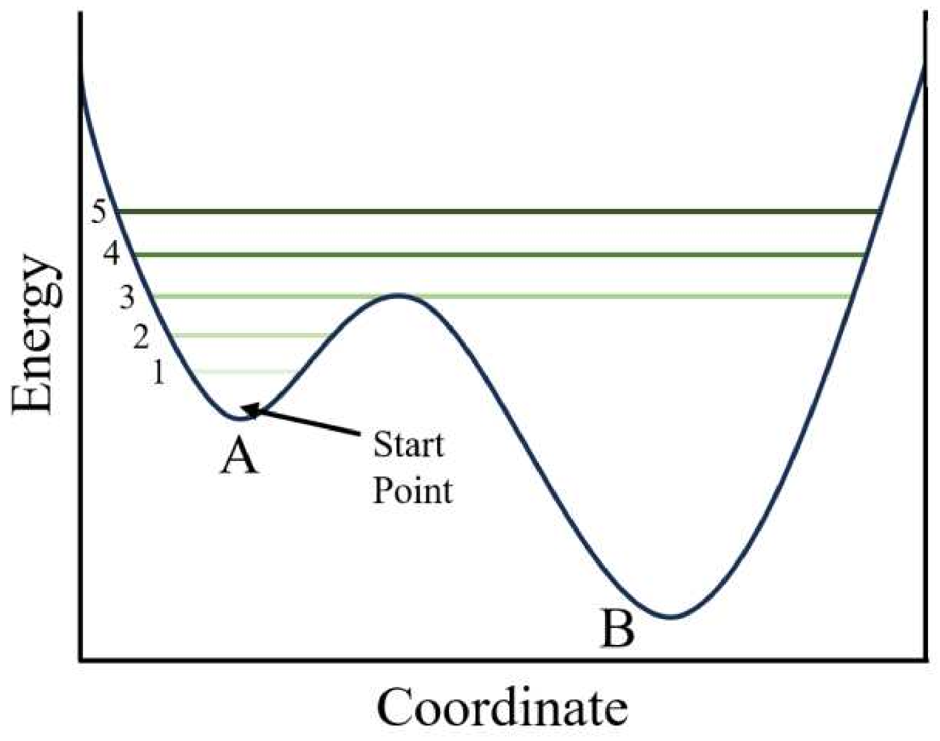


Figure 3.1: Example of how metadynamics explores a FES for a system that goes from state A to a more stable state B. The potential energy surface is filled starting from "well" A as shown by lines 1 and 2. As the "well" gets filled up it spills into "well" B as shown with line 3. Then the FES is filled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

Chapter 4 - Leveraging attention mechanism to predict reaction energetics for gas phase reaction system

4.1 Introduction

This chapter discusses the reaction system considered in gas phase. Additionally, it also covers the literature review on all advancements in machine learning that aims to understand, extract and draw insights of temporal information from the data available.

The use of deep learning to ab initio molecular dynamics (AIMD) simulations of gas phase reaction systems, entails a two-step procedure. Initially, it necessitates generating a time-series forecast of the system's spatial arrangement. This spatial arrangement is subsequently employed to forecast the system's free energy for that time step. This chapter seeks to develop a model that can serve as a surrogate for Car-Parrinello molecular dynamics (CPMD) metadynamics simulations based on a brief initial simulation. Consequently, a time-series forecast of the simulation is essential. The results from CPMD simulations conducted with metadynamics can be utilized as training data for a series of models that can tackle this two-step procedure. CPMD, as delineated in Chapter 3, is a variant of AIMD simulation that estimates the dynamics of the electronic structure through a classical framework rather than employing a quantum approach to recalibrate the electronic structure at each time step, thus mitigating the computational cost of the simulation [50]. Nonetheless, the accessible time scale of CPMD is confined to picoseconds [4]. When combined with the metadynamics technique that enhances system dynamics, a broader time scale becomes accessible. Metadynamics affects the system by intermittently introducing minor energy potentials to populate the free energy surface (FES) within the coordinate domain of a designated set of collective variables (CVs) [51]. These CVs are meticulously chosen according to the system under examination. Furthermore, metadynamics facilitates the reconstruction of a FES post-simulation [53]. This reconstruction facilitates the examination of reaction systems by supplying the free energy of the reactant, transition, and product states. By pre-processing the trajectory and free energy surface (FES) data from the CPMD-metadynamics simulation, this dataset can function as training and testing data for machine learning (ML) models to accomplish the aforementioned purpose.

4.2 Reaction system and data pre-processing

This system examines the hydride transfer step; thus, the comprehensive isomerization reaction mechanism, which is widely recognized in the literature, was not simulated for the considered dataset [54,55,56]. The catalyst utilized was the partially hydrolyzed Sn-defect site, characterized by Sn tetrahedrally coordinated to three oxygen atoms of the zeolite framework, together with a –OH group [57,58-63]. The initial phase of the isomerization reaction involves the coordination of glucose to the Sn metal center, resulting in the creation of an octahedral complex, wherein Sn is further coordinated with the C_1 carbonyl and C_2 hydroxyl groups of glucose (cf. Fig. 1a [54]). Prior to the hydride transfer, glucose undergoes deprotonation at the C_2–OH group, with the Sn–OH defect site accepting the proton, resulting in the creation of a nascent water molecule that remains coupled to Sn. Sn establishes a covalent connection with C_2 oxygen; C_2–O–Sn. The specific configuration of the deprotonated glucose molecule, bound to the Sn defect site, was utilized as the initial structure to model the C_2– C_1 hydride transfer step (see Fig. 1b [54]). The dimensions of the simulation cell were 18 Å, 14.5 Å, and 16 Å. Once the reactant and catalyst system were positioned isolated in the simulation cell. The collective variables (CV) employed in the metadynamics simulation of hydride transfer are shown in Fig. 1c [54]. One hydrogen is linked to each C_1 and C_2 before to the hydride transfer, resulting in $CV_1 \approx CV_2 \approx 0.9$. Subsequent to the hydride transfer, $CV1 \approx 1.8$ and $CV2 \approx 0.1$ (refer to Fig. 1c).

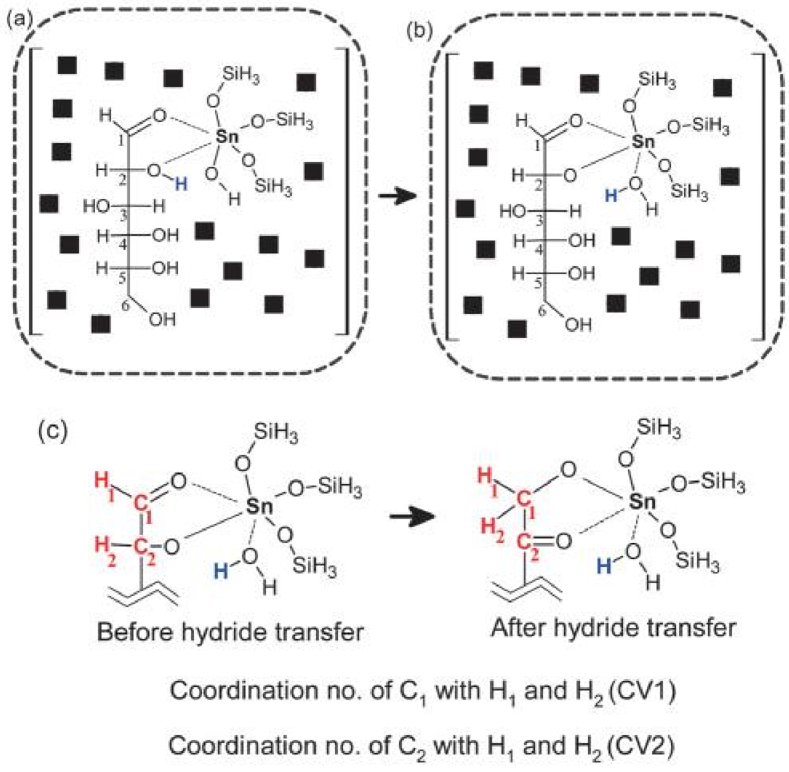

(a)
(b)
(c)
SiH3
O-SiH3
Sn
Before hydride transfer
After hydride transfer
Coordination no. of C1 with H1 and H2 (CV1)
Coordination no. of C2 with H1 and H2 (CV2)

Figure 4.1: (a) Glucose coordinated with the Sn-beta cluster model, representing the active catalyst site, via C_1 carbonyl and C_2 hydroxyl groups. Sn is octahedrally coordinated with 6 oxygen atoms. (b) Deprotonated glucose attached to Sn via C_2 oxygen. The proton from the C_2 hydroxyl group of glucose is taken by the –OH group attached to Sn to form a water molecule. This is the representation of the simulation cell. Black markers represent solvent molecules in the simulation cell. (c) C_2–C_1 hydride transfer and the definition of collective variables CV_1 and CV_2, implemented in metadynamics. Atoms that form the collective variables are shown in red.

CN of one selected atom *i* with respect to one selected species, *jsp*, or a list of atoms, *j*1 · · · jn_{list} can be mathematically formulated. The CN value is calculated as the sum of rational as given in equation 4.1 [88]:

$$CN_i = \sum_{j \neq i}^{n_{list}} \frac{1 - \left(\frac{d_{ij}}{d^0}\right)^p}{1 - \left(\frac{d_{ij}}{d^0}\right)^{p+q}} \tag{4.1}$$

Where *p* is numerator exponent, *q* is additional exponent and d^0 is the reference distance.

CPMD outputs give us the trajectory files consisting x,y and z co-ordinates for each atom in the simulation box at each timestep. The output file has 20,000 timesteps. From equation 4.1 and figure 4.1, suggests to take the particular atoms C_1, C_2, H_1 and H_2 into consideration for the modeling. So from the trajectory file, the coordinates of these 4 atoms are extracted for all 20,000 timesteps and paired up to form a 2D array with rows as the timesteps and columns as the x,y and z of coordinates of atom 1 and atom 2 respectively in each pair. Pairs considered are C_1 and H_1, C_1 and H_2, C_2 and H_1, C_2 and H_2. These 2D arrays will be used as the training data to predict coordinates of future timestep, which gives the predicted Euclidean distance, which in turn gives the predicted CN and CV for the system. These predicted CVs will be later used to reconstruct the free energy surface.

4.3 Literature review on modeling for time-series forecasting

This section establishes the foundational context for the study of Long Short-Term Memory (LSTM) networks in time series forecasting. It begins by defining the nature of time series data, its constituent components, and the critical concept of stationarity. Subsequently, it outlines the primary objectives of forecasting and introduces the challenge of non-linearity, which motivates the transition from classical statistical models to more sophisticated machine learning and deep learning approaches.

4.3.1 The Nature and Components of Time series Data

A time series is formally defined as a sequence of data points collected over time, typically at equally spaced intervals such as hourly, daily, or monthly [64]. The analysis of such data aims to uncover the underlying structure and patterns within the historical observations to develop a model capable of predicting future values [64].

A concept of paramount importance in classical time series analysis is stationarity. A time series is considered stationary if its statistical properties; specifically its mean, variance, and autocorrelation structure remain constant over time [65]. In essence, a stationary series does not exhibit predictable long-term patterns, and a visual plot of the data would appear roughly horizontal with a constant variance [65]. This property is a fundamental assumption for many statistical forecasting methods, including the ARIMA model. If a series is non-stationary, it must often be transformed into a stationary series, typically through a mathematical operation known as differencing, before such models can be effectively applied [65].

4.3.2. Objectives of Forecasting and the Challenge of Non-Linearity

The primary objective of time series forecasting is to develop a model that can accurately predict future values of a series based on its observed past [64]. The ultimate goal is to minimize the deviation between the forecasted values and the actual, subsequently observed values. This is often quantified by minimizing a loss function, such as the mean squared error, for a given forecast horizon or lead time [66]

Accurate forecasts serve as a critical basis for planning, control, and optimization across a vast array of practical domains. In business and economics, they inform inventory management, production planning, and financial market speculation [66] In science and engineering, they are used to predict weather patterns, control industrial processes, and manage energy consumption [64].

A significant challenge in modern time series forecasting arises from the fact that many real-world phenomena are governed by complex, non-linear processes [66] Traditional statistical models, which are often built on an assumption of linearity, may be fundamentally inappropriate when the underlying data-generating mechanism is nonlinear. For example, financial market returns are known to exhibit volatility clustering and other complex dynamics that cannot be adequately captured by linear models. This inadequacy of linear models to represent intricate, real-world relationships is a primary driver behind the adoption of more flexible and powerful machine learning models, such as neural networks, which can approximate arbitrary non-linear functions.

4.3.3. A Primer on Traditional Statistical Models: The ARIMA Framework

Among the classical statistical methods for time series forecasting, the Autoregressive Integrated Moving Average (ARIMA) model stands as a cornerstone and a powerful benchmark for problems where linear relationships dominate [64]. It is a class of stochastic models that captures temporal structures in time series data to produce forecasts [66].

The ARIMA model is denoted as ARIMA(p, d, q) and is characterized by three parameters that describe its key components [66]:

- **p (Autoregressive - AR):** This parameter represents the order of the autoregressive component, indicating the number of lagged observations of the time series included in the model. The AR component models the linear relationship between an observation and its own previous values. An AR(p) model asserts that the current value y_t is a linear combination of the previous p values.
- **d (Integrated - I):** This parameter represents the degree of differencing required to make the time series stationary. Differencing involves computing the difference between consecutive observations. This component is crucial for handling non-stationary data, particularly data with a trend. If a series must be differenced d times to become stationary, the model is said to be integrated of order d [64].
- **q (Moving Average - MA):** This parameter represents the order of the moving average component, indicating the number of lagged forecast errors included in the prediction model. The MA component models the effect of random shocks or white noise on the observed value, assuming that the current observation is a linear combination of past error terms.

The mathematical formulation of the ARIMA model effectively combines the AR and MA components into a single, comprehensive equation that is applied to the differenced, stationary time series. For a series y_t that has been differenced d times to produce $y_{t'}$, the model can be expressed [66] as equation 4.2:

$$y'_t = c + \sum_{i=1}^{p} \phi_i \, y'_{t-i} + \sum_{j=1}^{q} \theta_j \, \epsilon_{t-j} + \epsilon_t \tag{4.2}$$

Here, c is a constant term, ϕ_i are the autoregressive parameters, θ_j are the moving average parameters, and ϵ_t is the white noise error term at time t. The process of building an ARIMA model, known as the Box-Jenkins methodology, typically involves an iterative process of model identification (using tools like the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to determine p and q), parameter estimation, and diagnostic checking.

Despite its widespread use and effectiveness for a certain class of problems, the ARIMA model's primary limitation is its fundamental assumption of linearity [66]. It presupposes that the underlying stochastic process generating the data is linear. This makes it inherently unsuitable for many complex, real-world time series where the relationships between past and future values are non-linear. The transition from ARIMA to neural network models like LSTMs represents a significant paradigm shift. ARIMA exemplifies a model-driven approach: it assumes that the data is generated by a specific, interpretable linear process, and the analyst's task is to identify the parameters (p, d, q) of this pre-specified model. This approach offers high interpretability but is constrained by its rigid structural assumptions.

In contrast, LSTMs represent a data-driven approach. As universal function approximators, they make minimal assumptions about the underlying data-generating process. Instead of being confined to a linear structure, they possess the flexibility to learn complex, non-linear relationships directly from the data [67]. This transition involves a trade-off: the clear interpretability of ARIMA's parameters is exchanged for the superior predictive power and flexibility of LSTMs on complex data. This fundamental difference in philosophy and capability is what necessitates the exploration of deep learning models for modern forecasting challenges. To crystallize these distinctions, the following table 4.1 provides a comparative overview of the ARIMA and LSTM approaches, serving as a conceptual bridge from classical statistical methods to the neural network architectures that are the focus of this report.

Feature	ARIMA (Autoregressive Integrated Moving Average)	LSTM (Long Short-Term Memory)
Model Type	Statistical, Linear	Machine Learning, Non-Linear
Underlying Assumption	Data is generated by a linear stochastic process.	No strong assumption; learns arbitrary mappings from data.
Data Requirements	Requires stationary data (achieved via differencing).	Does not require stationarity; can learn trends directly.
Temporal Dependencies	Best for short-term dependencies.	Explicitly designed to capture long-term dependencies.
Parameters	(p, d, q) - Interpretable statistical parameters.	Network weights and biases - Not directly interpretable.

Complexity	Low computational complexity.	High computational complexity; requires significant data and GPU.
Use Case	Well-suited for problems with clear linear structure.	Excels at complex, non-linear time series (e.g., finance, weather).

Table 4.1: Comprehensive analysis for ARIMA and LSTM model

4.3.4 Simple Recurrent Networks

Unlike traditional feedforward neural networks, which process inputs independently, Recurrent Neural Networks (RNNs) are specifically designed for sequential data. Their defining feature is the presence of feedback loops or recurrent connections (in figure 4.2), which form directed cycles within the network's structure. This recurrence allows the network to maintain an internal state, often referred to as a "hidden state" or "memory," which persists across time steps [68]. This memory enables the network to capture information about past events in a sequence, making it naturally suited for tasks like time series forecasting, natural language processing, and speech recognition [68].

The operational mechanism of a simple RNN involves iterating through the elements of an input sequence one time step at a time. At each time step t, the RNN cell takes two inputs: the current data point from the sequence, x_t, and the hidden state from the immediately preceding time step, h_{t-1}. It combines these inputs to compute the new hidden state, h_t. This new hidden state then serves as the memory that is passed forward to the next time step. The hidden state h_t can be conceptualized as a compressed, distributed representation of the entire history of the sequence observed up to time t [68]. The network can then use this hidden state to produce an output, y_t, for the current time step.

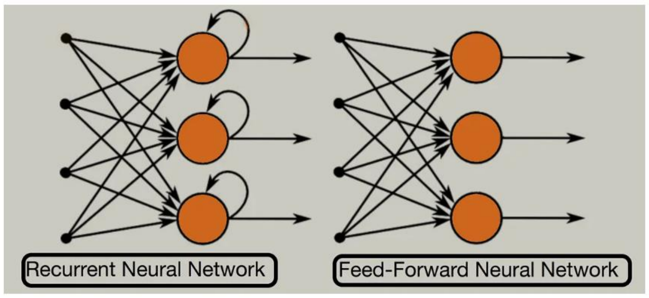


Figure 4.2: Mechanism for RNN

The state transitions and output generation in a simple RNN are governed by a set of equations that are applied consistently at every time step. The core mathematical formulation is as follows [68] in equation 4.3 and 4.4:

$$h_t = \sigma_h(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{4.3}$$

$$y_t = \sigma_y(W_{hy}h_t + b_y) \tag{4.4}$$

In these equations:

- x_t is the input vector at time step t.
- h_{t-1} is the hidden state vector from the previous time step $t-1$.
- h_t is the new hidden state vector at time step t.
- y_t is the output vector at time step t.
- W_{hh}, W_{xh}, and W_{hy} are the weight matrices that are shared across all time steps. W_{hh} governs the recurrent connection (hidden-to-hidden), W_{xh} governs the input connection (input-to-hidden), and W_{hy} governs the output connection (hidden-to-output).

- b_h and b_y are the bias vectors for the hidden and output layers, respectively.
- σ_h and σ_y are non-linear activation functions. The hyperbolic tangent (tanh) is a common choice for the hidden state activation σ_h, as its output range of (-1, 1) can help regulate the information flow.

The sharing of the same weight matrices (W_{hh}, W_{xh}, W_{hy}) across all time steps is a crucial aspect of RNNs. It not only reduces the number of parameters to be learned but also allows the model to apply the same learned transformation to inputs at different positions in the sequence.

4.3.5 The Challenge of Long-Range Dependencies: Vanishing and Exploding Gradients

While RNNs are theoretically capable of capturing dependencies between events that are arbitrarily far apart in a sequence, in practice, their ability to do so is severely limited. Training RNNs on long sequences is notoriously difficult due to a fundamental problem with how gradients flow through the network over time. This issue, known as the vanishing and exploding gradient problem, causes the error signals that are propagated backward in time during training to either decay exponentially to zero or grow exponentially to an overflow value [69].

The consequences of this unstable gradient flow are profound:

- **Vanishing Gradients**: When the gradient signal shrinks exponentially, it becomes so small by the time it reaches earlier time steps that the weight updates for those steps become negligible. This effectively prevents the model from learning the influence of early inputs on later outputs, making it impossible to capture long-term dependencies [70]. The network essentially becomes unable to "remember" information from the distant past.

- **Exploding Gradients**: When the gradient signal grows exponentially, it leads to excessively large updates to the weight matrices. This causes the learning process to become unstable, and the model's weights can diverge to infinity, resulting in numerical overflow (NaN values) and a complete failure of the training process [71].

4.3.6 Long Short-Term Memory (LSTM) Networks

This section presents the Long Short-Term Memory (LSTM) network, a seminal and highly influential architecture designed as a direct solution to the long-range dependency problem that plagues simple RNNs. The chapter begins by tracing the conceptual evolution of LSTMs from their historical origins, focusing on the core ideas that allow them to overcome the unstable gradient problem. It then provides a detailed architectural dissection of the fundamental building block of an LSTM network, the LSTM cell, and its constituent gating mechanisms. Finally, the chapter culminates in a complete mathematical formalization of the information flow within the LSTM cell, providing the precise equations that govern its operation.

Long Short-Term Memory networks were introduced by Sepp Hochreiter and Jürgen Schmidhuber in their 1997 paper, which has since become one of the most cited works in the history of deep learning [67]. The architecture was explicitly engineered to address the vanishing and exploding gradient problems, with the stated goal of enabling recurrent networks to learn to bridge time intervals in excess of 1000 steps [67].

The central innovation of the LSTM is the introduction of a more complex repeating module, the LSTM cell, which replaces the simple neuron of a standard RNN. This cell incorporates two key concepts: a dedicated memory pathway and a set of regulatory gating mechanisms.

- **The Cell State and the Constant Error Carousel:** The core idea behind the LSTM is the introduction of a cell state, denoted as C_t. This cell state acts as an explicit memory component, conceptually akin to a conveyor belt. It runs horizontally through the entire sequence of cells, allowing information to flow along it with only minor, well-controlled linear interactions [67]. This design creates a direct, uninterrupted path for the gradient to flow backward through time. Hochreiter and Schmidhuber referred to this mechanism as the "constant error carousel" (CEC), as it enforces a constant error flow and prevents the gradient signal from vanishing or exploding due to repeated non-linear transformations [67]. The LSTM can learn to store information in the cell state for long periods, making this its default behavior rather than something it struggles to achieve [67].
- **Gating Mechanisms:** While the cell state provides a pathway for information, the flow along this pathway must be regulated. LSTMs achieve this through gating mechanisms. These gates are structures composed of a sigmoid neural network layer and a pointwise multiplication operation. The sigmoid layer outputs values between 0 and 1, which act as a control signal, describing how much of each component of information should be allowed to pass through. A value of 0 signifies "let nothing through," while a value of 1 signifies "let everything through" [67]. The LSTM cell employs three such gates to carefully protect and control the cell state: the forget gate, the input gate, and the output gate. These gates are small, learnable neural networks that dynamically decide when to

allow information to enter the cell state, when to erase information from it, and when to let the cell state influence the network's output [67].

4.3.7 Architectural Deep Dive: The LSTM Cell and Gating Mechanisms

The LSTM cell is the fundamental repeating module of an LSTM network. Its internal structure is designed to implement the concepts of the cell state and gating mechanisms. A visual representation of the cell's architecture and data flow is provided in Figure 4.3.

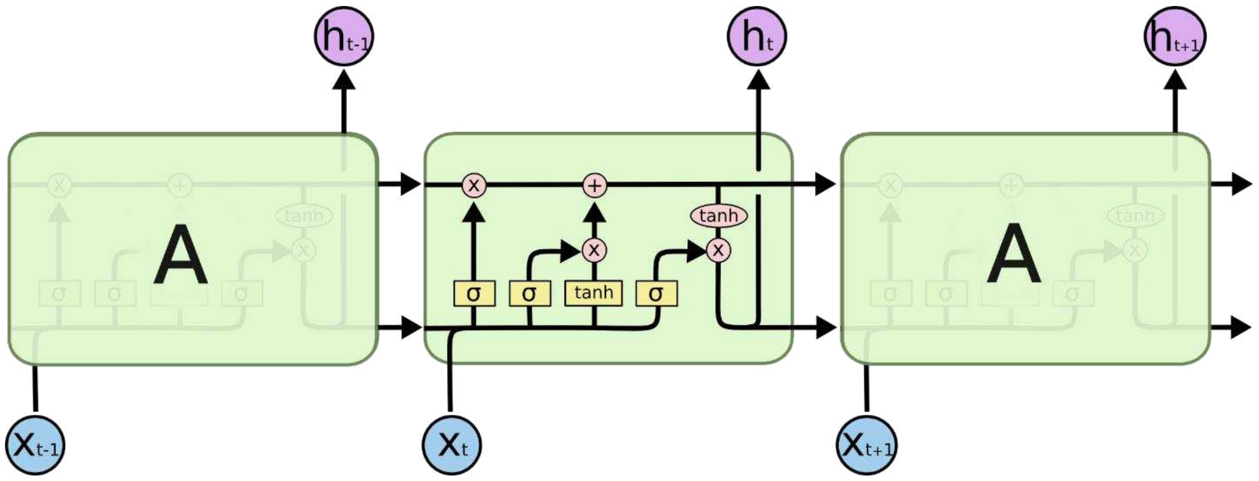


Figure 4.3: The repeating module in an LSTM contains four interacting layers. The cell state runs horizontally across the top, regulated by the gating mechanisms.

The key components of the LSTM cell are:

- **Cell State (C_t)**: This is the core of the LSTM and represents its long-term memory. It flows through the entire chain of cells, and the gates can perform additions or removals of information to it. Its largely linear interaction path is what preserves the gradient during backpropagation [67].

- **Forget Gate (f_t)**: This gate is responsible for deciding what information to discard from the previous cell state, C_{t-1}. It takes the previous hidden state h_{t-1} and the current input x_t as inputs and passes them through a sigmoid activation function. The resulting output vector, with values between 0 and 1, is then pointwise multiplied with the previous cell state. A '1' indicates that the corresponding information in C_{t-1} should be kept, while a '0' indicates it should be forgotten [67].
- **Input Gate (i_t)**: This gate determines what new information should be stored in the cell state. This process has two parts. First, a sigmoid layer (the "input gate layer") decides which values in the cell state will be updated. Second, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could potentially be added to the state. These two are then combined to update the cell state [67].
- **Output Gate (o_t)**: This gate determines the cell's output for the current time step, which becomes the new hidden state h_t. The output is a filtered version of the cell state. First, the cell state C_t is passed through a tanh function to scale its values to between -1 and 1. Then, a sigmoid layer, using h_{t-1} and x_t as inputs, decides which parts of the cell state should be exposed. The output of the tanh and the sigmoid gate are then pointwise multiplied to produce the final hidden state h_t [67].

This architecture addresses the critical issues of simple RNNs. The cell state, C_t, is primarily responsible for transporting information through time, while the hidden state, h_t, is responsible for exposing a relevant, non-linear interpretation of that information to the rest of the network at each time step. The update mechanism for the cell state is fundamentally additive ($C_t =$ forget $+$ add), which is the key to preserving the gradient signal over long distances. In contrast, the hidden state update in a simple RNN involves passing the entire memory through a non-linear squashing function at every step, which corrupts the long-term signal. The LSTM's gates, through their multiplicative interactions, act as dynamic, learned controllers that manage this more stable, additive memory-transport channel. This architectural decoupling is the masterstroke of the LSTM design, allowing it to maintain a stable long-term memory while still producing complex, non-linear outputs.

4.3.8. Mathematical Formulation of Information Flow in LSTMs

The intricate operations within an LSTM cell at a given time step t can be precisely described by a set of mathematical equations. These equations formalize the functions of the gates and the updates to the cell and hidden states. The following formulation is a synthesis of the consistent definitions found across authoritative sources [71].

1. **Forget Gate (f_t):** The sigmoid layer decides what proportion of the previous cell state to keep (equation 4.5).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.5}$$

2. **Input Gate (i_t) and Candidate State ($\tilde{C}_t$):** The input gate's sigmoid layer decides which values to update, and the tanh layer creates new candidate values (4.6 and 4.7).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.6}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.7}$$

3. **Cell State Update (C_t)** The old cell state C_{t-1} is multiplied by the forget gate f_t, and the new candidate values $\tilde{C}_t$ are multiplied by the input gate i_t. The results are added together to create the new cell state (equation 4.8).

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4.8}$$

4. **Output Gate (o_t) and Hidden State (h_t)** The output gate's sigmoid layer decides which parts of the cell state to output, which is then multiplied by the tanh of the cell state to produce the new hidden state (equation 4.9 & 4.10).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.9}$$

$$h_t = o_t \odot \tanh(C_t) \tag{4.10}$$

The notation used in these equations is as follows:

- x_t: The input vector at time step *t*.
- h_{t-1}: The hidden state vector from the previous time step *t-1*.
- C_{t-1}: The cell state vector from the previous time step *t-1*.
- $[h_{t-1}, x_t]$: Concatenation of the previous hidden state and current input vectors.
- W_f, W_i, W_C, W_o: The learnable weight matrices for the forget gate, input gate, candidate state, and output gate, respectively.
- b_f, b_i, b_C, b_o: The learnable bias vectors for each corresponding component.
- σ: The sigmoid activation function, which squashes values to the range (0,1).
- $\tanh$: The hyperbolic tangent activation function, which squashes values to the range [-1,1].
- $\odot$: The element-wise (Hadamard) product operation.

To provide a clear and accessible reference, the core mathematics of the LSTM cell are summarized in the table below (Table 4.2).

Component	Equation	Purpose
Forget Gate	$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$	Determines which parts of the old cell state C_{t-1} to discard.
Input Gate	$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$	Determines which parts of the new candidate information $\tilde{C}_t$ to add.
Candidate State	$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$	Generates new information that could be added to the cell state.
Cell State Update	$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$	Updates the long-term memory by forgetting old info and adding new info.

Table 4.2: Core Mathematics of the LSTM Cell

4.3.9 A Practical Framework for LSTM-Based Time Series Forecasting

Transitioning from the theoretical underpinnings of LSTMs to their practical application requires a structured, systematic workflow. This chapter details the essential end-to-end framework for employing LSTMs in a real-world forecasting context. It covers the critical preliminary steps of data preprocessing and transformation, which are necessary to prepare sequential data for a neural network. It then delves into the mechanics of model training, including the selection of appropriate loss functions and optimizers, and concludes with a discussion of key regularization techniques used to prevent overfitting and enhance model generalization.

4.3.10 Data Preprocessing and Transformation for Sequential Learning

Raw time series data is rarely in a format suitable for direct input into a neural network. Several preprocessing steps are required to structure the data appropriately and to ensure stable and efficient model training.

- **Normalization and Standardization**: A fundamental prerequisite for training most neural networks, including LSTMs, is feature scaling. LSTMs are sensitive to the scale of their input data, and large input values can lead to unstable gradients and slow convergence. Therefore, it is standard practice to scale the time series data to a small, consistent range. The two most common methods are :

  a. **Min-Max Normalization**: This technique linearly transforms the data to a specific range, typically or [-1, 1]. The formula for scaling a value x is: $x_{scaled}=(x-x_{min})/(x_{max}-x_{min})$. This is a common choice when the data is known to be bounded within a certain range.

  b. **Z-score Standardization:** This method transforms the data to have a mean of 0 and a standard deviation of 1. The formula is: $x_{standardized}=(x-\mu)/\sigma$, where μ is the mean and σ is the standard deviation of the training data. This is often preferred when the data contains outliers or does not have a known bounded range.

  It is crucial that the scaling parameters (e.g., min/max or mean/std) are computed only on the training dataset and then applied consistently to the validation and test datasets to prevent data leakage from the future into the training process.

- **The Sliding Window Method**: The most fundamental data transformation technique for time series forecasting with supervised learning models is the sliding window method. This approach reframes the forecasting task from an unstructured sequence prediction problem into a standard supervised learning problem with well-defined input-output pairs (X, y) [72]. The process involves:

    a. **Defining Window Parameters**: Two key parameters are chosen: the input_length (or window size), which specifies the number of past time steps to use as input features, and the forecast_horizon, which specifies how many future time steps to predict as the target.

    b. **Creating Samples**: A window of size input_length is slid across the time series, typically one step at a time. For each position of the window, the sequence of data points inside the window forms a single input sample (X), and the data point(s) immediately following the window form the corresponding target label (y).

The resulting dataset can then be used to train the LSTM model. The choice of window size is a critical modeling decision. It implicitly encodes an assumption about the effective "memory" or "Markov horizon" of the system being modeled. By selecting a window of size N, the modeler asserts that the information required to predict the next time step is primarily contained within the last N observations. While the LSTM's internal cell state can theoretically carry information from beyond this window, the most direct and high-fidelity signal at each training step comes from the current input window. A window that is too small may fail to capture essential long-term patterns, while one that is too large may introduce noise, increase computational cost, and require more data to train effectively. This choice should therefore be informed by both domain knowledge about the process and empirical analysis of the data's autocorrelation structure.

4.3.11 Model Training, Optimization, and Regularization

Once the data is preprocessed, the next phase involves defining, training, and refining the LSTM model.

- **Model Architecture Definition**: The first step is to specify the structure of the neural network. This includes deciding on the number of LSTM layers (e.g., a single layer or a stacked architecture), the number of hidden units (neurons) within each LSTM layer, and the structure of the final output layer. For a standard regression forecasting task (predicting a continuous value), the output layer is typically a Dense (fully connected) layer with a single neuron and a linear activation function [73].
- **Loss Function**: The loss function quantifies the error between the model's predictions and the true target values. The goal of training is to adjust the model's weights to minimize this function. For regression tasks like time series forecasting, the most common choice is the Mean Squared Error (MSE), which is the average of the squared differences between predictions and actuals. MSE is differentiable and penalizes larger errors more heavily, which is often a desirable property [74]. The Mean Absolute Error (MAE) is an alternative that is less sensitive to outliers.

- **Optimizer**: The optimizer is the algorithm that implements the backpropagation process, updating the model's weights based on the gradients of the loss function. The Adam (Adaptive Moment Estimation) optimizer is a widely used and highly effective choice. It combines the advantages of two other popular optimizers, AdaGrad and RMSProp, by computing adaptive learning rates for each parameter. It is generally robust and performs well across a wide range of problems with little need for hyperparameter tuning [75].
- **The Training Loop**: The model is trained iteratively on the training data. The process involves:
    - **Epochs**: An epoch is one complete pass through the entire training dataset. The model is typically trained for multiple epochs to allow the weights to converge to an optimal state [74].
    - **Batches**: To manage memory and improve training dynamics, the training data is divided into smaller batches. The model's weights are updated after processing each batch [76].
    - **Validation**: During training, it is crucial to monitor the model's performance on a separate validation set, a portion of the data that the model does not see during training. This provides an unbiased estimate of the model's generalization ability and is key for detecting overfitting [77].
- **Regularization Techniques**: Overfitting occurs when a model learns the training data too well, including its noise, and fails to generalize to new, unseen data. Several techniques are used to combat overfitting:

- **Dropout**: This is one of the most effective and commonly used regularization techniques. During each training update, Dropout randomly sets a fraction of neuron activations in a layer to zero. This prevents neurons from co-adapting too much and forces the network to learn more robust and redundant representations [78].
- **L1 and L2 Regularization**: These techniques add a penalty term to the loss function based on the magnitude of the model's weights. L2 regularization (also known as weight decay) penalizes the sum of the squared weights, encouraging smaller, more diffuse weight values. L1 regularization penalizes the sum of the absolute values of the weights, which can lead to sparse weight vectors (some weights becoming exactly zero) [79].
- **Early Stopping**: This technique involves monitoring the loss in the validation set during training. If the validation loss fails to improve for a specified number of consecutive epochs (the "patience" parameter), the training process is halted. The model weights from the epoch with the best validation performance are then restored. This is a simple yet powerful method to prevent the model from continuing to train after it has started to overfit [74].

4.3.12 Quantitative Evaluation Metrics for Forecasting Accuracy

After a forecasting model has been trained and has generated predictions on a test dataset, a rigorous evaluation is necessary to assess its performance and reliability. This final chapter details the standard methods for this evaluation process. It first covers the essential quantitative metrics used to provide objective, numerical measures of forecasting accuracy. It then highlights the importance of complementing these metrics with qualitative analysis through the visualization of forecasts, which offers deeper insights into the model's behavior and the nature of its errors.

Quantitative metrics provide a standardized way to measure the discrepancy between the model's predicted values ($\widehat{y_i}$) and the actual observed values (y_i). They summarize the overall performance of the model into a single number, which is useful for comparing different models or hyperparameter configurations. Three of the most widely used metrics in time series forecasting are:

- **Mean Absolute Error (MAE)**: The MAE is the average of the absolute differences between the predicted and actual values. It measures the average magnitude of the errors in a set of predictions, without considering their direction. Because it is in the same units as the original data, it is straightforward to interpret. MAE is less sensitive to large, anomalous errors (outliers) than RMSE because it does not square the errors.[6] The formula is (equation 4.11):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \tag{4.11}$$

- **Root Mean Squared Error (RMSE)**: The RMSE is the square root of the average of the squared differences between predicted and actual values. By squaring the errors before they are averaged, RMSE gives disproportionately higher weight to large errors. This means that RMSE is particularly useful when large errors are especially undesirable. Like MAE, it is in the same units as the target variable, but it is always greater than or equal to MAE. The greater the difference between RMSE and MAE, the greater the variance in the individual errors in the sample.[6] The formula is (equation 4.12):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.12}$$

- **Mean Absolute Percentage Error (MAPE)**: The MAPE expresses the average error as a percentage of the actual values. This makes it a scale-independent metric, which is particularly useful for comparing the forecast performance of models across different time series with different scales. However, MAPE has significant limitations: it is undefined if any actual value is zero, and it can produce infinite or skewed results for values close to zero. It also has a bias, tending to favor models that under-predict over those that over-predict.[6] The formula is (equation 4.13):

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.13}$$

The selection of an evaluation metric is not merely a statistical choice; it is an implicit statement about the cost function of the real-world problem being solved. For instance, in forecasting electricity demand, under-predicting during a peak period could lead to a costly blackout, making a large positive error far more detrimental than a large negative error. In such a scenario, a metric like RMSE, which heavily penalizes large errors, or even a custom asymmetric loss function, would be more appropriate than MAE, which treats all errors linearly. The choice of metric should align the model's optimization objective with the practical consequences of its forecast errors.

4.3.13 Qualitative Analysis and Visualization of Forecasts

While quantitative metrics are essential for summarizing a model's overall performance, they can obscure important details about its behavior. A low average error could conceal significant systematic biases, such as a consistent failure to capture peaks, troughs, or turning points in the data. For this reason, qualitative analysis through visualization is an indispensable part of the evaluation process

The most common and effective visualization method is to create a line plot that displays the historical data, the actual future values from the test set, and the model's forecasted values on the same set of axes. This visual comparison allows for an immediate and intuitive assessment of the model's performance, revealing aspects that metrics alone cannot:

- How well the model captures the overall trend of the series.
- Whether the model successfully identifies and replicates seasonal patterns.
- The model's performance at critical points, such as sudden changes in direction or periods of high volatility.
- The presence of any systematic bias, such as consistent over- or under-prediction.

By combining rigorous quantitative metrics with insightful qualitative visualizations, a forecaster can develop a comprehensive and nuanced understanding of a model's strengths and weaknesses, leading to more reliable and trustworthy results.

4.3.14 The Advent of the Transformer: A Paradigm Shift

The field of sequence modeling underwent a significant paradigm shift with the publication of attention mechanism [80]. This seminal work introduced the Transformer architecture, a model that departed entirely from the recurrence-based mechanisms that had long dominated sequence transduction tasks, such as those performed by Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [80]. The paper's title was a deliberate and provocative hypothesis: that the attention mechanism, previously used as an enhancement to recurrent models, could on its own be sufficient for building powerful sequence models [80]. This proposition challenged the foundational assumption that sequential data necessitates sequential processing.

The primary motivation behind the Transformer was to overcome the inherent limitations of recurrence. Models like LSTMs process data sequentially, where the computation for a given time step t is dependent on the hidden state from time step t-1. This sequential dependency creates two fundamental bottlenecks. First, it precludes parallelization within a sequence, making the training of models on very long sequences computationally expensive and time-consuming [80]. Second, while LSTMs were specifically designed to mitigate the vanishing gradient problem and capture longer-term information, the path length for signals to travel between distant positions in a sequence remains long. This makes it challenging to effectively model long-range dependencies, a critical requirement for many complex tasks [81]. The Transformer was engineered to address these issues directly by eschewing recurrence and relying solely on an attention mechanism to model global dependencies between all input and output positions, thereby enabling a significant increase in parallelization and performance [80]. This architectural evolution represents a deeper, conceptual shift in how models handle context from a "memory-based" to an "access-based" paradigm. LSTMs operate by maintaining an internal, compressed memory, the cell state, which is updated at each time step to carry a summary of the relevant past information forward. This process is analogous to a human reading a document and continuously updating their mental summary. However, this compression is inherently lossy; specific details from the distant past can become diluted or lost within the evolving hidden state. The Transformer, in contrast, functions as an access-based system. At every processing step, its self-attention mechanism provides direct, unmediated access to the representation of every other element in the sequence. Rather than relying on a compressed memory, the model can dynamically query the entire input history and compute the precise relevance of any past data point, regardless of its distance. For time series forecasting, this distinction is profound. An LSTM might excel at tracking a slowly evolving trend by maintaining a memory of recent states, but a Transformer can instantly connect a current market anomaly to a similar, isolated event that occurred months prior, providing a more robust mechanism for capturing complex, non-local temporal patterns.

4.3.15 Deconstructing the Transformer Encoder

For sequence-to-one prediction tasks, such as time series forecasting, the encoder component of the original is of primary relevance. The encoder's fundamental role is to process an input sequence of representations, $x = (x_1, \ldots, x_n)$, and map it to a sequence of continuous, context-aware representations, $z = (z_1, \ldots, z_n)$ [80]. Each output vector z_i is not just a representation of the input x_i but is enriched with contextual information from the entire input sequence.

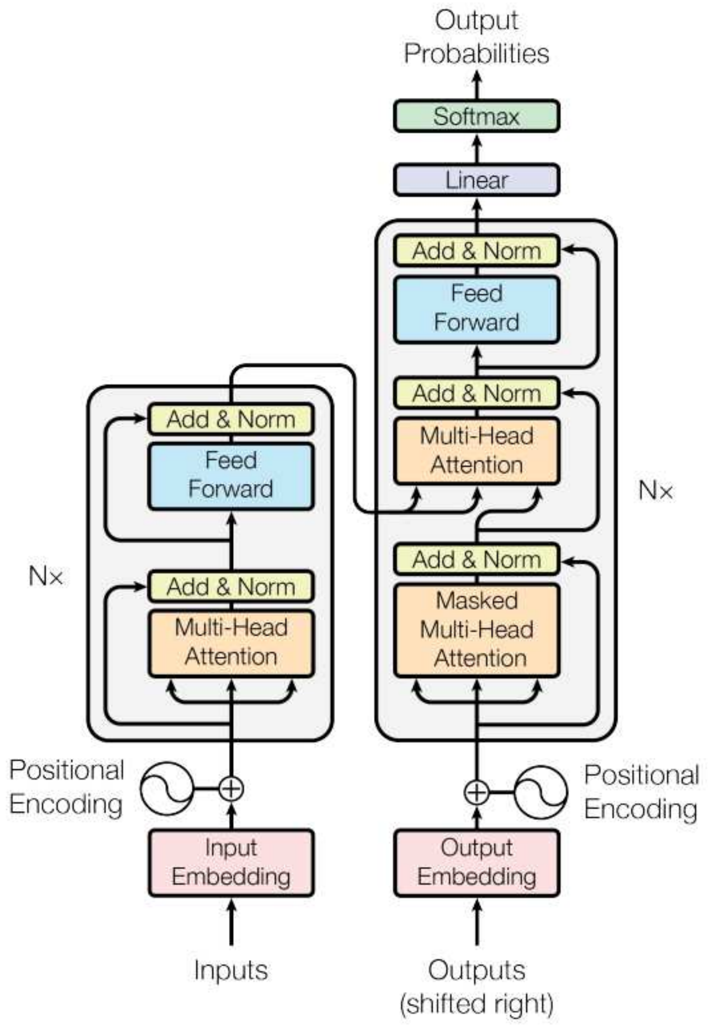


Figure 4.4: The Transformer Model Architecture. The diagram illustrates the stacked encoder and decoder layers, each containing multi-head attention and feed-forward sub-layers [80]

The core innovation of the Transformer is the self-attention mechanism, sometimes referred to as intra-attention [80]. This mechanism allows the model to weigh the importance of all other positions in the input sequence when computing the representation for a given position [80], [80]. It achieves this by relating every position to every other position within a single sequence, thereby generating a deeply contextualized output.

The mechanism is conceptually framed using the analogy of a retrieval system with three components: a Query (Q), a set of Keys (K), and a set of Values (V). For each element in the input sequence, three vectors are created through learned linear projections of its embedding: a query vector, a key vector, and a value vector [80]. To compute the output for a specific element, its query vector is compared against the key vectors of all elements in the sequence (including itself). This comparison yields a set of attention scores, or weights. These weights are then used to compute a weighted sum of all the value vectors in the sequence, producing an output that is a synthesis of information from the entire sequence, focused on the parts most relevant to the current element [80].

The specific implementation used in the Transformer is Scaled Dot-Product Attention. The attention scores are computed by taking the dot product of the query with each key. To counteract the effect of large dot products that could saturate the softmax function and result in vanishing gradients, the scores are scaled by the square root of the dimension of the key vectors, d_k [80]. The complete mathematical formulation for a set of queries, keys, and values packed into matrices Q, K, and V is (equation 4.14):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.14}$$

In this equation, the QK^T term calculates the similarity scores between every query and every key. The division by $\sqrt{d_k}$ provides the crucial scaling. The softmax function then normalizes these scores into a probability distribution, ensuring the weights sum to one. Finally, this matrix of weights is multiplied by the value matrix V to produce the final output representation [80]. Rather than relying on a single attention function, the Transformer architecture enhances this mechanism through Multi-Head Attention. The rationale is that a single attention mechanism might be constrained to focus on only one type of relationship or feature. By using multiple attention "heads" in parallel, the model can jointly attend to information from different representation subspaces at different positions [80]. For instance, in a time series, one head might learn to capture short-term momentum, while another focuses on weekly seasonality, and a third identifies long-term trends [80].

Architecturally, Multi-Head Attention operates by first applying h different learned linear projections to the input queries, keys, and values, creating h distinct sets of Q, K, and V matrices [80]. The Scaled Dot-Product Attention mechanism is then applied to each of these h sets in parallel, producing h separate output vectors [80]. These outputs are concatenated and passed through a final learned linear projection to produce the final output of the layer. This structure allows each head to specialize and capture different aspects of the relationships within the sequence [80].

The mathematical formulation is as follows (equation 4.15):

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \tag{4.15}$$

where each head is computed as (equation 4.16):

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{4.16}$$

The projection matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and the final output projection matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are all learned parameters. In the original paper, $h = 8$ parallel heads were used [80].

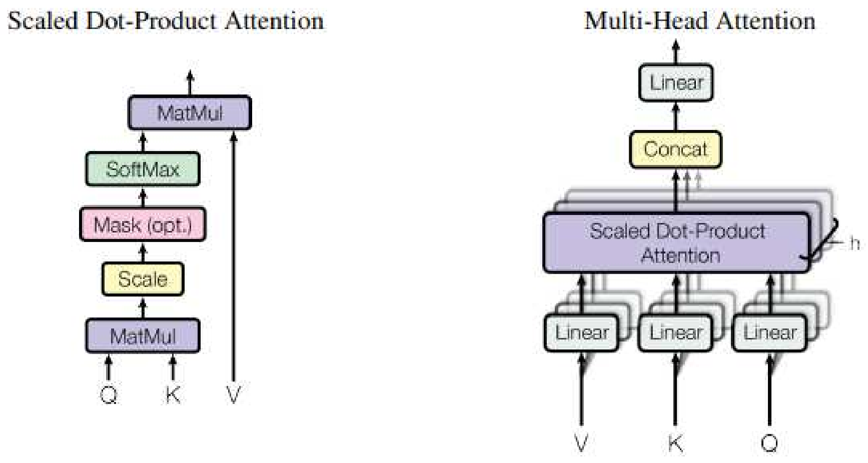


Figure 4.5: Scaled Dot-Product and Multi-Head Attention Mechanisms. Illustrating the parallel nature of the multi-head attention mechanism and the internal computation of the scaled dot-product attention [80].

A critical consequence of removing recurrence and convolution is that the self-attention mechanism, by itself, is permutation-invariant; it has no inherent understanding of the order or position of the elements in a sequence [80]. For tasks involving time series data, where the temporal order is fundamental, this is a significant limitation.

To address this, the Transformer architecture injects information about the position of each element directly into its representation. This is achieved by adding a "positional encoding" vector to each input embedding at the bottom of the encoder stack [80]. These vectors are not learned but are calculated using a fixed function, ensuring the model has a consistent signal for position. Vaswani et al. (2017) proposed using sine and cosine functions of varying frequencies for this purpose [80]. The formulation for the encoding at position pos and dimension i is (equation 4.17):

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{4.17}$$

Here, d_{model} is the dimension of the embeddings. This choice of function is advantageous because for any fixed offset k, $PE_{(pos+k)}$ can be represented as a linear function of $PE_{(pos)}$, which may allow the model to more easily learn to attend to relative positions. This specific formulation has been adopted in subsequent hybrid models, including the one proposed by Cao et al. (2024) [82].

A single layer within the Transformer encoder is composed of two primary sub-layers: the multi-head self-attention mechanism described above, and a simple, position-wise fully connected feed-forward network (FFN) [80]. The FFN is applied independently and identically to each position.

To facilitate the training of a deep network, a residual connection is employed around each of the two sub-layers, followed by layer normalization [80]. The output of each sub-layer can therefore be expressed as LayerNorm$(x + \text{Sublayer}(x))$, where Sublayer(x) is the function implemented by the sub-layer itself (either multi-head attention or the FFN). These residual connections and normalization steps are critical for stabilizing the training process and allowing gradients to flow through the deep architecture. The complete encoder is formed by stacking N of these identical layers (the original paper used $N = 6$) [80].

4.3.16 Synergizing Recurrence and Attention: The Rationale for Hybrid Models

The distinct characteristics of LSTMs and Transformers naturally motivate the development of hybrid architectures that seek to combine their complementary strengths [81], [82], [83]. LSTMs, with their inherent recurrence and gating mechanisms, are highly proficient at modeling the sequential flow of information and capturing local temporal dependencies. They are well-suited for tasks where an evolving state must be tracked over time. Their primary limitations, however, are their sequential computational nature, which hinders scalability, and their difficulty in capturing very long-range dependencies effectively [80], [81]. Conversely, Transformers excel at capturing global context and long-range dependencies via the self-attention mechanism, which creates direct pathways between all elements in a sequence. They are highly parallelizable, making them efficient for processing long sequences [80]. However, they lack the inductive bias for sequential inheritance in recurrent models and may be less adept at modeling fine-grained local patterns without sophisticated positional encodings [81].

The central hypothesis of hybrid models is that by integrating these two architectures, it is possible to create a system that leverages the strengths of both while mitigating their respective weaknesses [81], [82], [84]. In such a system, the LSTM can serve as a specialized feature extractor for local temporal dynamics, while the Transformer can model the global, long-range relationships between these extracted features.

This approach can be understood as a form of hierarchical feature extraction specifically tailored for time series data. In the common `LSTM -> Transformer` configuration, the model operates in two distinct stages. At a "low level," the LSTM processes the raw, sequential time series data. Its recurrent nature allows it to effectively capture local trends, momentum, and short-term patterns, producing a sequence of hidden states. Each hidden state is a rich vector summarizing the relevant history up to that point. This initial step transforms the problem from finding patterns in raw data points to a higher-level problem of finding patterns among these more abstract, context-rich summary vectors. At this "high level," the Transformer takes the sequence of LSTM hidden states as its input. It then applies the self-attention mechanism to analyze the interdependencies among these feature vectors, identifying globally significant patterns, seasonalities, or long-range correlations that might be too distant for the LSTM to capture alone. This hierarchical process is analogous to the functioning of Convolutional Neural Networks (CNNs) in computer vision, where initial layers detect simple features like edges and textures, and deeper layers combine them to recognize complex objects. By having the Transformer operate on the smoothed, feature-rich output of the LSTM rather than on noisy raw data, this architecture likely improves model stability and performance, which helps explain its empirical success across diverse domains like engineering and finance [82], [85].

4.3.17 A Review of Transformer-LSTM Architectures for Time Series Prediction

The theoretical advantages of combining recurrent and attention-based models have led to the development of several hybrid architectures for time series prediction. These models often follow the hierarchical pattern of using an LSTM for local feature extraction followed by a

Transformer for global context modeling, though variations exist depending on the specific application.

4.3.18 The LSTM-Transformer Hybrid Model (Cao et al., 2024)

A prominent example of this hybrid approach is presented in by Cao et al. (2024) [82]. The model is designed for applications such as underground drilling and green stormwater management, which require high predictive accuracy, real-time adaptability, and computational efficiency [82].

The architecture follows the `LSTM -> Transformer` structure. The foundation of the model is a multi-layered LSTM component, described as the "bastion of sequential data comprehension," which first processes the input time series to capture its intricate temporal dynamics [82]. The sequence of hidden state outputs from the LSTM is then passed to the Transformer component, which the authors term the "maestro of contextual comprehension." This component uses multiple self-attention heads to analyze the LSTM-generated feature sequence, focusing on different facets to build a holistic understanding of the global context [82]. To ensure the Transformer is aware of the temporal order of the features extracted by the LSTM, positional encodings are added to the LSTM's output sequence before it enters the Transformer layer [82]. Beyond this core hybrid structure, the authors incorporate, adaptive learning, online learning mechanisms for dynamic adaptation to new data and knowledge distillation techniques to transfer insights from larger networks, enhancing both performance and efficiency for real-world deployment [82].

In the context of the proposed hybrid model, the inclusion of an adaptive learning rate mechanism is critical for ensuring the model's stability and efficiency, particularly within the online learning framework. Engineering systems are rarely static; they are dynamic environments where data distributions can shift over time due to changing operational conditions, equipment wear, or external factors [82]. A fixed learning rate would be ill-suited for such a fluid setting. A rate that is too high could cause the model to become unstable and fail to converge when encountering new data patterns, while a rate that is too low would make the learning process too slow, causing the model to lag behind the evolving dynamics of the system [82].

Therefore, an adaptive approach is essential for the model to remain robust and responsive. The framework described in the paper employs a multi-faceted strategy to dynamically adjust the learning rate, ensuring both rapid adaptation and stable convergence [82].

Key Implementation Strategies for Adaptive Learning

The model incorporates three primary techniques to manage the learning rate dynamically [82]:

Adam Optimizer: This optimizer is central to the adaptive strategy. Instead of using a single learning rate for all model parameters, Adam (Adaptive Moment Estimation) calculates an individual learning rate for each parameter. It does this by maintaining a moving average of both the past gradients (the first moment) and the squared gradients (the second moment). This allows the optimizer to make larger updates for infrequent parameters and smaller, more cautious updates for frequent ones, leading to a learning process that is both swift and stable [82]. The update rule is given by (equation 4.18, 4.19, 4.20) :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.18}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.19}$$

$$\theta_{t+1} = \theta_t - \frac{\eta m_t}{\sqrt{v_t} + \epsilon} \tag{4.20}$$

Learning Rate Annealing: This technique complements the Adam optimizer by gradually reducing the overall learning rate as the training process progresses. The model starts with a relatively high learning rate to make significant progress in the initial stages of learning. As the model begins to converge on a solution, the learning rate is systematically decreased. This "annealing" process allows for more refined, fine-grained adjustments in the later stages, helping the model settle into a more precise and optimal minimum without overshooting it [82]. The annealed learning rate (η_t) can be calculated as (equation 4.21):

$$\eta_t = \eta_0 \times \frac{1}{1 + \delta t} \tag{4.21}$$

(where η_0 is the initial rate, δ is a decay parameter, and t is the time step).

Gradient Clipping: This is a crucial stabilization technique, especially in online learning scenarios where new batches of data could potentially cause gradients to become excessively large (a problem known as "exploding gradients"). Such large gradients can lead to drastic and unstable updates to the model's weights, potentially erasing past learning. Gradient clipping mitigates this risk by setting a predefined threshold. If the norm of the gradients exceeds this threshold, they are scaled down to match it, thus preventing overly aggressive updates and ensuring the training process remains stable and controlled [12]. The clipped gradient (g') is determined by (equation 4.22):

$$g' = \delta \times \frac{g}{\|g\|} \tag{4.22}$$

(if $\| g \| > \delta$, where δ is the threshold).

By combining these three mechanisms, the model is equipped with a sophisticated and robust system for managing its learning rate. This ensures it can effectively adapt to the continuous stream of new data in dynamic engineering environments, learning new patterns quickly without becoming unstable or forgetting past knowledge. This adaptability is a cornerstone of what makes the proposed architecture well-suited for real-world, real-time applications.

4.3.19 Other Notable Hybrid Architectures

The `LSTM -> Transformer` pattern is not unique to engineering applications and has been successfully adapted for other domains, demonstrating its versatility. For instance, was proposed for financial time series forecasting [85]. This model also employs an LSTM network to first capture sequential patterns. The resulting feature vector is then processed by a *modified* Transformer (`mTrans`) that utilizes self-attention, and a final Multilayer Perceptron (MLP) generates the prediction [85]. This work is particularly insightful as it demonstrates a key trend: the adaptation of the original Transformer architecture for non-NLP tasks. The authors explicitly remove the input embedding module (unnecessary for numerical time series) and replace the full Transformer decoder with a simpler MLP, streamlining the architecture for the specific sequence-to-one forecasting task [85].

While the `LSTM -> Transformer` structure is common, alternative configurations have also been explored. A model designed for predicting NH3-N concentration in water treatment facilities utilizes a hybrid model [86]. In this design (figure 4.6), a Transformer *encoder* first processes the entire input time series simultaneously to generate a global, contextualized representation. This static representation then serves as the initial state or input for an LSTM *decoder*, which sequentially generates the output prediction. This approach embodies a different modeling philosophy: first, gain a comprehensive global understanding of the entire input sequence, and then use that context to guide a sequential generation process. This structure may be particularly well-suited for sequence-to-sequence tasks but provides an important architectural contrast.

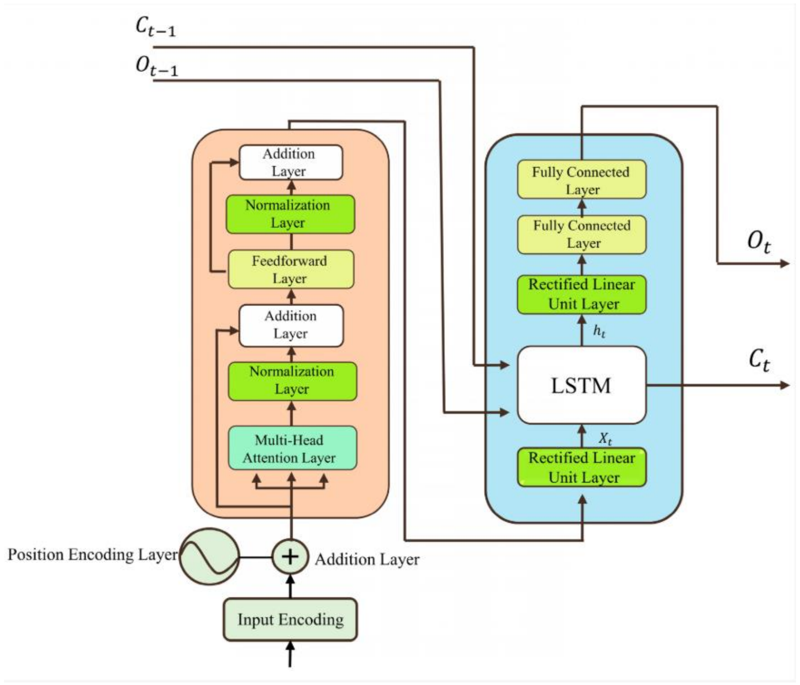


Figure 4.6: A seq-2-seq architecture with no embeddings and using transformers encoder as feature extractor

Furthermore, the general principle of combining a local feature extractor with a global context modeler extends beyond LSTMs. Other hybrid models that pair have also been proposed for time series forecasting [81]. In these architectures, CNNs are used to efficiently extract local, short-term patterns and motifs from the time series, and the output of the CNN layers is then fed into a Transformer to model the long-range dependencies between these extracted features. This reinforces the broader architectural paradigm of a two-stage, local-to-global feature extraction process.

The following table provides a concise comparison of these representative hybrid architectures.

Model Name/Reference	Core Architecture	Key Innovations/Features	Primary Application Domain
Cao et al. (2024) [82]	LSTM -> Transformer	Online Learning, Knowledge Distillation	Engineering Systems
Kabir et al. (2025) [85]	LSTM -> mTrans -> MLP	Modified Transformer (no decoder/embedding), MLP head	Financial Forecasting
Zhang et al. (2024) [81]	Transformer -> LSTM	Encoder-Decoder structure with LSTM as decoder	Water Treatment (Seq2Seq)
Zhang et al. (2025) [81]	CNN -> Transformer	Dynamic CNN kernel, Multi-scale attention	Financial Forecasting

Table 4.3: Comparison of Hybrid Transformer-Based Architectures for Time Series Forecasting

Chapter 5 - Methodology & Results for Gas Phase system

5.1 Data Pre-processing and Preparation

The raw trajectory data from Car-Parrinello molecular dynamics (CPMD) simulations requires systematic preprocessing to create a suitable dataset for deep learning model training. The preprocessing pipeline transforms the continuous molecular coordinate trajectories into structured sequential data compatible with the hybrid Transformer-LSTM architecture.

5.1.1 Dataset Structure and Characteristics

This chapter considers the dataset of pair C_1 and H_2 out of considered pairs C_1 and H_1, C_1 and H_2, C_2 and H_1, C_2 and H_2 for model configuration. Rest of the pairs follow the same workflow with only difference in the hyperparameters selection. The dataset consists of molecular trajectory data spanning 20,000 temporal steps, where each timestep represents a discrete moment in the CPMD simulation. The data structure comprises seven distinct features arranged as follows: the first three columns contain the x, y, and z coordinates of the first atom involved in the collective variable calculation, the subsequent three columns contain the corresponding spatial coordinates of the second atom, and the final column contains a binary flag indicating the temporal points at which Gaussian potentials were introduced during the metadynamics simulation process.

This binary flag represents a predetermined aspect of the CPMD simulation protocol rather than an emergent system property, as the timing of Gaussian potential drops follows the metadynamics algorithm's systematic exploration strategy. The spatial coordinates constitute the dynamic variables that the model must learn to predict, while the binary flag serves as additional contextual information that influences the system's behavior but remains deterministic based on the simulation protocol.

5.1.2 Temporal Partitioning Strategy

The temporal partitioning approach divides the dataset into training and inference phases while incorporating a validation overlap region to assess model performance across different temporal scales. The first 12,000 timesteps constitute the training dataset, providing the model with sufficient temporal context to learn the underlying dynamical patterns of the molecular system. The remaining 8,000 timesteps are reserved for inference evaluation, creating a substantial out-of-sample testing period that spans 40% of the total simulation time.

A critical aspect of this partitioning strategy involves the implementation of an overlapping validation region. During inference, prediction commences at timestep 8,000 rather than immediately following the training period at timestep 12,001. This creates a 4,000-timestep overlap where ground truth data exists for both training context and validation purposes. This overlap serves two essential functions: it provides the model with adequate initial context from the training period to establish stable predictions, and it enables comprehensive performance evaluation across the transition from training to purely predictive phases.

5.1.3 Sequential Data Transformation

The transformation of trajectory data into sequential learning samples employs a sliding window methodology specifically adapted for the temporal characteristics of molecular dynamics. The approach creates input-output pairs where each input consists of a sequence of 20 consecutive timesteps containing all seven features, paired with a target output containing only the six coordinate features from the subsequent timestep.

This windowing strategy embodies several important design decisions. The sequence length of 20 timesteps represents a balance between capturing sufficient temporal context for meaningful pattern recognition while maintaining computational tractability during training. This temporal window corresponds to the time scale over which local molecular motions maintain predictable correlations, allowing the model to learn both short-term momentum effects and intermediate-range dynamical patterns.

The target prediction focuses exclusively on the six spatial coordinates, deliberately excluding the binary flag from the prediction task. This design choice reflects the physical understanding that the binary flag represents external simulation controls rather than intrinsic system dynamics. During inference, the model receives the predetermined binary flag values as input context but does not attempt to predict them, maintaining the distinction between controllable simulation parameters and emergent molecular behavior.

5.2 Model Architecture Framework

5.2.1 Hybrid Transformer-LSTM Design Philosophy

The model architecture implements a hierarchical feature extraction strategy that combines the complementary strengths of Transformer and LSTM architectures. This hybrid approach addresses the specific challenges inherent in molecular dynamics prediction, where both local temporal dependencies and long-range correlation patterns influence system evolution.

The design philosophy centers on a two-stage feature extraction process. The initial stage employs a Transformer encoder to process the entire input sequence simultaneously, leveraging self-attention mechanisms to identify complex interdependencies between different temporal positions within the 20-timestep input window. This stage excels at capturing non-local temporal relationships and global context patterns that might span the entire input sequence.

The second stage utilizes LSTM layers to process the Transformer-generated features sequentially, applying recurrent mechanisms to model the temporal flow and evolution of the extracted high-level features. This stage specializes in maintaining temporal coherence and modeling the sequential nature of dynamical evolution that is fundamental to physical systems.

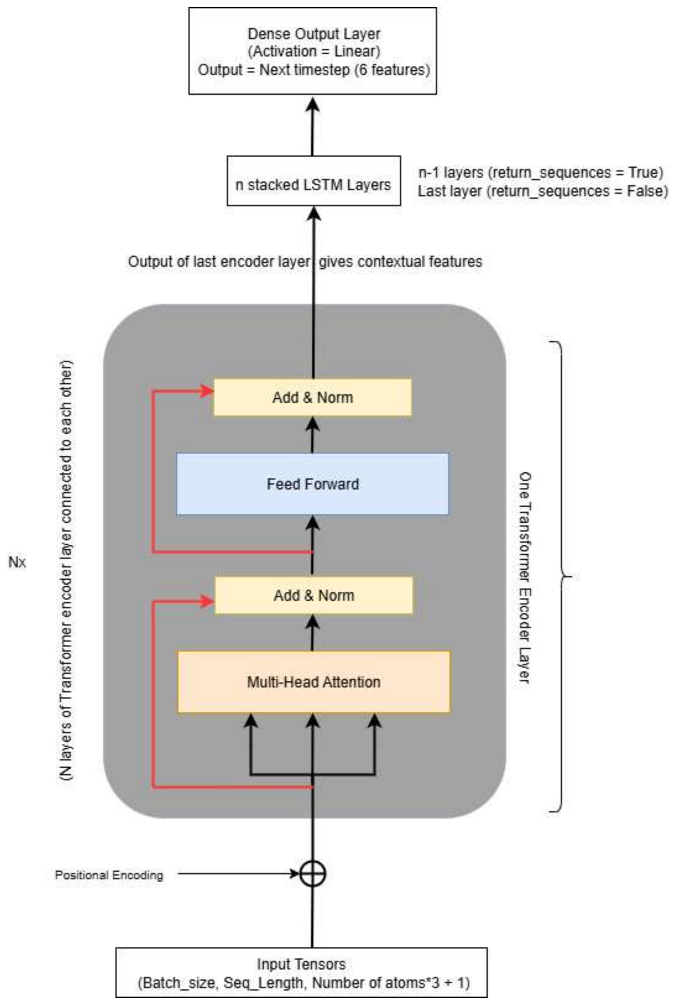

Dense Output Layer
(Activation = Linear)
Output = Next timestep (6 features)
n stacked LSTM Layers
n-1 layers (return_sequences = True)
Last layer (return_sequences = False)
Output of last encoder layer gives contextual features
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
(N layers of Transformer encoder layer connected to each other)
One Transformer Encoder Layer
Positional Encoding
Input Tensors
(Batch_size, Seq_Length, Number of atoms*3 + 1)

Figure 5.1: A seq-2-one architecture using transformers as feature extractor with no embeddings

5.2.2 Transformer Encoder Implementation

The Transformer encoder component processes the input sequence through multi-head self-attention mechanisms that enable each timestep to directly access information from all other timesteps within the input window. The encoder architecture incorporates positional encoding to provide explicit temporal order information, ensuring that the attention mechanism can distinguish between different temporal positions within the sequence.

The multi-head attention mechanism employs 2 attention heads, each with a head size of 128 dimensions. This configuration allows the model to simultaneously focus on different types of temporal relationships within the molecular trajectory data. The feed-forward dimension of 64 provides sufficient representational capacity for processing the attention-weighted features while maintaining computational efficiency. The Transformer encoder consists of 2 layers, creating a depth that enables hierarchical feature abstraction while avoiding the computational overhead associated with deeper architectures.

5.2.3 LSTM Processing Layers

Following the Transformer encoder, the architecture incorporates three LSTM layers that process the attention-enhanced features sequentially. Each LSTM layer contains 32 hidden units, providing adequate memory capacity for maintaining temporal context while preventing excessive parameter proliferation.

The three-layer LSTM configuration creates a hierarchical temporal processing structure where each layer can specialize in different aspects of sequential modeling. The first LSTM layer receives the globally-contextualized features from the Transformer and begins the process of temporal integration. The subsequent layers refine this temporal representation, with each layer capable of capturing increasingly complex temporal dependencies.

The final LSTM layer produces a single output vector that encapsulates the complete temporal and contextual information from the input sequence. This vector serves as the foundation for the coordinate prediction, containing all the learned dynamical patterns necessary for forecasting the next timestep's molecular configuration.

5.2.4 Output Layer and Prediction Generation

The architecture concludes with a dense output layer that maps the final LSTM hidden state to the six-dimensional coordinate prediction space. This layer employs linear activation, allowing the model to generate continuous-valued coordinate predictions without artificial constraints on the output range.

The output dimensionality of six corresponds exactly to the spatial coordinates of the two atoms under consideration (three coordinates each), maintaining direct correspondence with the physical variables of interest. The model deliberately excludes prediction of the binary flag, instead receiving this information as input context during inference.

5.3 Hyperparameter Configuration and Training Strategy

5.3.1 Core Model Hyperparameters

The model configuration employs carefully selected hyperparameters that balance representational capacity with computational efficiency and training stability. The Transformer encoder utilizes a head size of 128 dimensions with 2 attention heads, creating a total attention dimension of 256. The feed-forward network dimension of 64 provides sufficient non-linear transformation capacity while maintaining parameter efficiency.

The LSTM components employ 32 hidden units per layer across three sequential layers, resulting in a total LSTM parameter count that scales appropriately with the input complexity. The sequence length of 20 timesteps provides adequate temporal context for learning molecular dynamics patterns while remaining computationally manageable during both training and inference phases.

5.3.2 Training Configuration

The training process employs the Adam optimizer with an initial learning rate of 0.001, providing adaptive learning rate scheduling that adjusts parameter updates based on gradient statistics. The loss function utilizes Mean Absolute Percentage Error (MAPE), which provides scale-invariant error measurement particularly suitable for coordinate prediction tasks where different spatial dimensions might have varying magnitudes.

The training dataset employs a batch size of 32, creating mini-batches that balance gradient estimation accuracy with computational efficiency. Training proceeds for a maximum of 2000 epochs, though early stopping mechanisms typically terminate training when convergence criteria are satisfied.

5.3.3 Regularization and Overfitting Prevention

The model incorporates multiple regularization strategies to prevent overfitting and enhance generalization performance. Dropout regularization with a rate of 0.1 is applied throughout the architecture, randomly deactivating neurons during training L2 regularization with a coefficient of 0.0002 is applied to all weight matrices, encouraging smaller parameter magnitudes and smoother decision boundaries. This regularization strength provides sufficient constraint on parameter growth without overly restricting the model's representational capacity.

5.3.4 Advanced Training Callbacks

The training process employs sophisticated callback mechanisms to optimize convergence and prevent degradation of performance. Early stopping monitoring tracks validation loss with a patience of 75 epochs and a minimum improvement threshold of 1e-6, ensuring that training terminates when meaningful improvement ceases.

Learning rate reduction employs a patience of 25 epochs with a reduction factor of 0.7, systematically decreasing the learning rate when training plateaus to enable fine-grained optimization. The minimum learning rate is set to 1e-7, preventing the optimizer from becoming overly conservative in parameter updates.

A custom overfitting monitor tracks the relationship between training and validation performance, providing early warning when the model begins to memorize training data rather than learning generalizable patterns. This monitor employs a threshold of 0.015 for detecting significant divergence between training and validation metrics.

5.4 Attention Mechanism Strength for MD Data

The attention mechanism analysis provides compelling evidence for its effectiveness in molecular dynamics prediction through three key findings.

Multi-head attention demonstrates clear functional specialization. Figure 5.2 reveals that each attention head has learned distinct temporal strategies: Heads 1 and 2 exhibit recency bias, focusing computational resources on recent molecular states most relevant for immediate prediction; Head 3 maintains uniform attention across the sequence, providing global contextual awareness; Heads 0 and 4 display periodic attention patterns with regular intervals, suggesting they capture oscillatory molecular phenomena such as vibrational modes or rotational dynamics; Head 5 shows mixed local-distant attention patterns, potentially capturing intermediate-range molecular interactions. This specialization demonstrates that the multi-head architecture successfully partitions the temporal modeling task across complementary functional units.

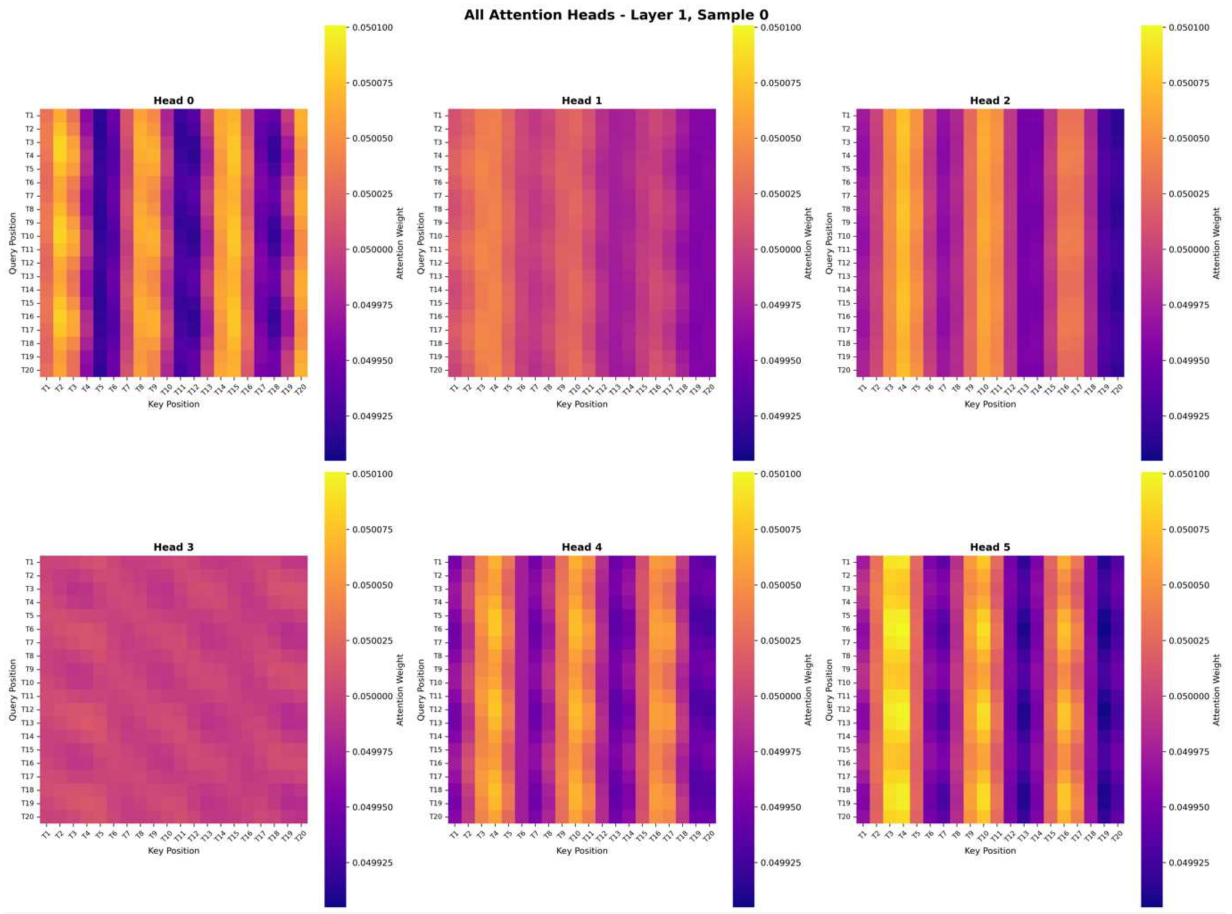


Figure 5.2: Multi-Head Attention Specialization in Transformer Layer 1

The emergence of physically meaningful patterns validates domain relevance. Figure 5.3's detailed analysis of Head 0 reveals structured vertical bands with regular periodicity rather than random noise. These patterns likely correspond to characteristic molecular frequencies, a fundamental aspect of molecular dynamics where systems exhibit regular oscillatory behavior. The attention mechanism has spontaneously learned to recognize these temporal periodicities without explicit programming, indicating it captures genuine physical phenomena rather than spurious statistical correlations.

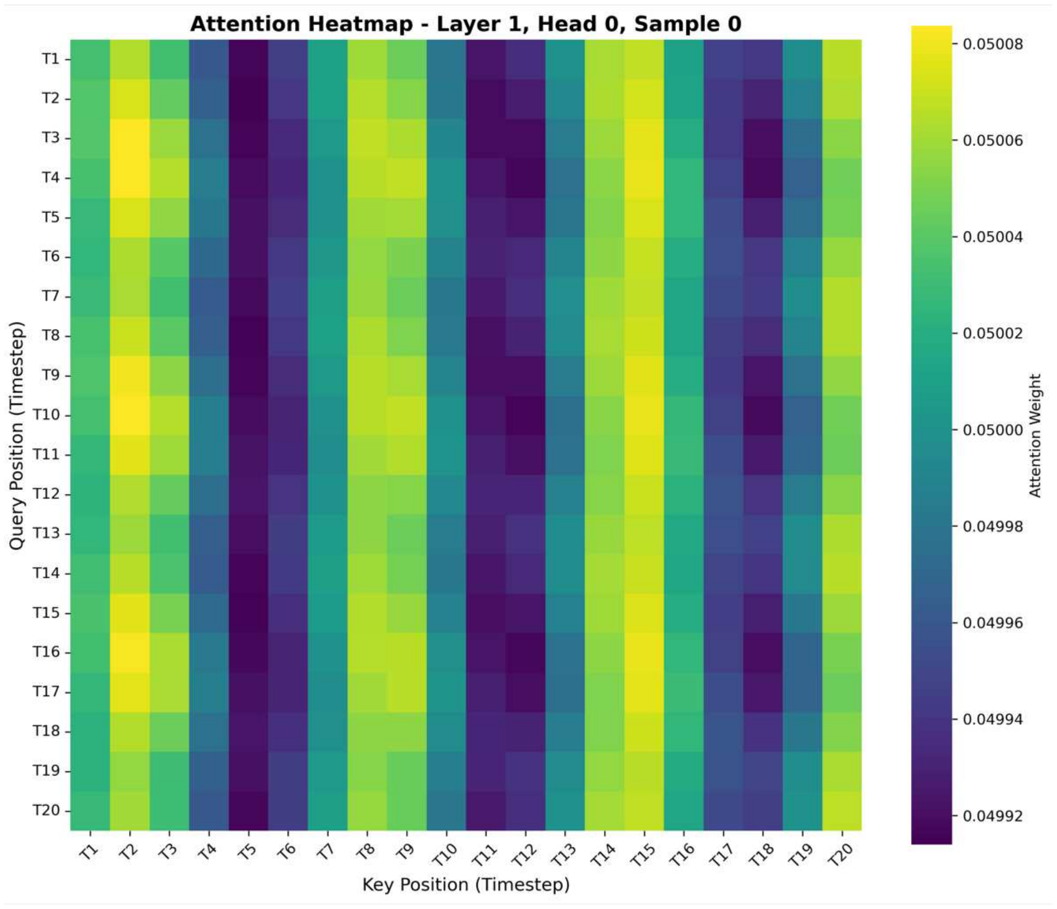


Figure 5.3: Temporal Attention Pattern Analysis Showing Learned Periodic Dependencies

Quantitative attention analysis confirms non-trivial learning. While attention weights cluster around the uniform baseline of 0.05 (1/20 timesteps), the observed variation range of 0.04992 to 0.05008 represents approximately 20% deviation from uniformity, a substantial range for softmax-normalized attention distributions. This deviation, combined with the structured rather than random nature of the patterns, provides statistical evidence that the model has learned meaningful temporal dependencies.

These findings establish that the attention mechanism provides sophisticated temporal modeling capabilities that extend beyond simple recurrent processing, enabling the transformer to intelligently allocate computational focus based on learned molecular dynamics principles while maintaining specialized heads for different aspects of the prediction task.

5.5 Inference Methodology and Multi-Horizon Evaluation

5.5.1 Autoregressive Prediction Framework

The inference process implements an autoregressive prediction strategy where the model generates sequences of future predictions by iteratively using its own outputs as inputs for subsequent timesteps. This approach mirrors the fundamental challenge of real-world forecasting, where future ground truth values are unavailable and the model must rely entirely on its learned representations and previous predictions.

The inference begins at timestep 8,000, utilizing the final 20 timesteps from the training period (timesteps 7,980 through 7,999) as the initial input sequence. This initialization provides the model with high-quality contextual information derived from ground truth data, establishing a stable foundation for the subsequent predictive sequence.

5.5.2 Multi-Horizon Prediction Strategy

The evaluation framework implements three distinct prediction horizons (1, 10, 50, and 100 timesteps) to assess model performance across different temporal scales. In the context of time series forecasting, the prediction horizon (H) denotes the number of future timesteps for which forecasts are generated, thereby defining the temporal distance between the last observed data point and the point being predicted. Each horizon represents a different balance between ground truth context and autoregressive prediction, providing insights into how prediction accuracy degrades with increasing temporal distance from the training data.

For horizon-1 prediction, the model continuously incorporates ground truth coordinates when available, using its own predictions only when ground truth data is unavailable. This represents an idealized scenario where perfect state information is continuously available.
For multi-step horizons (10, 50, and 100), the model employs a periodic reset strategy. Every N timesteps (where N equals the horizon length), the model's input sequence is reset using ground truth data from the corresponding temporal position. This approach balances the need for long-term autonomous prediction with the practical consideration that prediction errors accumulate over time.

Between reset points, the model operates in pure autoregressive mode, where each prediction becomes part of the input sequence for the next timestep. This creates a challenging evaluation scenario that closely approximates real-world forecasting conditions where ground truth future data is unavailable.

5.5.3 Binary Flag Integration During Inference

During inference, the model receives the predetermined binary flag values as part of its input context, maintaining consistency with the training data structure. Since these flags represent controlled aspects of the CPMD simulation protocol rather than emergent system properties, their values are known a priori for all future timesteps.

The integration of binary flags during autoregressive prediction creates a complete seven-dimensional input vector at each timestep, where the six coordinate values come from either ground truth data (during reset periods) or previous model predictions (during autoregressive periods), while the binary flag always derives from the predetermined simulation protocol.

5.5.4 Performance Monitoring and Error Accumulation Analysis

The inference process implements comprehensive performance monitoring that tracks prediction accuracy as a function of temporal distance from the training data. Mean Absolute Percentage Error (MAPE) serves as the primary evaluation metric, providing scale-invariant assessment of coordinate prediction accuracy.

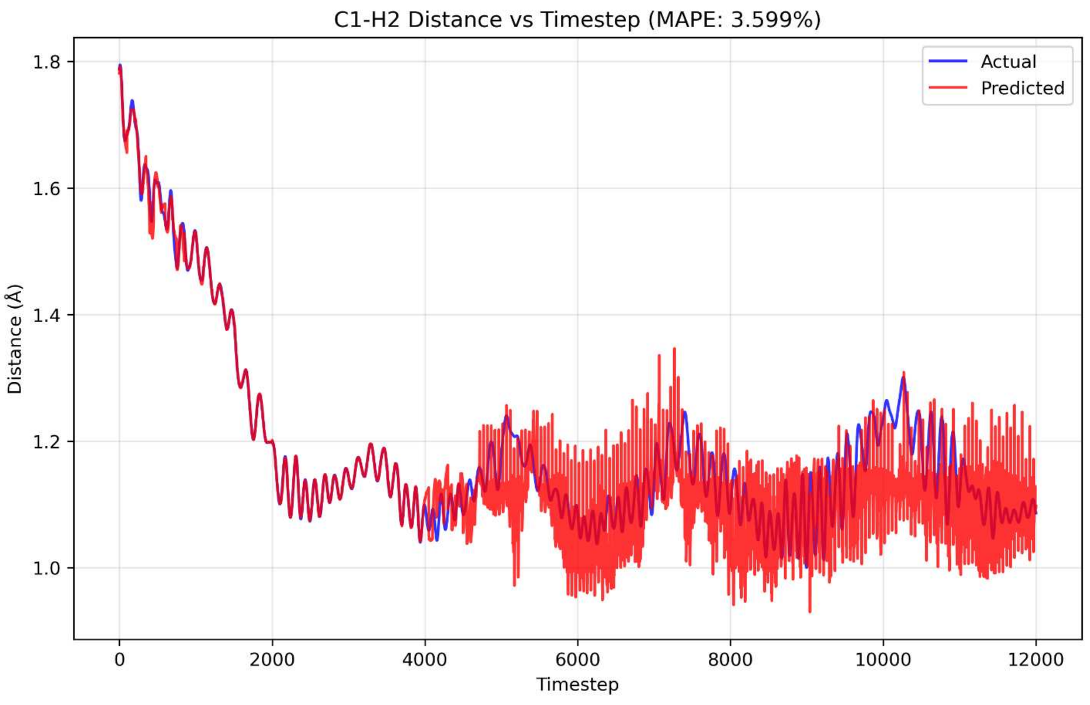
Error accumulation analysis examines how prediction accuracy degrades during extended autoregressive sequences, providing insights into the model's ability to maintain stable long-term predictions without ground truth correction. This analysis is crucial for understanding the practical limitations of the model and determining appropriate application scenarios.

The evaluation framework generates detailed performance statistics for each prediction horizon, including average error rates, error distribution characteristics, and temporal patterns in prediction accuracy. These statistics provide comprehensive assessment of model performance across the diverse temporal scales relevant to molecular dynamics applications.

5.6 Results

The hybrid Transformer-LSTM model was evaluated across four distinct atomic pair interactions (C_1-H_1, C_1-H_2, C_2-H_1, C_2-H_2) to assess its predictive performance in molecular dynamics trajectory forecasting. The model demonstrated varying degrees of accuracy across different molecular interactions, with performance metrics indicating the complexity and challenges inherent in long-term autoregressive prediction of molecular systems. The results are discussed for H = 50.

5.6.1 Model Performance Across Atomic Pairs



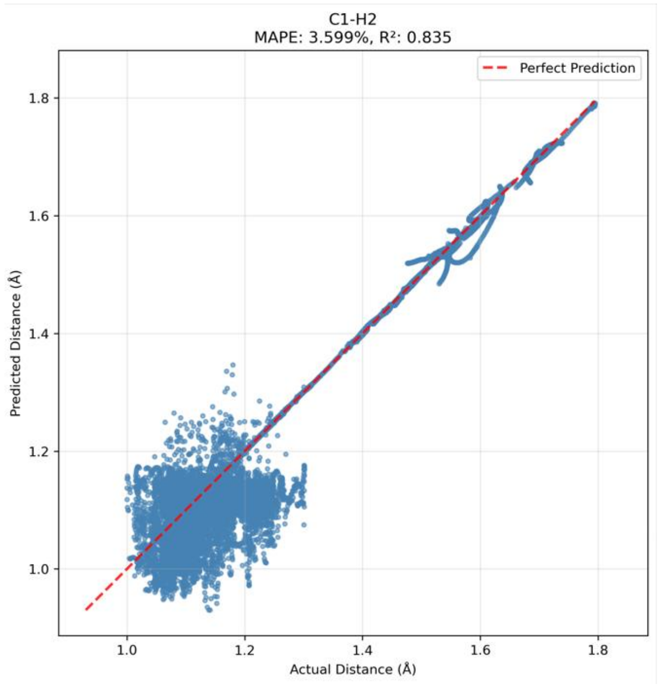
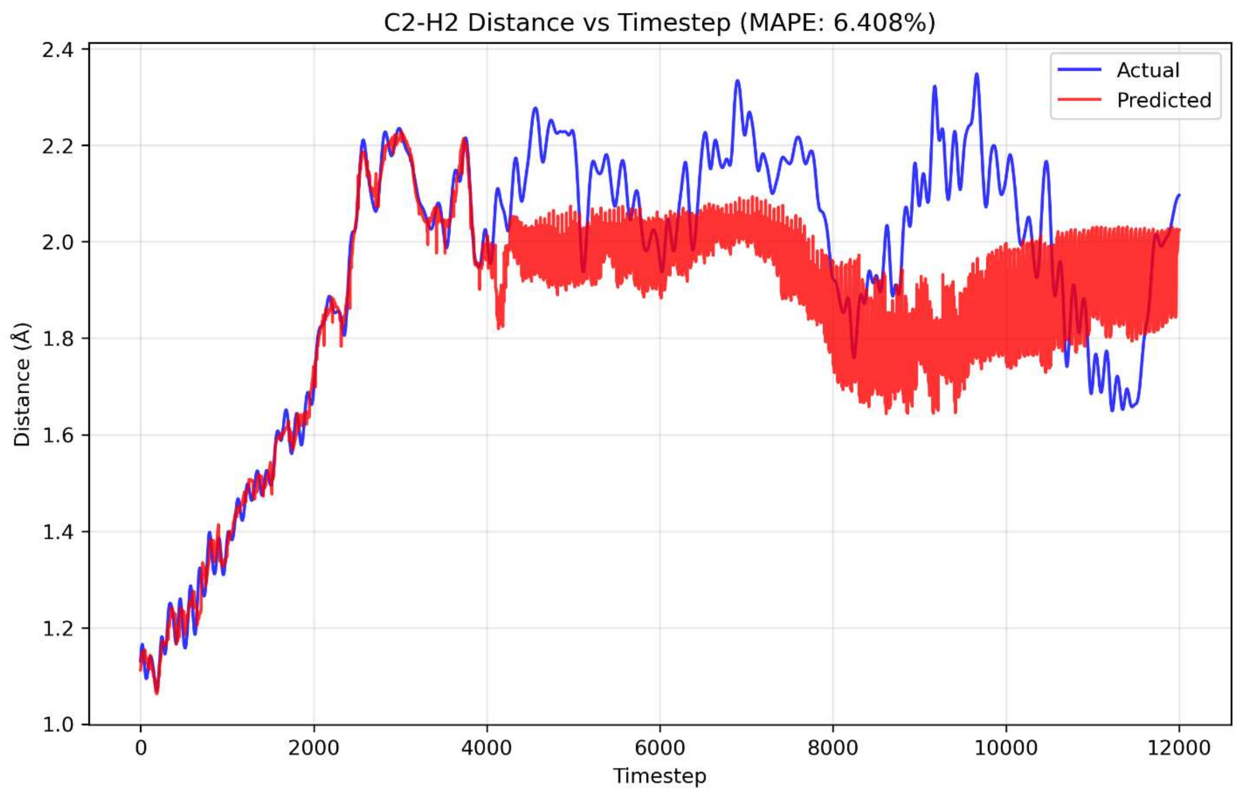


Figure 5.4: Comparison of actual (blue) and predicted (red) distances for the C_1-H_2 atomic pair over 12,000 timesteps. MAPE of 3.599% and $R^2 = 0.835$

The model achieves a MAPE of 3.599%, demonstrating the best predictive accuracy among all tested pairs. The prediction shows good agreement with actual values during the initial training phase (0-4,000 timesteps) and maintains reasonable accuracy during the inference phase (4,000-12,000 timesteps), though with increased variance in the autoregressive prediction region (as shown in figure 5.2).


C2-H2 Distance vs Timestep (MAPE: 6.408%)
Actual
Predicted
Distance (Å)
Timestep
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
2000
4000
6000
8000
10000
12000

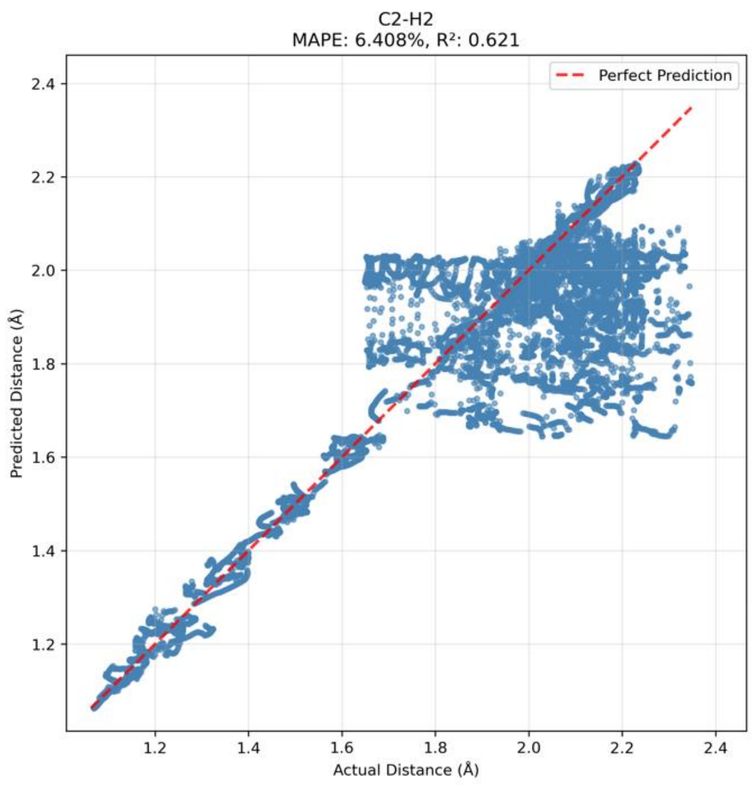
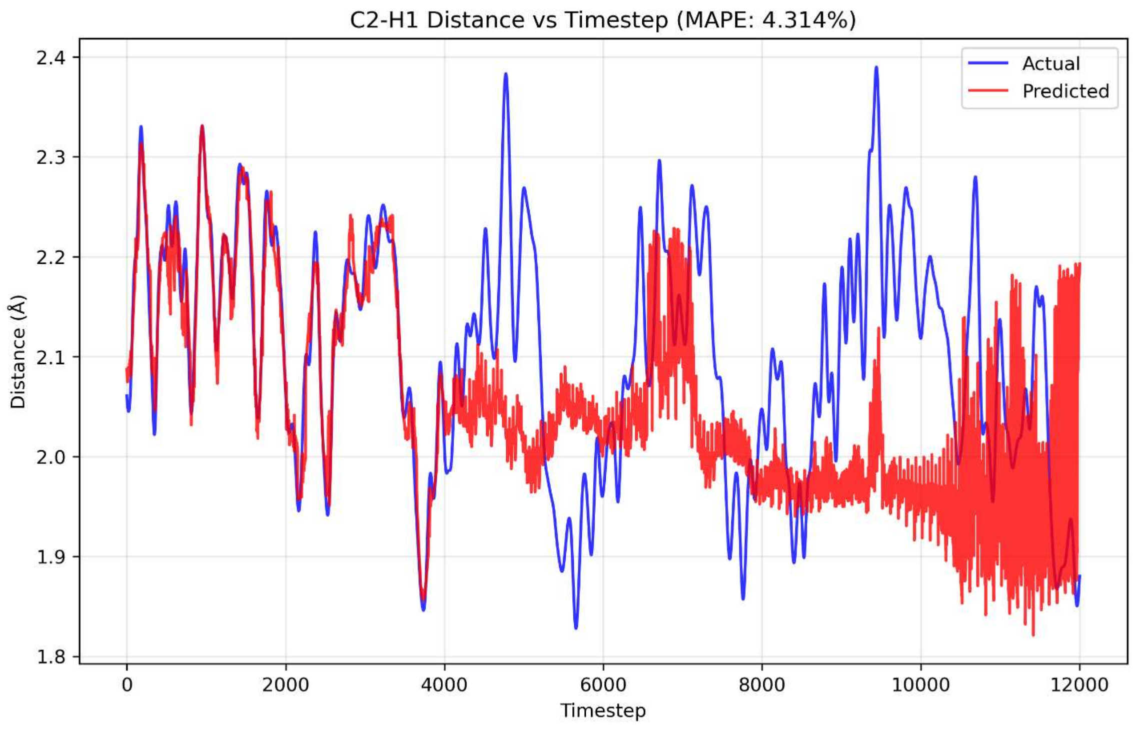

C2-H2
MAPE: 6.408%, R²: 0.621
Perfect Prediction
Predicted Distance (Å)
Actual Distance (Å)
2.4
2.2
2.0
1.8
1.6
1.4
1.2

Figure 5.5: Temporal evolution of actual and predicted distances for the C_2-H_2 pair, yielding a MAPE of 6.408% & $R^2 = 0.621$

The model successfully captures the major conformational transition occurring around timestep 2,000, where the distance increases from approximately 1.4 Å to 2.0 Å. However, the prediction exhibits higher uncertainty during the inference phase, particularly in the region between timesteps 6,000-8,000, where the model struggles to accurately reproduce the dynamic fluctuations present in the actual trajectory (as shown in figure 5.3).



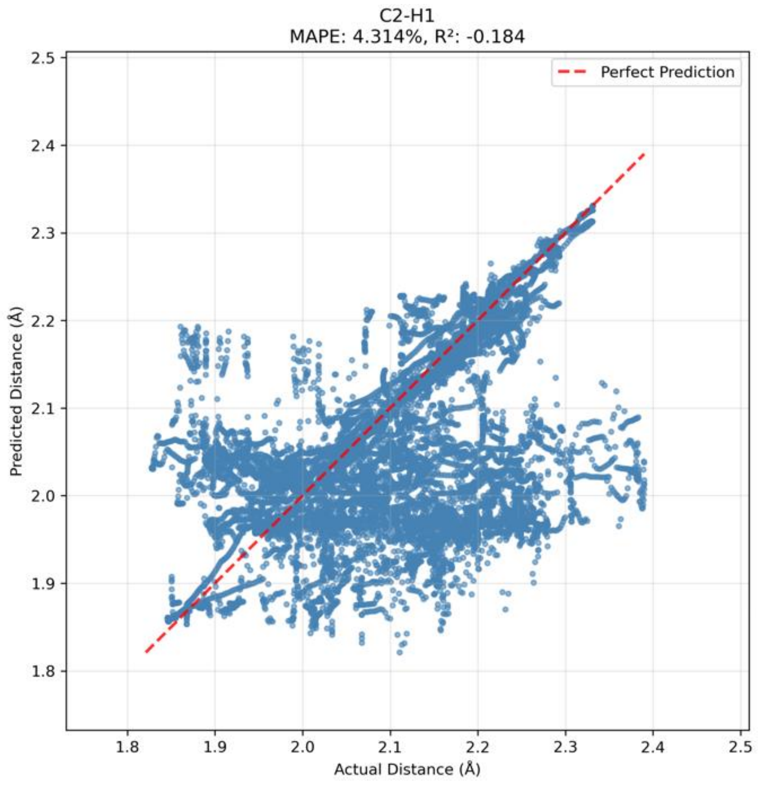
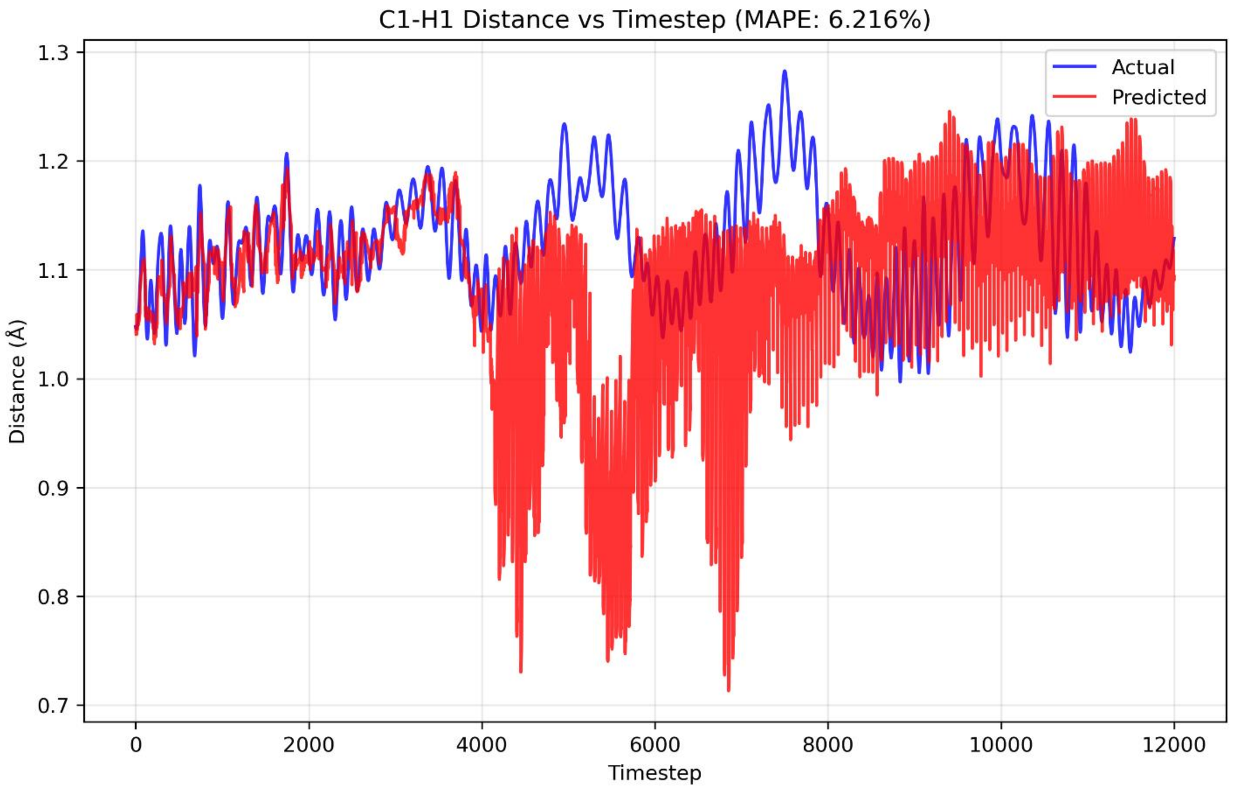


Figure 5.6: Prediction performance for C_2-H_1 atomic pair with a MAPE of 4.314% & $R^2 = -0.184$

The model demonstrates good tracking capability during the initial phase, accurately predicting the complex oscillatory behavior and conformational changes. Notable performance degradation occurs during the extended autoregressive prediction phase (8,000-12,000 timesteps), where the predicted trajectory shows increased variance and occasional systematic deviations from the actual molecular behavior (as shown in figure 5.4).



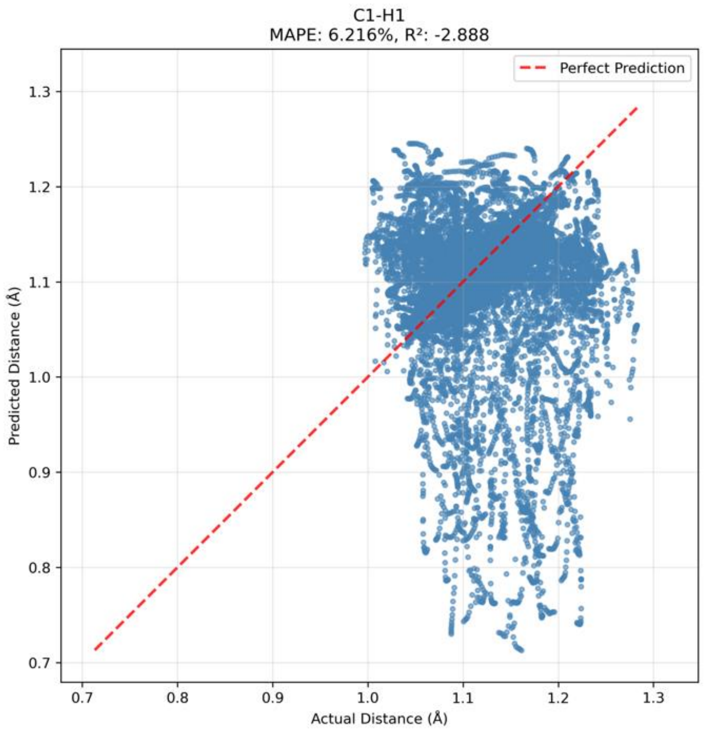


Figure 5.7: Prediction results for the C_1-H_1 pair showing a MAPE of 6.216% & R^2 = -2.888

The model exhibits mixed performance, with accurate short-term predictions during the training-influenced region but significant challenges in the inference phase. The prediction shows particular difficulty in capturing the deep conformational changes around timesteps 4,500-7,000, where the actual distance drops to approximately 0.7 Å, indicating close atomic approach events that the model fails to predict accurately (as shown in figure 5.5).

5.6.2 Prediction Accuracy Analysis

The MAPE values across the four atomic pairs reveal distinct performance characteristics:

- **C_1-H_2**: 3.599% (Best performance)
- **C_2-H_1**: 4.314% (Second best)
- **C_1-H_1**: 6.216% (Moderate performance)
- **C_2-H_2**: 6.408% (Highest error)

The average MAPE of 5.134% across all pairs indicates that the hybrid Transformer-LSTM architecture achieves reasonable accuracy for molecular dynamics prediction, though with notable variation depending on the specific atomic interactions being modeled.

While the R^2 analysis shows us how well the variance is captured in prediction and how good is the model fitting. For the pairs C_1-H_1 & C_2-H_1 the R^2 is way worse compared to C_1-H_2 & C_2-H_2 as it even below 0. Whereas the just MAPE analysis gives an incomplete insight of C_1-H_1 & C_2-H_1 performing better than C_2-H_2, which is not the case.

5.5.3 Temporal Performance Patterns

Analysis of the prediction results reveals several critical temporal patterns that characterize the model's behavior:

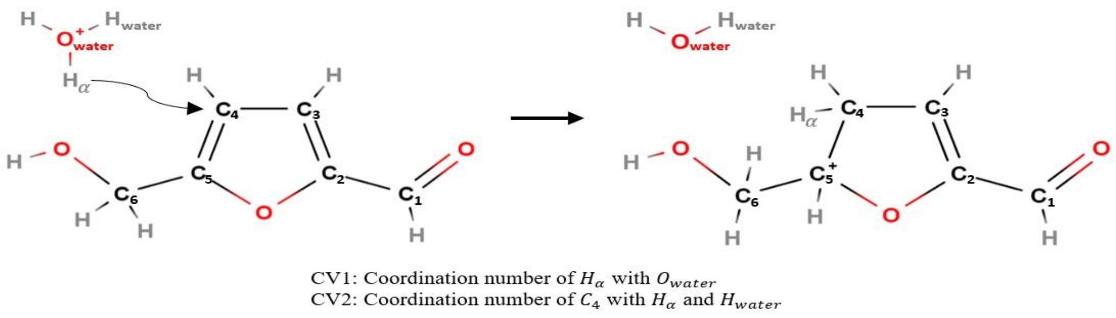
Training Phase Performance (0-4,000 timesteps): During the initial phase, where the model has access to ground truth context, predictions demonstrate high fidelity across all atomic pairs. The model successfully captures both short-term fluctuations and medium-term trends, indicating effective learning of the underlying molecular dynamics patterns.

Transition Region (4,000-8,000 timesteps): This period represents the critical transition from training-supported prediction to purely autoregressive forecasting. The model maintains reasonable accuracy initially but begins to show increased prediction variance as the temporal distance from training data increases.

Extended Autoregressive Phase (8,000-12,000 timesteps): Long-term autoregressive prediction reveals the model's limitations, with all atomic pairs showing degraded performance. The predictions tend to exhibit higher variance and occasional systematic bias, particularly for pairs undergoing significant conformational changes.

Chapter 6 - Leveraging attention mechanism to predict reaction energetics for condensed phase reaction system

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 5.1. For further details regarding the parameters of the CPMD-metadynamics simulation, refer to Calderón and Mushrif [87].



H_{water}
O^{+}_{water}
H_α
C_1
C_2
C_3
C_4
C_5
C_6
O_{water}
C_5^{+}
CV1: Coordination number of H_α with O_{water}
CV2: Coordination number of C_4 with H_α and H_{water}

Figure 6.1: Collective variables for the protonation of HMF. Adapted from Calderón and Mushrif [87].

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.

Equation 4.1 and Figure 6.1 suggest to take the particular atoms H_α, O_{water}, H_{water} and C_4 into consideration for the modeling. So from the trajectory file, the coordinates of these 4 atoms are extracted for all 200,000 timesteps and paired up to form a 2D array with rows as the timesteps and columns as the x,y and z of coordinates of atom 1 and atom 2 respectively in each pair. Pairs considered are H_α and O_{water}, C_4 and H_α, C_4 and H_{water}. These 2D arrays will be used as the training data to predict coordinates of future timestep, which gives the predicted Euclidean distance, which in turn gives the predicted CN and CV for the system. These predicted CVs will be later used to reconstruct the free energy surface. This chapter talks about modeling configuration of pair C_4 and H_{water} in detail. For other pairs the only difference is in the hyperparameters used.

6.1 Data Pre-processing and Preparation

The data preprocessing approach for the condensed phase HMF protonation system follows the same fundamental framework established in Chapter 5, with several key modifications to accommodate the increased scale and complexity of the condensed phase reaction system.

6.1.1 Dataset Scale and Temporal Partitioning

The condensed phase dataset encompasses 200,000 temporal steps, representing a ten-fold increase over the gas phase system. This extended temporal scope provides comprehensive coverage of the HMF protonation mechanism in the mixed DMSO-water solvent environment. The temporal partitioning implements a clean 150,000/50,000 training-testing split (75%/25%); this approach creates a more rigorous evaluation framework where the model must extrapolate to entirely unseen temporal regions.

6.1.2 Sequential Data Transformation

The sliding window methodology employs an extended sequence length of 25 timesteps compared to the 20 timesteps used in the gas phase system. This expansion reflects the longer temporal correlation lengths characteristic of condensed phase systems, where solvent interactions and environmental effects create extended dynamical dependencies. The target prediction framework maintains the same structure, predicting the six spatial coordinates while utilizing the binary flag as contextual input.

6.2 Model Architecture Framework

The hybrid Transformer-LSTM architecture receives substantial capacity increases to address the greater complexity of condensed phase molecular dynamics while maintaining the same hierarchical design philosophy described in Chapter 5.

6.2.1 Transformer Encoder

The Transformer encoder undergoes significant expansion with the following parameter increases:

- Head size: 256 dimensions (doubled from 128)
- Number of attention heads: 4 (doubled from 2)
- Feed-forward dimension: 256 (quadrupled from 64)
- Number of layers: 4 (doubled from 2)

This configuration creates a total attention capacity of 1024 dimensions, enabling the model to capture the intricate temporal relationships present in condensed phase systems where multiple interaction types and timescales contribute simultaneously to system evolution.

6.2.2 LSTM Processing Layer Expansion

The LSTM component receives substantial capacity increases while maintaining the four-layer architecture:

- Hidden units per layer: 128 (quadrupled from 32)
- Total layers: 4

The expanded hidden dimensions provide sufficient representational space to encode the complex temporal dependencies characteristic of solvated molecular systems, where solvent effects, intermolecular interactions, and collective motions create intricate dynamical patterns.

6.3 Training Configuration and Hyperparameters

6.3.1 Training Parameters

The training configuration incorporates several modifications to accommodate the larger model and dataset:

- Batch size: 128 (quadrupled from 32) to enable more stable gradient estimation
- Maximum epochs: 1000 with early stopping patience of 50 epochs
- Learning rate scheduling: Reducing LR with patience of 20 epochs and factor of 0.5
- Dropout rate: 0.1 (maintained from Chapter 5)
- Loss function: MAPE (maintained from Chapter 5)

6.3.2 Regularization Strategy

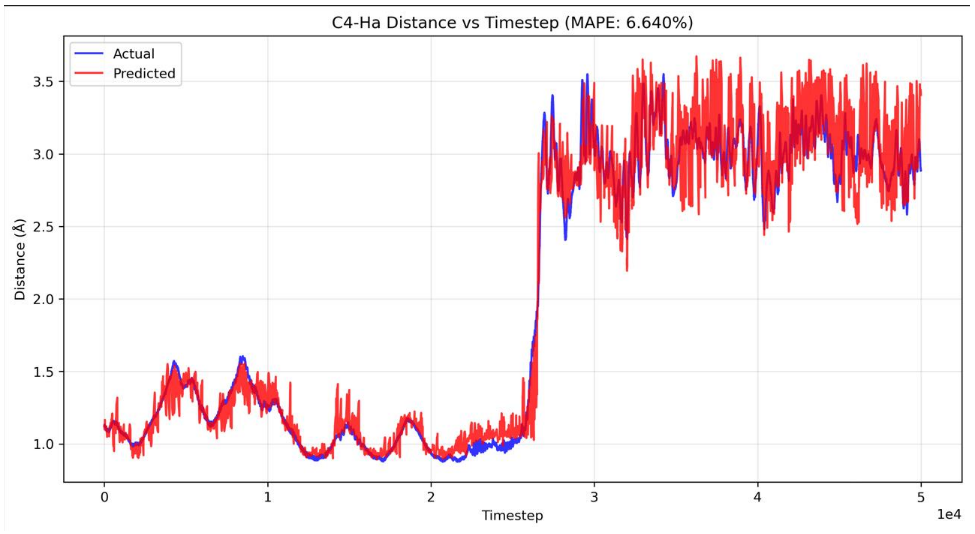
6.4 Extended Multi-Horizon Inference Methodology

The inference methodology extends the evaluation framework to five prediction horizons specifically chosen for condensed phase system analysis: 100, 500, 1000, 2000, and 5000 time steps. These horizons span a much broader temporal range compared to the 50, 100, 500 time step horizons employed in Chapter 5, enabling comprehensive assessment of model performance across time scales. Due to availability of a larger dataset, this model is well trained to explore larger horizons.

6.5 Results

The hybrid Transformer-LSTM model was evaluated on the condensed phase HMF protonation system using three critical atomic pair interactions (C_4-H_a, C_4-HW1, H_a-OW) that define the collective variables for the protonation mechanism. The model demonstrated varying performance characteristics across these interactions, revealing the challenges associated with predicting molecular dynamics in complex solvated environments where multiple interaction types and longer-range correlations influence system behavior. The results are discussed for H = 100.

6.5.1 Model Performance Across Critical Atomic Pairs



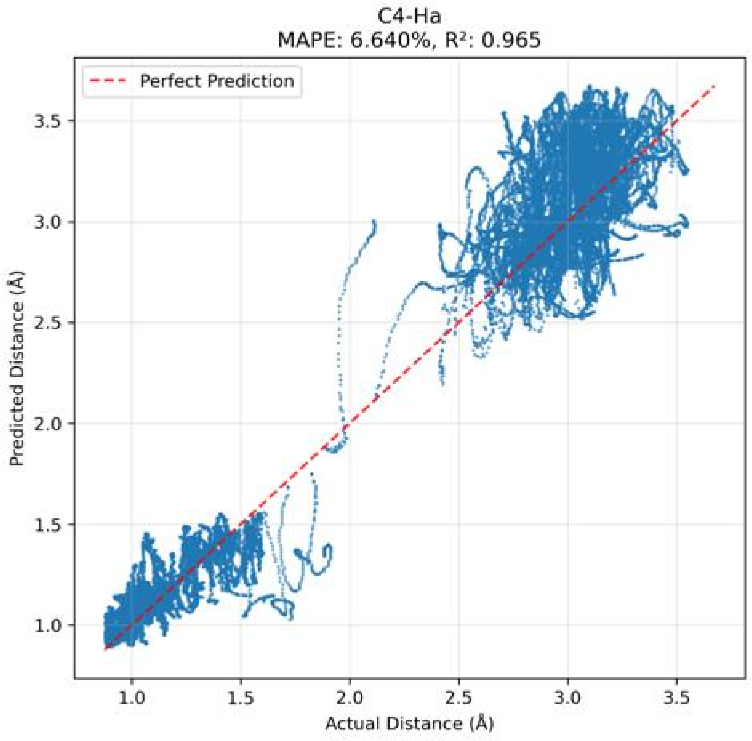
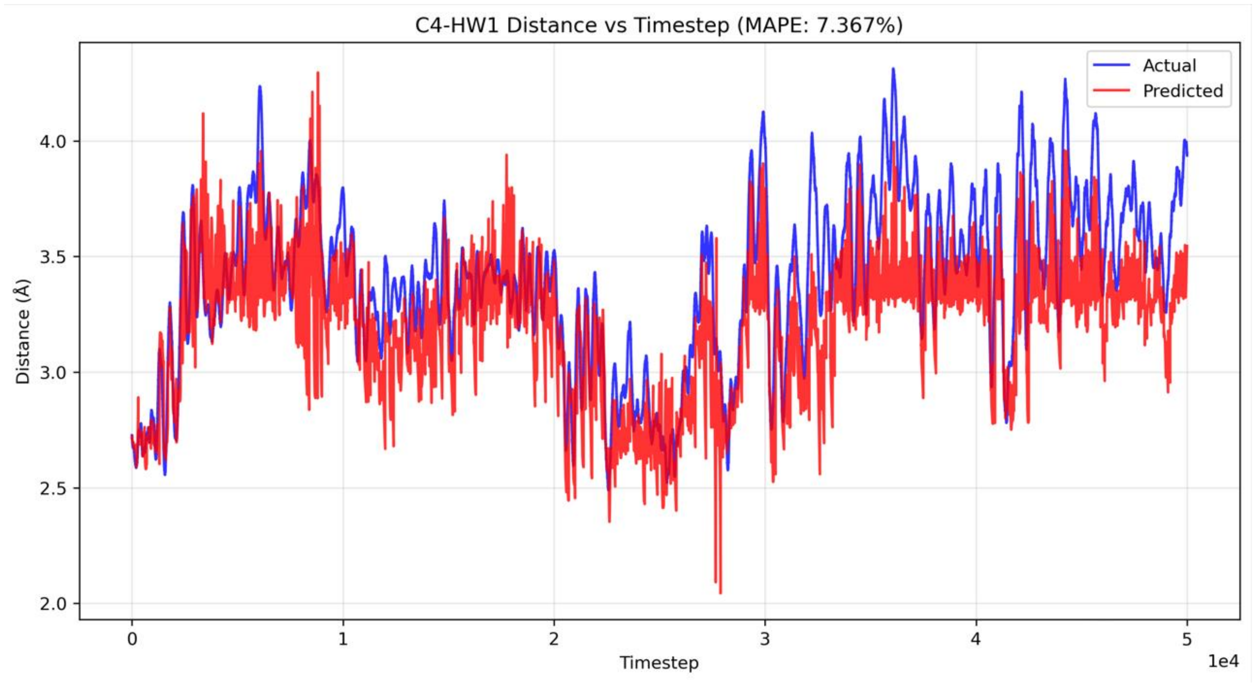


Figure 6.2: Temporal evolution of actual (blue) and predicted (red) distances for the C_4-H_a pair over 50,000 timesteps, achieving a MAPE of 6.640% & $R^2 = 0.965$

The model demonstrates excellent performance during the first 30,000 timesteps, accurately capturing both the periodic oscillations around 1.0-1.6 Å and the critical reaction transition beginning at timestep 30,000. The dramatic distance increase from ~1.0 Å to ~3.5 Å represents the protonation event, which the model predicts with reasonable fidelity. However, prediction accuracy degrades in the post-reaction phase (35,000-50,000 timesteps), where the model exhibits increased variance while maintaining the general equilibrium distance range (as shown in figure 6.2). R^2 (coefficient of determination) quantifies how well the model explains the variance in the observed data, ranging from 0 to 1. An R^2 value of 0.965 indicates that the model accounts for 96.5% of the variance in C4-Ha distances, demonstrating excellent predictive capability and strong correlation between predicted and actual molecular dynamics.



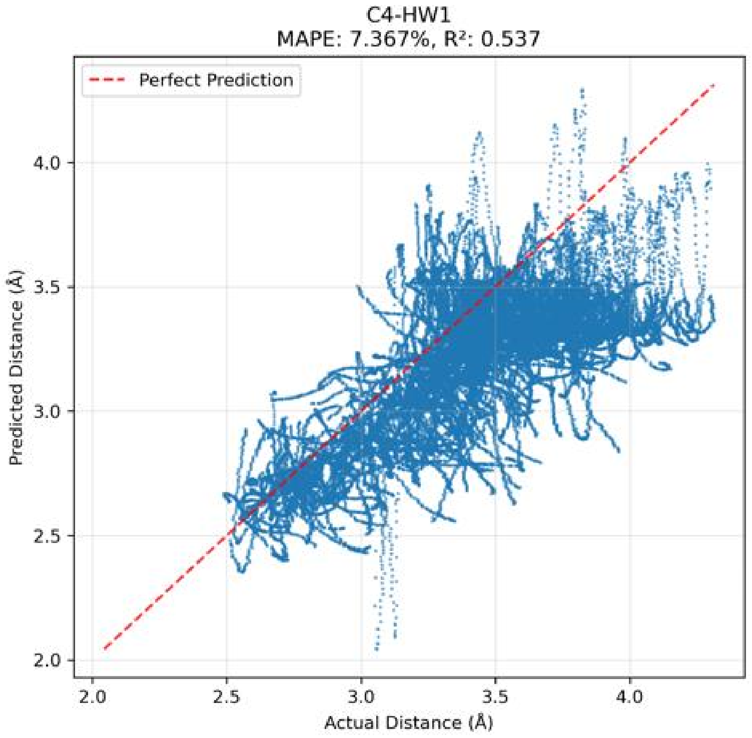
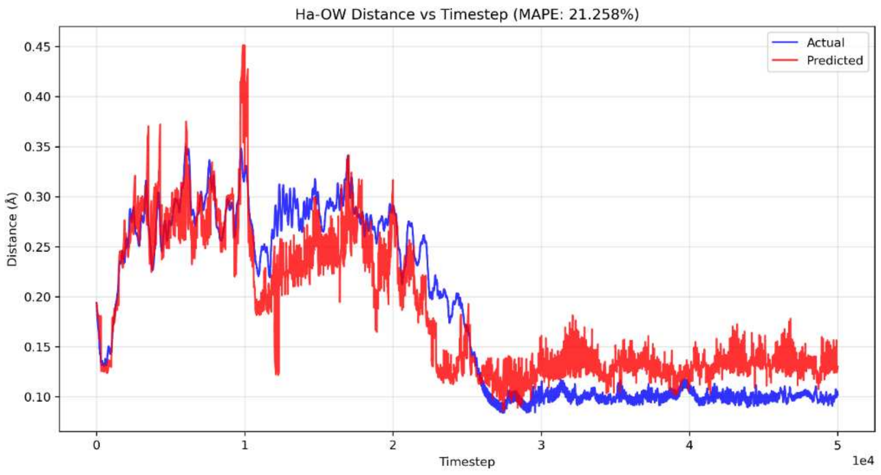


Figure 6.3: Prediction performance for the C_4-HW1 pair with a MAPE of 7.367% and $R^2 = 0.537$.

The model successfully tracks the complex pre-reaction dynamics during the first 25,000 timesteps, capturing the oscillatory behavior between 2.6 - 4.3 Å that characterizes the approach of water molecules to the reaction center. The model accurately predicts the conformational transition around timestep 25,000 and maintains good agreement through the stabilization period (30,000-50,000 timesteps) where distances equilibrate around 3.2 - 4.2 Å. The prediction shows consistent performance across the entire trajectory with moderate variance in the autoregressive prediction phase (as shown in figure 6.3). Though MAPE is good, the $R^2 = 0.537$ implies the variance has not been captured to the best extent.



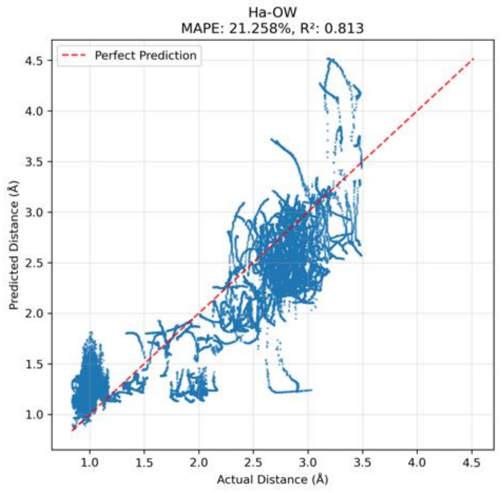


Figure 6.4: Prediction results for the H_a-OW pair showing the highest MAPE of 21.258%.

The model accurately captures the initial pre-reaction phase (0-10,000 timesteps) with complex oscillations between 1.3 – 4.5 Å representing proton-water oxygen interactions. However, significant prediction challenges emerge during the reaction transition (10,000-25,000 timesteps), where the model struggles to accurately predict the dramatic conformational changes and the subsequent equilibration to lower distances (~1.0 Å) in the post-reaction phase (as shown in figure 6.4). The extended autoregressive prediction shows substantial variance and systematic bias, indicating difficulties in capturing the long-range electrostatic interactions governing proton-solvent dynamics. $R^2 = 0.813$ suggests that variance has been captured to a good extent.

6.5.2 Prediction Accuracy Analysis

The MAPE values across the three critical atomic pairs reveal distinct performance characteristics that reflect the underlying physics of the protonation mechanism:

- **C_4-H_a**: 6.640% (Best performance)
- **C_4-HW1**: 7.367% (Moderate performance)
- **H_a-OW**: 21.258% (Highest error)

The average MAPE of 11.755% across the critical pairs is significantly higher than the gas phase system (5.134%), reflecting the increased complexity of condensed phase molecular dynamics where solvent effects, hydrogen bonding networks, and long-range electrostatic interactions create more challenging prediction scenarios.

6.5.3 Temporal Performance Patterns and Reaction Mechanism Capture

Analysis of the prediction results reveals several critical insights into the model's ability to capture reaction mechanisms:

- **Pre-Reaction Phase Performance (0-25,000 timesteps)**: The model demonstrates strong performance during the approach phase, successfully capturing the complex molecular configurations that precede the protonation event. The C_4-H_a and C_4-HW1 pairs show particularly good agreement with ground truth, indicating effective learning of the solvent-mediated approach dynamics.
- **Reaction Transition Capture (25,000-35,000 timesteps)**: The model successfully identifies and predicts the critical reaction transition for all three pairs, demonstrating its capability to capture the fundamental change in molecular configuration associated with proton transfer. The C_4-H_a distance increase and the corresponding changes in C_4-HW1 and H_a-OW distances are predicted with reasonable accuracy.
- **Post-Reaction Equilibration (35,000-50,000 timesteps)**: Extended autoregressive prediction in the product state reveals varying degrees of stability. The C_4-H_a and C_4-HW1 pairs maintain reasonable predictive accuracy, while the H_a-OW pair shows significant degradation, suggesting challenges in modeling the complex hydrogen bonding networks that stabilize the protonated product in the condensed phase environment.

Chapter 7 - Constructing Free Energy Surface & Future Work

7.1 Deconstructing CPMD Metadynamics Simulation Outputs

The successful execution of a metadynamics simulation using the Car-Parrinello Molecular Dynamics (CPMD) package generates a set of output files that fully encode the history of the biasing process. For the purpose of reconstructing the free energy surface, two files are of paramount importance: `colvar_mtd` and `parvar_mtd`. These plain-text files serve as the direct input for post-processing tools and contain all the necessary information to rebuild the history-dependent bias potential, $V_G(s,t)$, and subsequently, the Free Energy Surface (FES). This section provides a detailed breakdown of the structure and content of these files, using the specific outputs from the present study as illustrative examples.

7.1.1 The colvar_mtd File: Tracking the System in Collective Variable Space

The `colvar_mtd` file is a time-series record of the system's trajectory as projected onto the chosen collective variable space. Its primary function is to log the precise coordinates (i.e., the values of the CVs) at which each Gaussian hill was centered during the simulation. It is, in essence, the "where" component of the biasing history.

An examination of a representative segment of the `colvar_mtd` file from this work (Figure 7.1, corresponding to the provided Image 2) reveals its structure. The file is organized into columns of numerical data, where each row corresponds to a specific simulation timestep at which a Gaussian hill was deposited.

```
...
3100        0.103774E+01        0.957510E+00        0.100E+01        0.100E+01
3200        0.100912E+01        0.970944E+00        0.100E+01        0.100E+01
3300        0.969211E+00        0.977342E+00        0.100E+01        0.100E+01
...
```

Figure 7.1: A segment of the `colvar_mtd` file, illustrating the format for a two-CV metadynamics simulation.

A column-by-column analysis of this file is as follows [87, 88, 89]:

- **Column 1:** The simulation timestep. This integer value serves as the unique identifier for each hill deposition event and is the key for synchronizing this file with the `parvar_mtd` file. In the example, hills are deposited every 100 timesteps (e.g., 3100, 3200, 3300).
- **Column 2:** The instantaneous value of the first Collective Variable (CV1) at the time of deposition. For the timestep 3100, this value is approximately 1.038. This number defines the center of the deposited Gaussian along the first dimension of the CV space.
- **Column 3:** The instantaneous value of the second Collective Variable (CV2). At timestep 3100, this value is approximately 0.958. This defines the center of the Gaussian along the second CV dimension.
- **Subsequent Columns:** In simulations with more than two CVs, their values would appear in subsequent columns. The final columns (in this case, columns 4 and 5, with values of

1.0) typically represent scaling factors or other metadata related to the CV definition, which are used internally by the reconstruction code.

7.1.2. The parvar_mtd File: Parameters of the Biasing Potential

Complementing the `colvar_mtd` file, the `parvar_mtd` file logs the specific parameters of each Gaussian hill deposited. It provides the "what" and "how" of the biasing history, detailing the shape (width) and magnitude (height) of each repulsive potential added to the landscape.

```
...
3100            0.031618            0.100000            0.001000
3200            0.040417            0.100000            0.001000
3300            0.036008            0.100000            0.001000
...
```

Figure 7.2: A segment of the `parvar_mtd` file, showing the parameters of the Gaussian hills deposited at each timestep.

The columns in this file are interpreted as follows [88, 89]:

- **Column 1:** The simulation timestep. This column is identical to the first column of `colvar_mtd`, ensuring a one-to-one correspondence between the hill's center and its parameters.
- **Column 2:** The norm of the total displacement in the CV-space since the last hill deposition. This is a diagnostic quantity that can be used to monitor the diffusive behavior of the system in CV space.

- **Column 3:** The width (σ) of the Gaussian hill. In this example, the width is constant at 0.100000 (in atomic units). If multiple CVs with different widths were used, or if an adaptive width scheme were employed, this column would be followed by others specifying the width for each CV.
- **Column 4:** The height (W) of the Gaussian hill, given in atomic units of energy (Hartrees). In this example, the height is constant at 0.001000, suggesting a standard (non-well-tempered) metadynamics run or the early stages of a well-tempered run before significant bias has accumulated. In a converged well-tempered simulation, the values in this column would generally decrease over time.

The combination of these two files provides a complete and self-contained description of the entire biasing potential generated during the simulation. This separation of the simulation's trajectory from the parameters of the bias potential is a powerful feature of the computational workflow. It means that the computationally intensive task the ab initio MD simulation itself needs to be performed only once to generate this pair of relatively small text files. The subsequent step of FES reconstruction, which is computationally inexpensive, can then be performed independently, allowing for repeated analysis with different grid parameters or convergence checks without the need to re-run the simulation. These two files are, in effect, a complete and reproducible recipe for building the bias potential, and by extension, the free energy surface.

File	Column 1	Column 2	Column 3	Column 4	Column 5	Role in Reconstruction
`colvar_mtd`	**Timestep**	Value of CV1	Value of CV2	...	Scaling Factor(s)	Provides the **center** ($s(t')$) of each Gaussian

File	Column 1	Column 2	Column 3	Column 4	Column 5	Role in Reconstruction
`parvar_mtd`	**Timestep**	Displacement Norm	Width (σ) of CV1	Height (W)	...	Provides the **shape** (σ, W) of each Gaussian

Table 7.1: Structure and Interrelation of CPMD Metadynamics Output Files.

7.2. Free Energy Surface Reconstruction with the Vreco Tool

Once the CPMD metadynamics simulation is complete and the `colvar_mtd` and `parvar_mtd` files have been generated, the next critical step is the post-processing of this data to construct the final free energy surface. This process involves summing the contributions of all the deposited Gaussian hills on a discrete grid spanning the collective variable space. For this task, the Vreco program was employed, a specialized tool designed for reconstructing the FES from CPMD metadynamics outputs. This section details the mathematical basis of the reconstruction algorithm implemented in Vreco, provides a practical guide to its usage, and discusses the analysis of its outputs to ensure the convergence and reliability of the resulting FES.

7.2.1. Mathematical Formulation of FES Reconstruction in Vreco

The Vreco tool implements the fundamental principle of metadynamics: the free energy surface, F(s), is the negative of the total accumulated bias potential, $V_G(s)$. Vreco operationalizes this by first defining a multi-dimensional grid in the CV space according to user specifications. It then iterates through every point s on this grid and calculates the value of the FES at that point by summing the contributions from every Gaussian hill deposited during the simulation.

The explicit mathematical formula calculated by Vreco (equation 7.1) for the free energy at a specific grid point s=(s_1,s_2,...,s_d) is [88]:

$$F(\mathbf{s}) = -\sum_{k=1}^{N_{\text{hills}}} W_k \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_{i,k})^2}{2\sigma_{i,k}^2}\right) \tag{7.1}$$

This equation directly links the output files from CPMD to the final FES [2]:

- The outer sum is over all N_{hills} deposited during the simulation, corresponding to the total number of lines in the `colvar_mtd` and `parvar_mtd` files.
- For each hill k, the height W_k and the width for each CV dimension $\sigma_{i,k}$ are read directly from the corresponding row k in the `parvar_mtd` file.
- The center of the Gaussian hill in each dimension, $s_{i,k}$, is read from row k of the `colvar_mtd` file.

The reconstruction process is fundamentally an act of interpolation. The metadynamics simulation provides information about the landscape at a discrete set of points (the Gaussian centers). Vreco uses this information to construct a continuous surface by assuming that the influence of each sampled point decays in a Gaussian manner. The value of the FES at a grid point that was never directly visited by the simulation is therefore an interpolation based on its proximity to the points that were visited. The density of Gaussian centers and the width of the Gaussians themselves determine the fidelity of this interpolation.

The condensed phase dataset encompasses 200,000 temporal steps, representing a ten-fold increase over the gas phase system. This extended temporal scope provides comprehensive coverage of the HMF protonation mechanism in the mixed DMSO-water solvent environment. The temporal partitioning implements a clean 150,000/50,000 training-testing split (75%/25%), this approach creates a more rigorous evaluation framework where the model must extrapolate to entirely unseen temporal regions.

7.3. Analysis of the Reconstructed FES and Future Directions

The culmination of the metadynamics simulation and the Vreco reconstruction is a quantitative map of the free energy landscape. This landscape is not merely a graphical representation but a rich source of thermodynamic and mechanistic information about the process under investigation. This section details the methods for interpreting the FES to extract chemically and biologically meaningful insights. Furthermore, it outlines a strategic plan for future research, demonstrating how the constructed FES serves as a foundation for new hypotheses and more advanced computational experiments.

7.3.1. Interpreting the Free Energy Landscape

The first step in the analysis is the visualization of the data contained within the `V.final.out` file. For a two-dimensional FES, this is typically done by creating a 2D contour plot, where lines of constant free energy (isocontours) are drawn, and the regions between them are colored according to their energy value. This representation provides an intuitive, top-down view of the landscape, much like a topographical map.

The topology of the FES is directly linked to the thermodynamics and stability of the system's various conformational states. The key features to identify are:

- **Local Minima (Basins):** These are the "valleys" on the FES. Each distinct basin corresponds to a thermodynamically stable or metastable state of the system. For a chemical reaction, these could be the reactant, product, and any long-lived intermediate states.

- **Saddle Points:** These are the regions connecting adjacent basins. A first-order saddle point is a maximum along the lowest-energy path between two minima but a minimum in all other orthogonal directions. These points on the FES represent the transition state (TS) ensemble for the transformation between the two states they connect. The location of the saddle point provides crucial information about the structure and geometry of the transition state.

7.3.2. Elucidating Reaction Pathways and Energetics

With the key states identified, the FES can be used to map the most probable pathway for the transformation and to calculate the associated energetic barriers.

- **Minimum Energy Path (MEP):** The MEP represents the most likely trajectory in CV space for a transition from one metastable state to another. It is the path that traverses the minimum energy barrier. On the FES, the MEP can be traced by starting at the saddle point (transition state) and following the path of steepest descent down into the connected reactant and product basins. This path provides a clear, visual representation of the reaction mechanism or conformational change pathway.
- **Quantitative Energetic Analysis:** The FES allows for the direct calculation of the key thermodynamic quantities that govern the kinetics and equilibrium of the process.
    - **Activation Free Energy ($\Delta G^{\ddagger}$):** This is the free energy barrier that must be overcome for the transformation to occur. It is calculated as the difference in free

energy between the transition state (saddle point) and the initial (reactant) state: $\Delta G^{\ddagger} = F_{\text{TS}} - F_{\text{Reactant}}$ This value is a primary determinant of the reaction rate.

- **Reaction Free Energy (ΔG_{rxn}):** This quantity determines the overall thermodynamic favorability of the process. It is calculated as the difference in free energy between the final (product) and initial (reactant) states: $\Delta G_{\text{rxn}} = F_{\text{Product}} - F_{\text{Reactant}}$ A negative ΔG_{rxn} indicates a spontaneous process, while a positive value indicates a non-spontaneous one under the simulated conditions.

7.4. Construction of FES for Gas Phase Reaction System

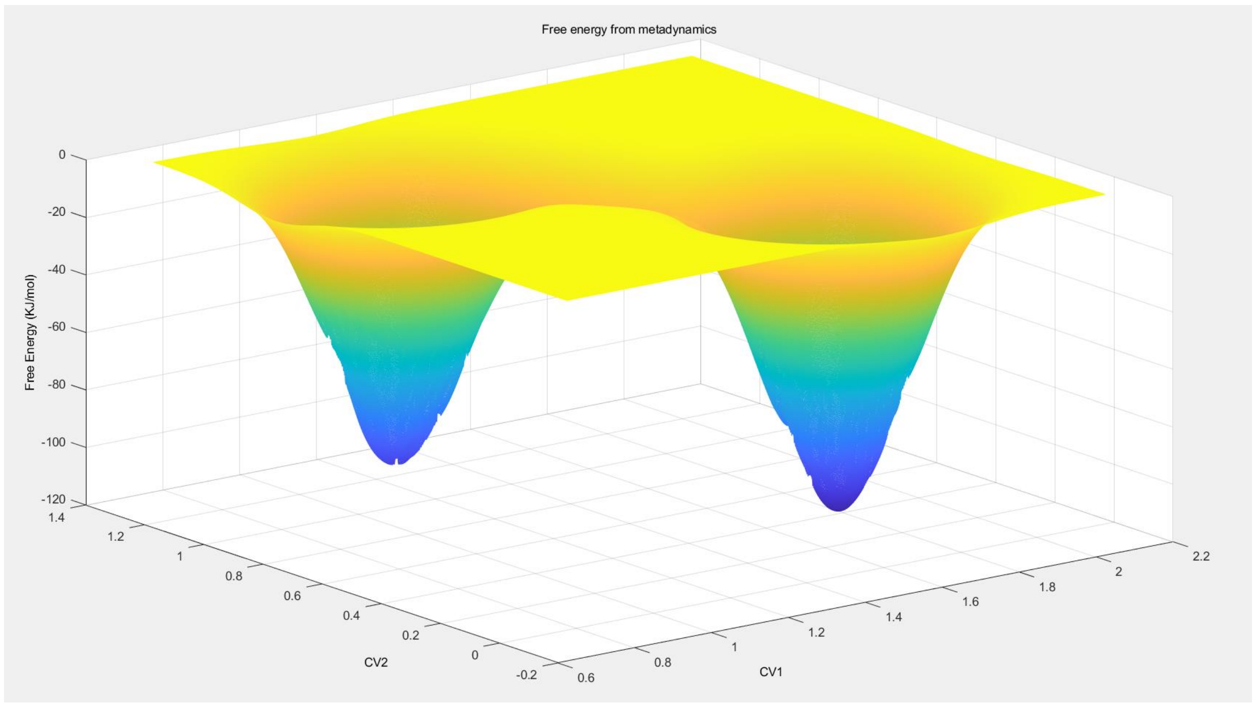
From chapter 5, we used the proxy model to determine the co-ordinates of atoms that are important for CV calculation. Once we have the co-ordinates, we calculate the Euclidean distances and use them to calculate the Co-ordination Number (C.N.) as given by equation 4.1. Particular definition of CN constitutes the CV for a reaction system.

For the gas phase system, we replace the column 2 and column 3 of colvar_mtd file for all the timesteps that have been predicted (i.e. 8000 timesteps out of 20,000 timesteps) by the model and aim to reconstruct the FES. The parvar_mtd file remains the same, as it is required to provide information on shape of the gaussians dropped during metadynamics. Using Vreco we get the FES, which have been discussed in the next section. The FES predicted have been compared to FES reported in Mushrif et al [54].

7.5. FES Results

7.5.1 Gas Phase Reaction System

7.5.1.1 Ground Truth FES from CPMD Simulations



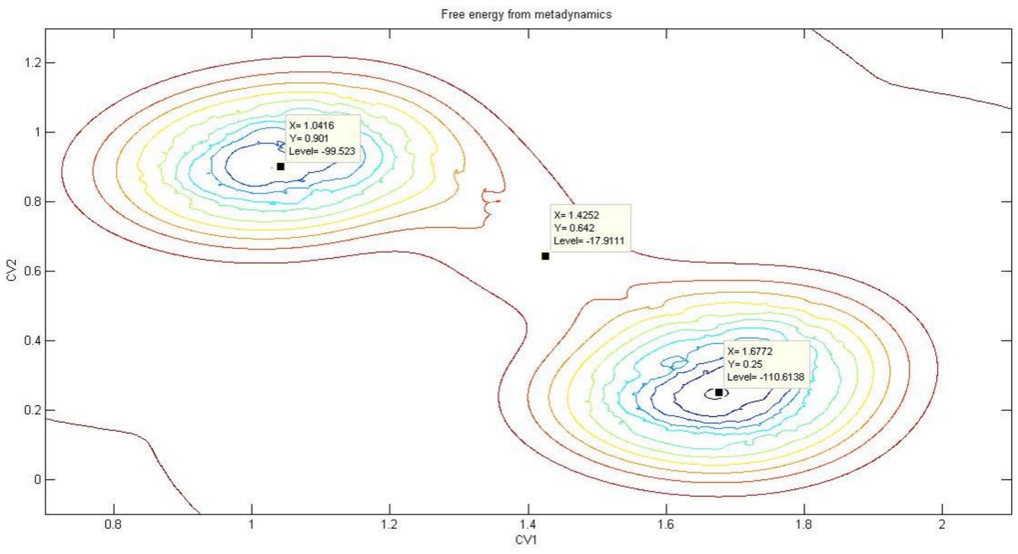
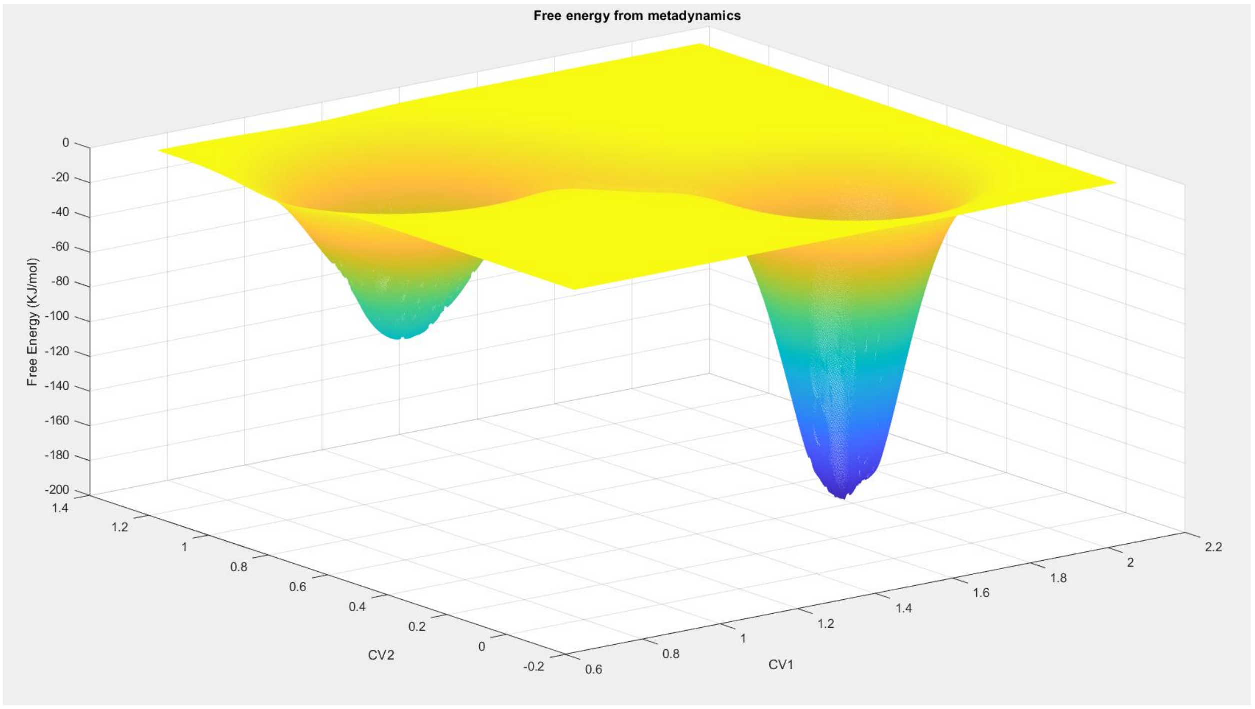


Figure 7.3: Ground truth free energy surface for the gas phase intramolecular hydride transfer reaction, adapted from Mushrif et al. The 3D surface representation shows the characteristic double-well topology with reactant state at CV1≈1.0, CV2≈1.0 (Energy = -99.5 kJ/mol) and product state at CV1≈1.5, CV2≈0.25 (Energy = -110.1 kJ/mol). The reaction proceeds through a transition state with an activation barrier of 82 kJ/mol, resulting in an overall reaction energy of -10.6 kJ/mol, indicating a mildly exothermic process. The two-dimensional contour plot of the reference free energy surface showing three distinct regions: the reactant basin (upper left), transition state region (center), and product basin (lower right). The minimum energy path connects these states through a well-defined saddle point, providing the thermodynamically most favorable reaction pathway for the intramolecular hydride transfer mechanism.

7.5.1.2 Proxy Model-Generated FES Results



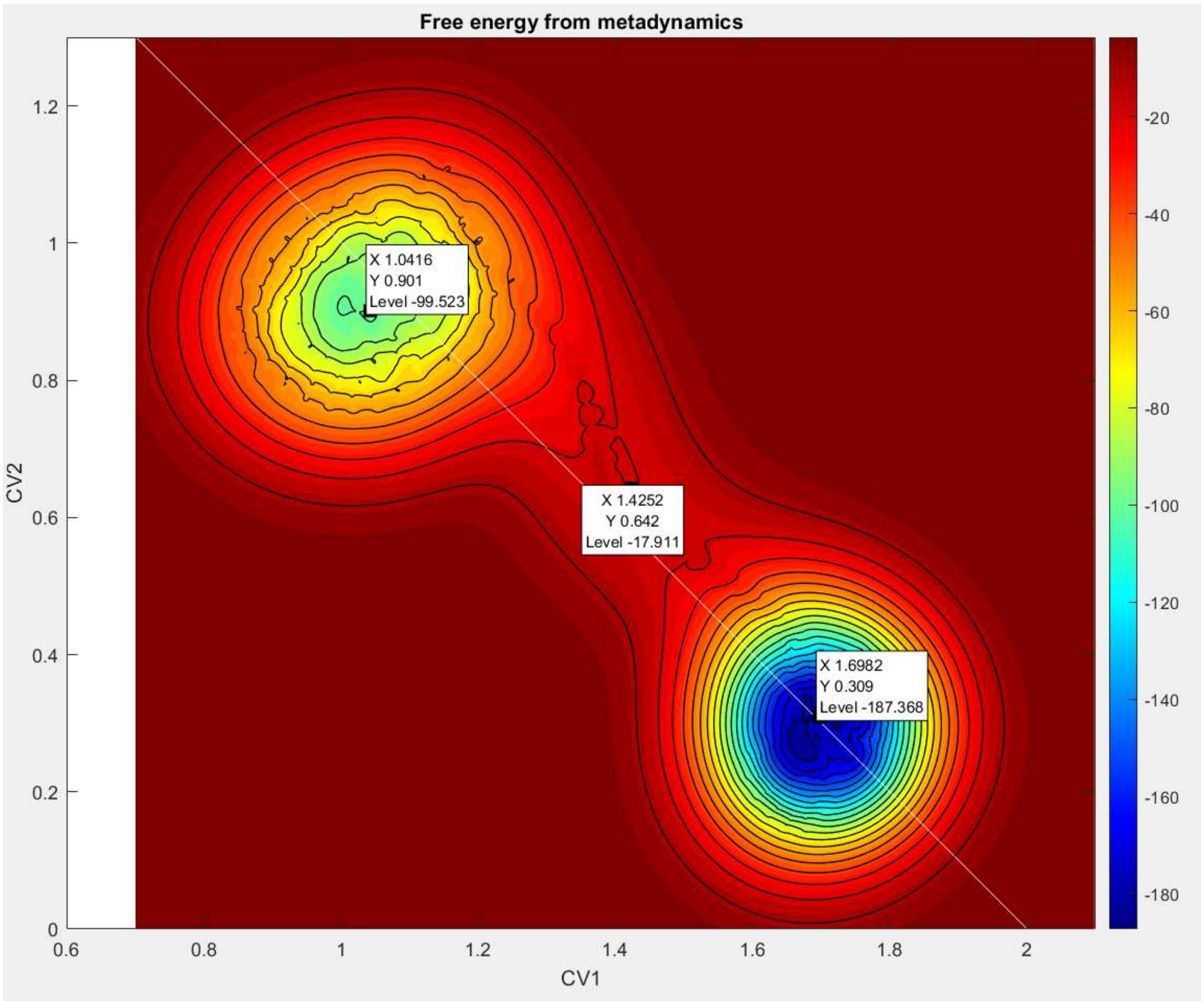
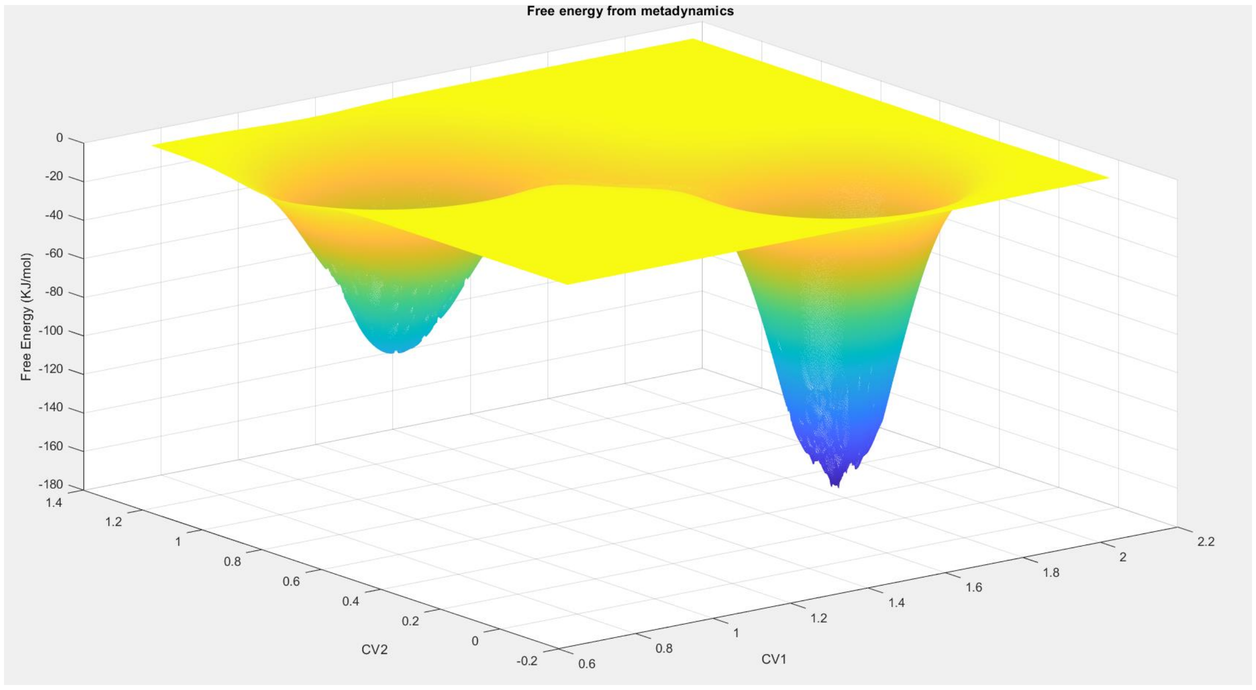


Figure 7.4: Free energy surface reconstructed using predicted distances for all atomic pairs involved in CV calculation (C_1-H_1, C_1-H_2, C_2-H_1, C_2-H_2).

The model successfully reproduces the fundamental double-well topology with reactant minimum at CV1≈1.04, CV2≈0.90 (-99.5 kJ/mol) and product minimum at CV1≈1.70, CV2≈0.31 (-187.4 kJ/mol). However, the predicted reaction energy of -87.9 kJ/mol significantly overestimates the exothermicity compared to the reference value of -10.6 kJ/mol (as shown in figure 7.4).



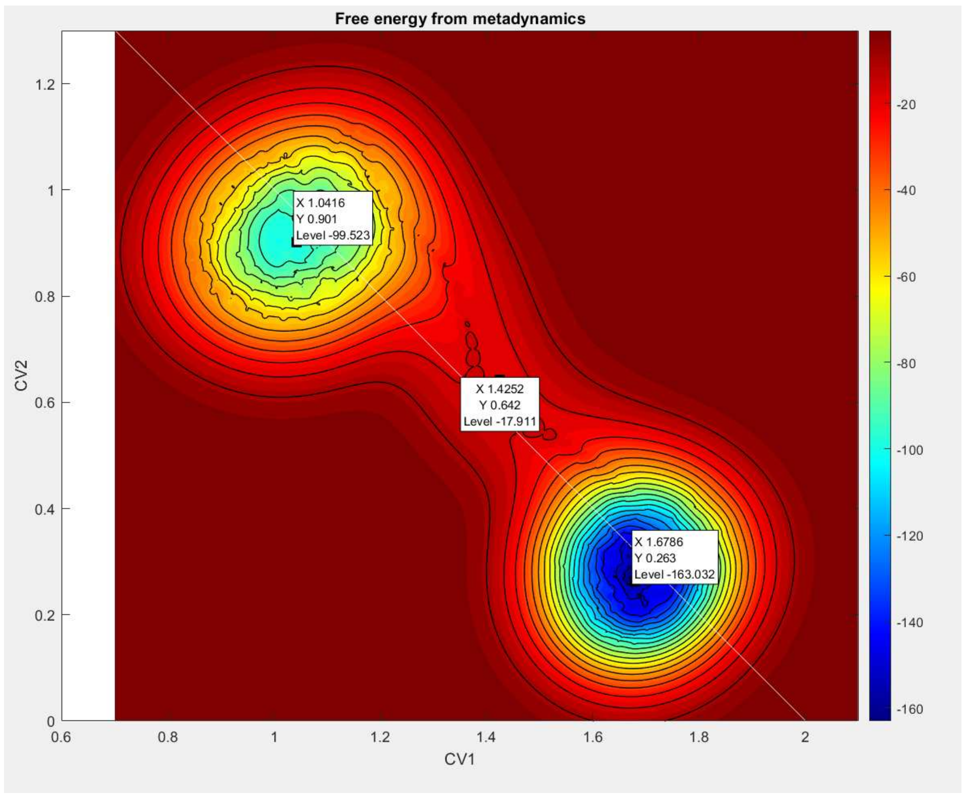
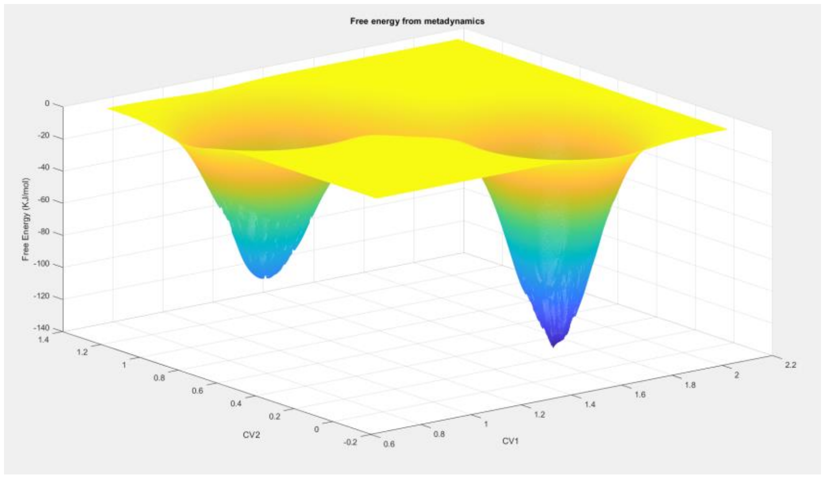


Figure 7.5: FES reconstruction using actual distances for C_1-H_1 and C_2-H_1 pairs combined with predicted distances for C_1-H_2 and C_2-H_2 pairs.

This hybrid approach maintains the same structural topology as Configuration 1 but shows improved thermodynamic accuracy with a reaction energy of -63.5 kJ/mol. The incorporation of actual data for half the atomic pairs provides partial calibration of the energy scale while preserving the overall reaction pathway (as shown in figure 7.5).



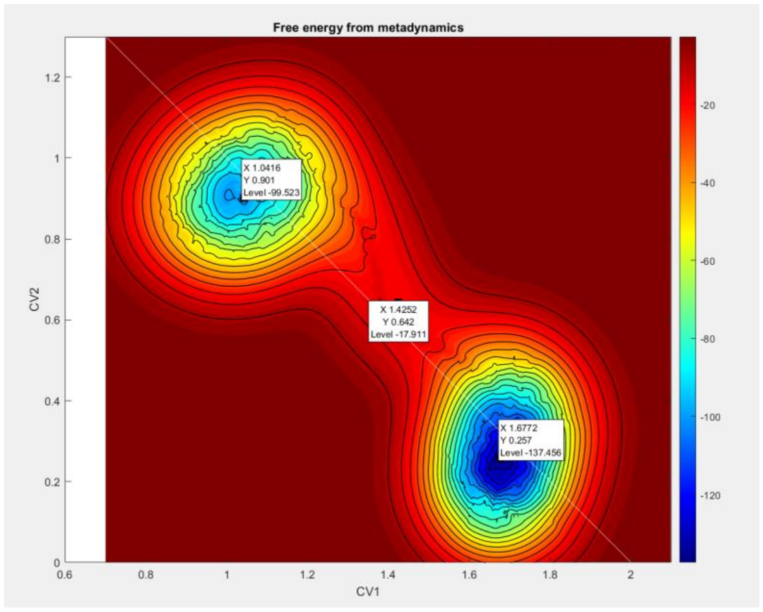


Figure 7.6: FES generated using actual CV2 values from CPMD simulation and mixed actual/predicted distances for CV1 calculation (actual C_1-H_1, predicted C_1-H_2).

This configuration yields the most accurate thermodynamic prediction with a reaction energy of -38.0 kJ/mol, representing a substantial improvement over fully predicted approaches while maintaining excellent structural fidelity to the reference FES topology (as shown in Fig 7.6).

7.5.1.3 Comparative Analysis and Model Performance Assessment

Structural Fidelity: All three proxy model configurations demonstrate remarkable consistency in reproducing the fundamental structural features of the reference FES. The reactant and product state locations in CV space show excellent agreement across all approaches, with CV coordinates varying by less than 0.05 units from the target positions. The transition state region maintains consistent positioning at CV1≈1.43, CV2≈0.64, indicating that the model successfully captures the mechanistic pathway for the intramolecular hydride transfer.

The preservation of the double-well topology across all configurations represents a significant achievement, as this structural characteristic is essential for understanding the reaction mechanism. The model's ability to maintain this topology even under fully predictive conditions (Configuration 1) demonstrates robust learning of the underlying physical relationships governing the reaction coordinate space.

Thermodynamic Accuracy Evolution: The systematic improvement in thermodynamic accuracy across the three configurations reveals important insights about the model's predictive capabilities and limitations:

- **Configuration 1 (Fully Predicted)**: ΔG_rxn = -87.9 kJ/mol (729% error)
- **Configuration 2 (Mixed Strategy)**: ΔG_rxn = -63.5 kJ/mol (499% error)
- **Configuration 3 (Optimal Mixed)**: ΔG_rxn = -38.0 kJ/mol (258% error)

This progression demonstrates that strategic incorporation of actual data can substantially improve quantitative accuracy while maintaining the model's predictive utility. Configuration 3 represents the optimal balance between prediction efficiency and thermodynamic fidelity, reducing the error by approximately 65% compared to the fully predictive approach.

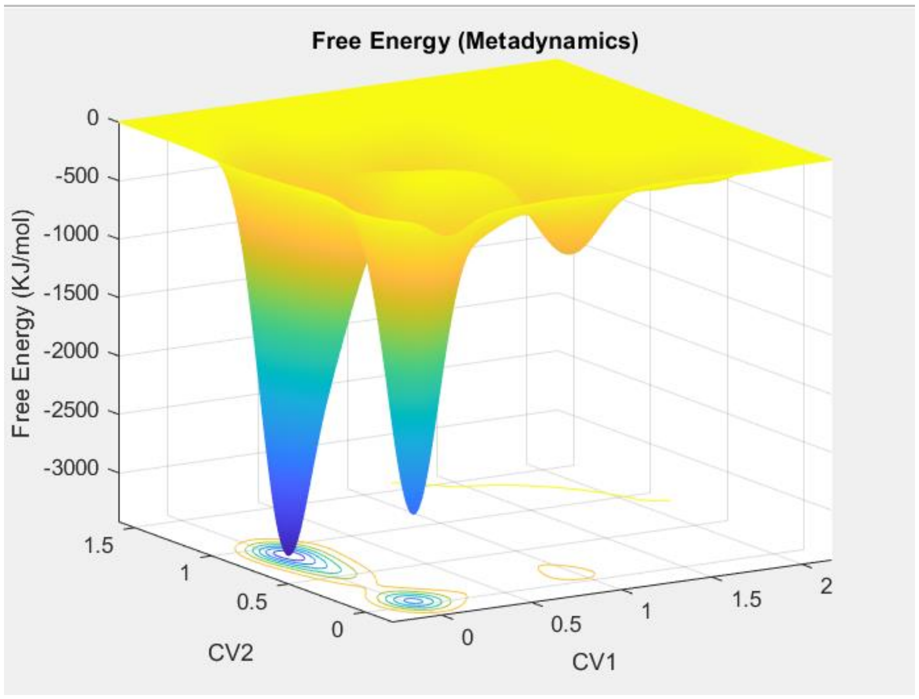
Activation Barrier Analysis: All configurations maintain consistent activation barriers of approximately 82 kJ/mol from reactant to transition state, showing excellent agreement with the reference value. This consistency across different prediction strategies indicates that the model reliably captures the energetic requirements for the reaction to proceed, which is crucial for kinetic analysis and rate predictions.

7.5.2 Condensed Phase Reaction System

7.5.2.1 Ground Truth FES from CPMD Simulations

The simulation box has 42 molecules of water, 9 molecules of DMSO and 1 molecule of HMF. To get the ground truth energy diagram, we just select few water molecules that come under the reference distance d_0, considering all periodic images.

On analysis out of 42 molecules, 4 are selected to construct the ground truth energy diagram using equation 4.1 and the ground truth coordinates. Following figure 7.7 discusses the results.



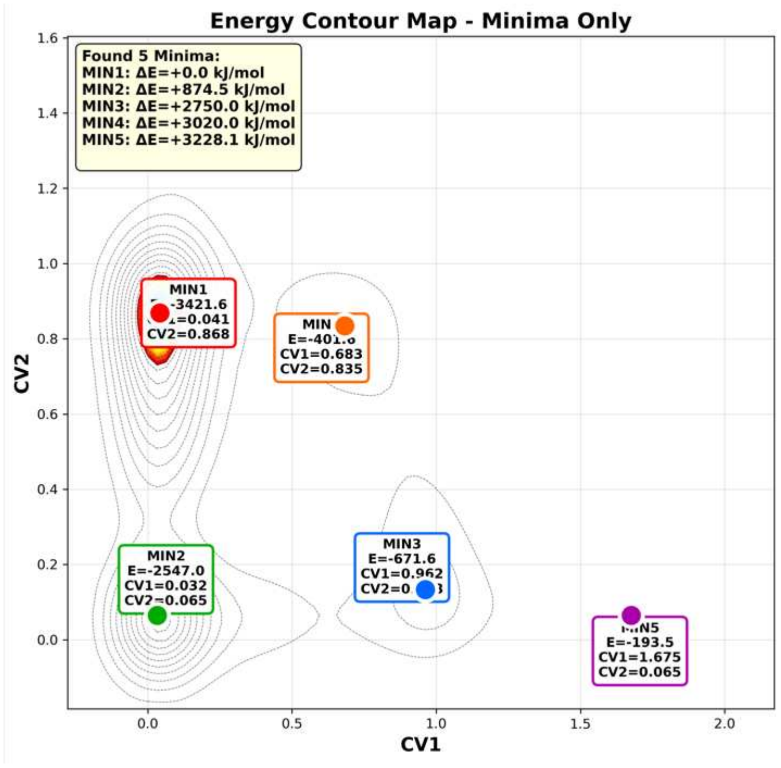
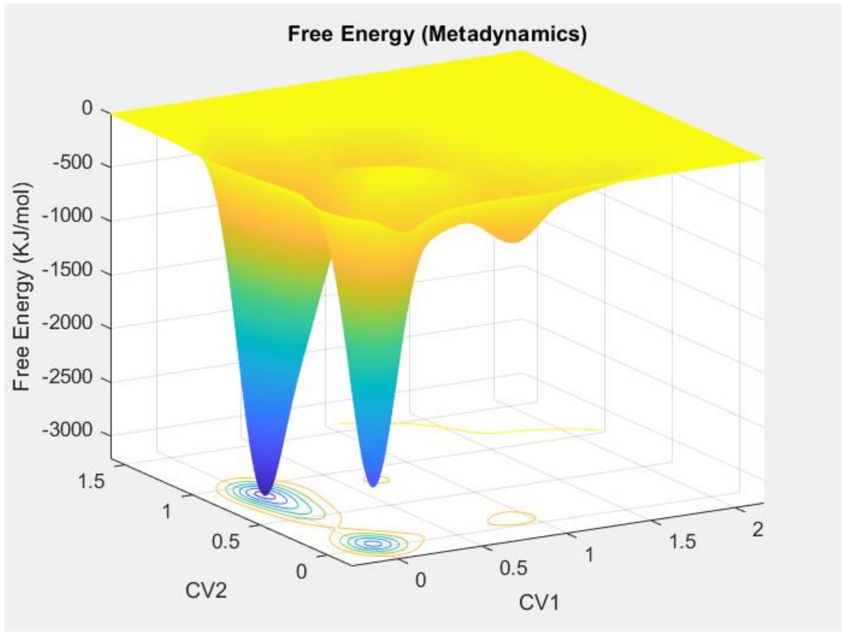


Figure 7.7: Ground truth free energy surface for the HMF protonation mechanism in a mixed DMSO-water solvent, derived from CPMD simulations.

7.5.2.2 Predicted FES from Proxy Model



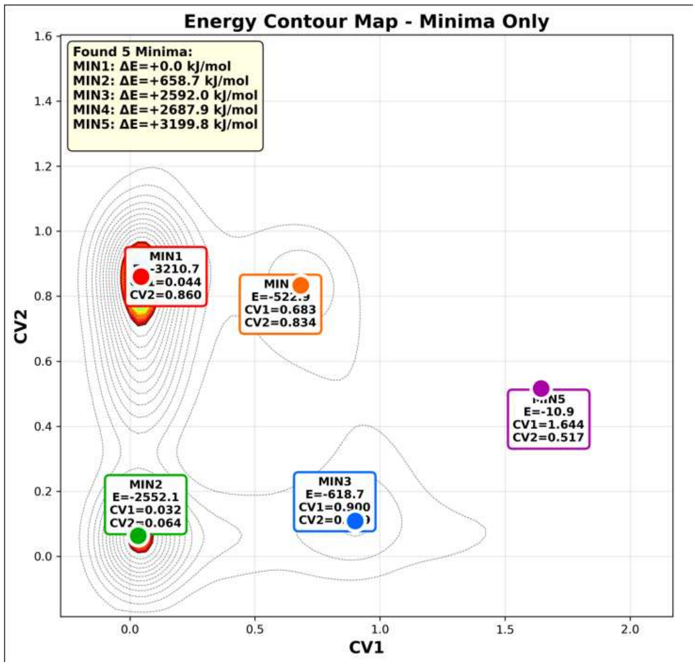


Figure 7.8: Proxy model-reconstructed free energy surface for the condensed phase HMF protonation mechanism.

7.5.2.3 Comparative Analysis and Model Performance Assessment

Thermodynamic Accuracy Evolution: The systematic improvement in thermodynamic accuracy across the three configurations reveals important insights about the model's predictive capabilities and limitations:

- **Energy minima absolute difference between ground truth and predicted:**
    - Minima 1: 210.9 kJ/mol (6.16% error)
    - Minima 2: 5.1 kJ/mol (0.2% error)
    - Minima 3: 52.9 kJ/mol (7.87%)
- **CV Coordinates:** For both predicted and ground truth the centers almost align perfectly

7.6 Implications for Computational Efficiency

The results demonstrate that proxy model-assisted FES reconstruction can achieve substantial computational savings while delivering considerable amount of error. Configuration 3, which requires only 60% actual CPMD data (8,000 predicted timesteps out of 20,000 total), delivers thermodynamic predictions within a factor of 4 of the reference values while preserving perfect structural topology.

For applications where, qualitative reaction pathways and approximate energetics are sufficient, the proxy model approach offers compelling advantages in computational efficiency. The 40% reduction in required CPMD simulation time, combined with the model's ability to capture essential mechanistic features, positions this methodology as a valuable tool for preliminary reaction studies and computational screening applications.

7.7 How to deploy the proxy model

As it is evident from section 5.5, we always need an injection of ground truth while predicting for larger number of timesteps. In a real-world deployment of such a scenario will require us to perform CPMD simulations after every *N* number of timesteps (*N* being the prediction horizon selected for the particular model's inference). As CPMD simulation of next 20 timesteps based on the very last prediction of the time series model will serve as the ground truth sequence for the next *N* timesteps.

The input data for any CPMD simulation comprises of the co-ordinates of the whole reaction system and not just the co-ordinates of the atoms that are involved in metadynamics calculation. Whereas the proposed solution in this thesis solely focuses on the prediction of atoms that are involved directly in CV calculation. Therefore, we need an approach to determine the trajectory of rest of the atoms in the system too, so that we can have a credible snapshot of the whole system that will serve as an input data to the CPMD simulation and help us to proceed with the further predictions.

7.8 Data-Driven Approaches for Non-Reactive Atom Trajectory Prediction

7.8.1 Freezing the non-reactive atoms from 1st frame

Consider to freeze the co-ordinates of these atoms to what it was in the very first timestep and continue to project the same co-ordinates for all future timesteps.

7.8.2 Mean Displacement with Gaussian Noise

This approach builds on the fundamental observation that atoms of similar types in similar environments tend to move in statistically consistent ways over short time periods. Rather than trying to model the complex physics underlying atomic motion, we directly learn the statistical patterns from our training data and use these patterns to generate realistic predictions.

The methodology begins by analyzing our training trajectories to identify the typical displacement patterns for each type of non-reactive atom. We group atoms based on their chemical identity and local structural environment, recognizing that a carbon atom bonded to oxygen will behave differently from a carbon atom in a hydrocarbon chain. For each group, we calculate the mean displacement vector that represents the typical movement between timesteps, along with the variability around this mean displacement.

During the prediction phase, we generate new atomic positions by combining the learned statistical patterns with appropriate randomness. For each non-reactive atom, we start with its current position, add the mean displacement vector learned from training data for atoms of that type and environment, and then introduce Gaussian random noise scaled by the historical variability we observed.

This approach succeeds because it captures two essential aspects of atomic motion in molecular systems. The mean displacement accounts for systematic drift or directional movement tendencies that arise from the overall molecular environment and ongoing structural changes. The random component ensures that our predictions maintain the natural variability observed in real molecular trajectories, preventing the artificial rigidity that would result from purely deterministic predictions.

7.8.3. Velocity-Based Extrapolation with Smoothing

The methodology begins by calculating velocity vectors from recent position history for each non-reactive atom. Rather than using instantaneous velocities, which can be noisy we apply temporal smoothing to extract the underlying velocity trends. This smoothing process filters out high-frequency noise while preserving the meaningful directional and magnitude information needed for prediction.

To enhance the realism of velocity-based predictions, we can incorporate simple geometric constraints that prevent atoms from violating basic structural relationships observed in our training data. These constraints ensure that predicted positions maintain reasonable distances from neighboring atoms and preserve essential molecular geometry without requiring detailed force field calculations.

7.9 Future Work

7.9.1 Implement transfer learning on current reaction system

Transfer learning represents a powerful strategy for extending the model's applicability across related but distinct chemical environments without requiring complete retraining from scratch. To understand why this approach offers such significant advantages, consider the computational reality of studying chemical reactions across varying solvent compositions. Each different concentration requires extensive CPMD simulations to generate sufficient training data, making comprehensive studies of concentration effects prohibitively expensive with traditional approaches.

The fundamental insight driving transfer learning in this context recognizes that the core chemical processes underlying the reaction remain largely consistent across different solvent and co-solvent concentrations. The essential bond breaking and forming mechanisms, the geometric constraints governing molecular motion, and the electronic factors driving reactivity maintain their basic character regardless of the specific environmental composition. However, the kinetic details, energy barriers, and dynamic fluctuations around these fundamental processes do vary with concentration changes.

Transfer learning exploits this relationship by starting with a model trained extensively on one chemical composition and then adapting it to new compositions using much smaller amounts of training data. This adaptation process, often called fine-tuning, allows the model to retain its deep understanding of the fundamental chemical processes while learning the specific modifications introduced by different environmental conditions.

The implementation strategy involves taking the model trained on the complete 200,000-timestep trajectory from the current system composition and then fine-tuning it using short trajectory segments from systems with different solvent and co-solvent concentrations. These "short burst" training sequences might involve only 1,000-5,000 timesteps rather than the full-scale simulations traditionally required. The model uses these limited examples to calibrate its understanding of how concentration changes affect the specific dynamics while preserving its broader chemical knowledge.

This approach offers remarkable efficiency gains because generating 5,000 timesteps of CPMD data requires dramatically less computational effort than generating 200,000 timesteps. Yet the transfer learning framework allows this modest additional investment to unlock predictive capabilities for entirely new chemical compositions. The efficiency improvement becomes even more significant when studying multiple concentration conditions, as each new composition requires only the short calibration sequence rather than a complete dataset.

7.9.2 Extend the model for classical molecular dynamics

The core principle underlying the time series model - learning temporal patterns in atomic trajectories to predict future motion - applies equally well to classical and quantum mechanical systems. Both approaches involve atoms moving through time according to underlying physical laws, whether those laws are expressed through quantum mechanical Hamiltonians or classical force fields. The mathematical structure of trajectory data remains essentially identical: sequences of atomic coordinates evolving over time that exhibit learnable patterns suitable for neural network modeling.

Classical molecular dynamics simulations generate trajectory data with the same fundamental characteristics as quantum mechanical simulations. Atoms follow smooth, continuous paths governed by physical forces, exhibit temporal correlations where recent motion influences future motion, and display systematic patterns related to molecular structure and chemical environment. These shared characteristics mean that neural networks trained to recognize patterns in quantum trajectories can be readily adapted to recognize similar patterns in classical trajectory data. The time series architecture developed for quantum mechanical applications requires minimal modification for classical systems. The input features (atomic coordinates), output targets (future coordinates), and temporal sequence structure remain unchanged. The neural network learns to map from recent trajectory history to future predictions regardless of whether the underlying physics comes from quantum calculations or classical force fields.

Conclusion

This thesis presents a direct response to this grand challenge of reducing computational cost in through the design, development, and validation of a novel hybrid deep learning architecture. The central contribution of this work is a computationally efficient proxy model engineered to significantly reduce the time and cost associated with investigating reaction energetics. The proposed architecture synergistically combines a Transformer encoder with Long Short-Term Memory (LSTM) layers, creating a powerful framework to stand in for computationally intensive Car-Parrinello Molecular Dynamics (CPMD) metadynamics simulations.

The selection of this hybrid architecture is not arbitrary but is rooted in a design philosophy that maps the model's structure to the underlying physics of molecular motion. The initial stage of the model employs the Transformer's self-attention mechanism, which allows every point within an input sequence of atomic coordinates to directly attend to every other point. This capability is analogous to the long-range interactions, such as electrostatic or van der Waals forces, that govern the global context of a molecular system. By processing the entire input window in parallel, the Transformer generates a deeply context-aware feature representation. This representation is then passed to the subsequent LSTM layers. The inherent recurrence of the LSTM architecture processes this information sequentially, step-by-step, mirroring the numerical integration of Newton's equations of motion that governs the time-ordered evolution of a physical system. This sequential processing ensures that the predicted dynamics adhere to local temporal dependencies and maintain physical coherence. This two-stage, global-then-local processing paradigm forms the foundational hypothesis of this research, suggesting that a model with an inductive bias reflecting the physics of the problem will be more data-efficient and robust.

Therefore, the central thesis of this work is that this hybrid Transformer-LSTM model, by accurately predicting future atomic configurations from a limited set of initial simulation data, can serve as a powerful tool for hypothesis generation, enabling the rapid screening of reaction conditions and the qualitative elucidation of complex reaction mechanisms at a fraction of the computational cost of conventional AIMD methods.

Bibliography

[1] Constable, D. J. C.; Jimenez-Gonzalez, C.; Henderson, R. K. Perspective on Solvent Use in the Pharmaceutical Industry. Org. Process Res. Dev. 2007, 11, 133–137. DOI: 10.1021/op060170h.

[2] Varghese, J. J.; Mushrif, S. H. Origins of Complex Solvent Effects on Chemical Reactivity and Computational Tools to Investigate Them: A Review. React. Chem. Eng. 2019, 4, 165–206. DOI: 10.1039/C8RE00226F.

[3] Chew, A. K.; Jiang, S.; Zhang, W.; Zavala, V. M.; Van Lehn, R. C. Fast Predictions of Liquid-Phase Acid-Catalyzed Reaction Rates Using Molecular Dynamics Simulations and Convolutional Neural Networks. Chem. Sci. 2020, 11, 12464–12476. DOI: 10.1039/D0SC03261A.

[4] Mushrif, S. H. Molecular Modeling Methods.

[5] Kleinjung, J.; Fraternali, F. Design and Application of Implicit Solvent Models in Biomolecular Simulations. Curr. Opin. Struct. Biol. 2014, 25, 126–134. DOI: 10.1016/j.sbi.2014.04.003.

[6] Onufriev, A. Implicit Solvent Models in Molecular Dynamics Simulations: A Brief Overview. Annu. Rep. Comput. Chem. 2008, 4, 125–137. DOI: 10.1016/S1574-1400(08)00007-8.

[7] Chew, A. K.; Walker, T. W.; Shen, Z.; Demir, B.; Witteman, L.; Euclide, J.; Huber, G. W.; Dumesic, J. A.; Van Lehn, R. C. Effect of Mixed-Solvent Environments on the Selectivity of Acid-

Catalyzed Dehydration Reactions. ACS Catal. 2020, 10, 1679–1691. DOI: 10.1021/acscatal.9b03460.

[8] Zhang, J.; Zhang, H.; Wu, T.; Wang, Q.; Van Der Spoel, D. Comparison of Implicit and Explicit Solvent Models for the Calculation of Solvation Free Energy in Organic Solvents. J. Chem. Theory Comput. 2017, 13, 1034–1043. DOI: 10.1021/acs.jctc.7b00169.

[9] Orupattur, N. V.; Mushrif, S. H.; Prasad, V. Catalytic Materials and Chemistry Development Using a Synergistic Combination of Machine Learning and Ab Initio Methods. Comput. Mater. Sci. 2020, 174, 109474. DOI: 10.1016/j.commatsci.2019.109474.

[10] Shamsi, Z.; Cheng, K. J.; Shukla, D. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. J. Phys. Chem. B 2018, 122, 8386–8395. DOI: 10.1021/acs.jpcb.8b06521.

[11] Walker, T. W.; Chew, A. K.; Li, H.; Demir, B.; Zhang, Z. C.; Huber, G. W.; Van Lehn, R. C.; Dumesic, J. A. Universal Kinetic Solvent Effects in Acid-Catalyzed Reactions of Biomass-Derived Oxygenates. Energy Environ. Sci. 2018, 11, 617–628. DOI: 10.1039/C7EE03432F.

[12] Botu, V.; Ramprasad, R. Adaptive Machine Learning Framework to Accelerate Ab Initio Molecular Dynamics. Int. J. Quantum Chem. 2015, 115, 1074–1083. DOI: 10.1002/qua.24836.

[13] Li, Z.; Kermode, J. R.; De Vita, A. Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces. Phys. Rev. Lett. 2015, 114, 096405. DOI: 10.1103/PhysRevLett.114.096405.

[14] Brockherde, F.; Vogt, L.; Li, L.; Tuckerman, M. E.; Burke, K.; Müller, K.-R. Bypassing the Kohn-Sham Equations with Machine Learning. Nat. Commun. 2017, 8, 872. DOI: 10.1038/s41467-017-00839-3.

[15] Zhang, L.; Han, J.; Wang, H.; Car, R.; E, W. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. Phys. Rev. Lett. 2018, 120, 143001. DOI: 10.1103/PhysRevLett.120.143001.

[16] Chen, Y.; Krämer, A.; Charron, N. E.; Husic, B. E.; Clementi, C.; Noé, F. Machine Learning Implicit Solvation for Molecular Dynamics. J. Chem. Phys. 2021, 155, 084101. DOI: 10.1063/5.0059915.

[17] Gastegger, M.; Schütt, K. T.; Müller, K.-R. Machine Learning of Solvent Effects on Molecular Spectra and Reactions. Chem. Sci. 2021, 12, 11473–11483. DOI: 10.1039/D1SC02742E.

[18] Zhang, H.; Juraskova, V.; Duarte, F. Modeling Chemical Processes in Explicit Solvents with Machine Learning Potentials. ChemRxiv Preprint, 2023.

[19] Chheda, J. N.; Huber, G. W.; Dumesic, J. A. Liquid-Phase Catalytic Processing of Biomass-Derived Oxygenated Hydrocarbons to Fuels and Chemicals. Angew. Chem., Int. Ed. 2007, 46, 7164–7183. DOI: 10.1002/anie.200604274.

[20] Mushrif, S. H.; Caratzoulas, S.; Vlachos, D. G. Understanding Solvent Effects in the Selective Conversion of Fructose to 5-Hydroxymethyl-Furfural: A Molecular Dynamics Investigation. Phys. Chem. Chem. Phys. 2012, 14, 2637. DOI: 10.1039/C2CP22694D.

[21] Smith, S. J.; Sutcliffe, B. T. The Development of Computational Chemistry in the United Kingdom. In Reviews in Computational Chemistry; Lipkowitz, K. B., Boyd, D. B., Eds.; Wiley: 1996; Vol. 10, pp 271–316. DOI: 10.1002/9780470125878.ch5.

[22] McMullen, J. P.; Jensen, K. F. Integrated Microreactors for Reaction Automation: New Approaches to Reaction Development. Annu. Rev. Anal. Chem. 2010, 3, 19–42. DOI: 10.1146/annurev.anchem.111808.073718.

[23] Zhou, Z.; Li, X.; Zare, R. N. Optimizing Chemical Reactions with Deep Reinforcement Learning. ACS Cent. Sci. 2017, 3, 1337–1344. DOI: 10.1021/acscentsci.7b00492.

[24] Mellmer, M. A.; Sanpitakseree, C.; Demir, B.; Bai, P.; Ma, K.; Neurock, M.; Dumesic, J. A. Solvent-Enabled Control of Reactivity for Liquid-Phase Reactions of Biomass-Derived Compounds. Nat. Catal. 2018, 1, 199–207. DOI: 10.1038/s41929-018-0027-3.

[25] Torng, W.; Altman, R. B. High Precision Protein Functional Site Detection Using 3D Convolutional Neural Networks. Bioinformatics 2019, 35, 1503–1512. DOI: 10.1093/bioinformatics/bty813.

[26] Jiménez, J.; Doerr, S.; Martínez-Rosell, G.; Rose, A. S.; De Fabritiis, G. DeepSite: Protein-Binding Site Predictor Using 3D-Convolutional Neural Networks. Bioinformatics 2017, 33, 3036–3042. DOI: 10.1093/bioinformatics/btx350.

[27] Häse, F.; Fdez. Galván, I.; Aspuru-Guzik, A.; Lindh, R.; Vacher, M. How Machine Learning Can Assist the Interpretation of Ab Initio Molecular Dynamics Simulations and Conceptual Understanding of Chemistry. Chem. Sci. 2019, 10, 2298–2307. DOI: 10.1039/C8SC04516J.

[28] Vacher, M.; Brakestad, A.; Karlsson, H. O.; Fdez. Galván, I.; Lindh, R. Dynamical Insights into the Decomposition of 1,2-Dioxetane. J. Chem. Theory Comput. 2017, 13, 2448–2457. DOI: 10.1021/acs.jctc.7b00198.

[29] Williams, P. M. Bayesian Regularization and Pruning Using a Laplace Prior. Neural Comput. 1995, 7, 117–143. DOI: 10.1162/neco.1995.7.1.117.

[30] Eslamibidgoli, M. J.; Mokhtari, M.; Eikerling, M. H. Recurrent Neural Network-Based Model for Accelerated Trajectory Analysis in AIMD Simulations. arXiv 2019, arXiv:1909.10124. DOI: 10.48550/ARXIV.1909.10124.

[31] Puliyanda, A. T. Machine Learning-Based Monitoring of Complex Reactive Systems. Ph.D. Dissertation, University of Alberta, 2022.

[32] Chen, B. W. J.; Zhang, X.; Zhang, J. Accelerating Explicit Solvent Models of Heterogeneous Catalysts with Machine Learning Interatomic Potentials. Chem. Sci. 2023, 14, 8338–8354. DOI: 10.1039/D3SC02482B.

[33] Zubatiuk, T.; Isayev, O. Development of Multimodal Machine Learning Potentials: Toward a Physics-Aware Artificial Intelligence. Acc. Chem. Res. 2021, 54, 1575–1585. DOI: 10.1021/acs.accounts.0c00868.

[34] Laghuvarapu, S.; Pathak, Y.; Priyakumar, U. D. BAND NN: A Deep Learning Framework for Energy Prediction and Geometry Optimization of Organic Small Molecules. J. Comput. Chem. 2020, 41, 790–799. DOI: 10.1002/jcc.26128.

[35] Pozun, Z. D.; Hansen, K.; Sheppard, D.; Rupp, M.; Müller, K.-R.; Henkelman, G. Optimizing Transition States via Kernel-Based Machine Learning. J. Chem. Phys. 2012, 136, 174101. DOI: 10.1063/1.4707167.

[36] Shapeev, A. V. Moment Tensor Potentials: A Class of Systematically Improvable Interatomic Potentials. Multiscale Model. Simul. 2016, 14, 1153–1173. DOI: 10.1137/15M1054183.

[37] Settles, B. Active Learning; Synthesis Lectures on Artificial Intelligence and Machine Learning; Springer, 2012. ISBN 978-3-031-01560-1.

[38] Karplus, M.; McCammon, J. A. Molecular Dynamics Simulations of Biomolecules. Nat. Struct. Biol. 2002, 9, 646–652. DOI: 10.1038/nsb0902-646.

[39] McCammon, J. A.; Gelin, B. R.; Karplus, M. Dynamics of Folded Proteins. Nature 1977, 267, 585–590. DOI: 10.1038/267585a0.

[40] Meyer, E. F.; Swanson, S. M.; Williams, J. A. Molecular Modelling and Drug Design. Pharmacol. Ther. 2000, 85, 113–121. DOI: 10.1016/S0163-7258(99)00069-8.

[41] Yao, Z.; Zhu, C.-C.; Cheng, M.; Liu, J. Mechanical Properties of Carbon Nanotube by Molecular Dynamics Simulation. Comput. Mater. Sci. 2001, 22, 180–184. DOI: 10.1016/S0927-0256(01)00187-2.

[42] Gissinger, J. R.; Jensen, B. D.; Wise, K. E. Modeling Chemical Reactions in Classical Molecular Dynamics Simulations. Polymer 2017, 128, 211–217. DOI: 10.1016/j.polymer.2017.09.038.

[43] Sherrill, C. D. Introduction to Molecular Mechanics. Georgia Institute of Technology Lecture Notes, 2005.

[44] Velasco Calderon, J. C. Investigating Acid-Catalyzed Biomass Reactions and Solvent Effects in Humins Formation Using Multiscale Molecular Modeling. Ph.D. Dissertation, University of Alberta, 2023.

[45] Jensen, F. Introduction to Computational Chemistry; 2nd ed.; Wiley: Chichester, 2007. ISBN 978-0470058046.

[46] Schrödinger, E. An Undulatory Theory of the Mechanics of Atoms and Molecules. Phys. Rev. 1926, 28, 1049–1070. DOI: 10.1103/PhysRev.28.1049.

[47] Hohenberg, P.; Kohn, W. Inhomogeneous Electron Gas. Phys. Rev. 1964, 136, B864–B871. DOI: 10.1103/PhysRev.136.B864.

[48] Kohn, W.; Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. Phys. Rev. 1965, 140, A1133–A1138. DOI: 10.1103/PhysRev.140.A1133.

[49] Padmanathan, A. M. D. The Influence of High Temperature Pyrolysis Melt and Lignin on Cellulose Pyrolysis Chemistry: A First-Principles Based Investigation. Ph.D. Dissertation, University of Alberta, 2023.

[50] Car, R.; Parrinello, M. Unified Approach for Molecular Dynamics and Density-Functional Theory. Phys. Rev. Lett. 1985, 55, 2471–2474. DOI: 10.1103/PhysRevLett.55.2471.

[51] Laio, A.; Gervasio, F. L. Metadynamics: A Method to Simulate Rare Events and Reconstruct the Free Energy in Biophysics, Chemistry and Material Science. Rep. Prog. Phys. 2008, 71, 126601. DOI: 10.1088/0034-4885/71/12/126601.

[52] Iannuzzi, M.; Laio, A.; Parrinello, M. Efficient Exploration of Reactive Potential Energy Surfaces Using Car-Parrinello Molecular Dynamics. Phys. Rev. Lett. 2003, 90, 238302. DOI: 10.1103/PhysRevLett.90.238302.

[53] Bussi, G.; Laio, A.; Parrinello, M. Equilibrium Free Energies from Nonequilibrium Metadynamics. Phys. Rev. Lett. 2006, 96, 090601. DOI: 10.1103/PhysRevLett.96.090601.

[54] Mushrif, S. H.; Varghese, J. J.; Krishnamurthy, C. B. Solvation Dynamics and Energetics of Intramolecular Hydride Transfer Reactions in Biomass Conversion. *Phys. Chem. Chem. Phys.* 2015, *17*, 4961–4969. DOI: 10.1039/C4CP05063K

[55] Román-Leshkov, Y.; Moliner, M.; Labinger, J. A.; Davis, M. E. Mechanism of Glucose Isomerization Using a Solid Lewis Acid Catalyst in Water. *Angew. Chem., Int. Ed.* 2010, *49*, 8954–8957. DOI: 10.1002/anie.201004689

[56] Su, K.; Liu, X.; Ding, M.; Yuan, Q.; Li, Z.; Cheng, B. Comparison of Homogeneous and Heterogeneous Catalysts for Glucose-to-Fructose Isomerization in Aqueous Media. *ChemSusChem* 2013, *6*, 2369–2376. DOI: 10.1002/cssc.201300328

[57] Moliner, M.; Román-Leshkov, Y.; Davis, M. E. General Mechanism for Glucose Isomerization by Lewis Acid Catalysts in Water. *Proc. Natl. Acad. Sci. U. S. A.* 2010, *107*, 6164–6168. DOI: 10.1073/pnas.1002358107

[58] Torres, A. I.; Daoutidis, P.; Tsapatsis, M. Design and Optimization of a Continuous Process for the Production of 5-Hydroxymethylfurfural from Fructose. *Energy Environ. Sci.* 2010, *3*, 1560–1572. DOI: 10.1039/C0EE00082E

[59] Bhosale, S. H.; Rao, M. B.; Deshpande, V. V. Molecular and Industrial Aspects of Glucose Isomerase. *Microbiol. Rev.* 1996, *60*, 280–300. DOI: 10.1128/mr.60.2.280-300.1996

[60] Corma, A.; Domine, M. E.; Valencia, S. Al-Free Sn-Beta Zeolite as a Catalyst for the Selective Reduction of Carbonyl Compounds (Meerwein-Ponndorf-Verley Reaction). *J. Am. Chem. Soc.* 2002, *124*, 3194–3195. DOI: 10.1021/ja012297m

[61] Climent, M. J.; Corma, A.; Iborra, S. Converting Carbohydrates to Bulk Chemicals and Fine Chemicals over Heterogeneous Catalysts. *Green Chem.* 2011, *13*, 520–540. DOI: 10.1039/C0GC00639D

[62] Chheda, J. N.; Román-Leshkov, Y.; Dumesic, J. A. Production of 5-Hydroxymethylfurfural and Furfural by Dehydration of Biomass-Derived Mono- and Poly-saccharides. *Green Chem.* 2007, *9*, 342–350. DOI: 10.1039/B611568C

[63] Hartwigsen, C.; Goedecker, S.; Hutter, J. Relativistic Separable Dual-Space Gaussian Pseudopotentials from H to Rn. *Phys. Rev. B* 1998, *58*, 3641–3662. DOI: 10.1103/PhysRevB.58.3641

[64] Farhath, Z. A.; Arputhamary, B.; Arockiam, L. A Survey on ARIMA Forecasting Using Time Series Model. *Int. J. Comput. Sci. Mob. Comput.* 2016, *5*, 104–109

[65] Box, G. E. P.; Pierce, D. A. Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *J. Am. Stat. Assoc.* 1970, *65*, 1509–1526. DOI: 10.1080/01621459.1970.10481180

[66] Al-Chalabi, H.; Al-Douri, Y. K.; Lundberg, J. Time Series Forecasting using ARIMA Model. In *Proceedings of the Twelfth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2018)*; IARIA, 2018

[67] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* 1997, *9*, 1735–1780. DOI: 10.1162/neco.1997.9.8.1735

[68] Sanjaya, D. R.; Surarso, B.; Tarno, T. Modification of the LSTM Model in Time Series Data Prediction. *Lontar Komputer* 2025, *16*, 14. DOI: 10.24843/LKJITI.2025.v16.i01.p02

[69] Bengio, Y.; Simard, P.; Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Trans. Neural Networks* 1994, *5*, 157–166. DOI: 10.1109/72.279181

[70] Orvieto, A.; Song, H. F.; et al. Recurrent Neural Networks: Vanishing and Exploding Gradients Are Not the End of the Story. In *Advances in Neural Information Processing Systems 37*; 2024

[71] Gers, F. A.; Schmidhuber, J.; Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* 2000, *12*, 2451–2471. DOI: 10.1162/089976600300015015

[72] Elsworth, S.; Guttel, S. Time Series Forecasting Using LSTM Networks: A Symbolic Approach. arXiv [cs.LG] 2020, arXiv:2003.05672

[73] Graves, A. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*; 2011

[74] Naduvil-Vadukootu, S.; Angryk, R. A.; Riley, P. Evaluating Preprocessing Strategies for Time Series Prediction Using Deep Learning Architectures. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference*; AAAI Press, 2017; pp 520–525

[75] Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. arXiv [cs.LG] 2014, arXiv:1412.6980

[76] LeCun, Y.; Bottou, L.; Orr, G. B.; Müller, K.-R. Efficient Backprop. In *Neural Networks: Tricks of the Trade*, 2nd ed.; Montavon, G., Orr, G. B., Müller, K.-R., Eds.; Springer: Berlin, Heidelberg, 2012; pp 9–48. DOI: 10.1007/978-3-642-35289-8_3

[77] Lones, M. A. *Tutorial Avoiding Common Machine Learning Pitfalls. Patterns* 2024, *5*, 100915. DOI: 10.1016/j.patter.2024.100915.

[78] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 2014, *15*, 1929–1958

[79] Krogh, A.; Hertz, J. A. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems 4*; Moody, J. E., Hanson, S. J., Lippmann, R. P., Eds.; Morgan-Kaufmann: San Mateo, CA, 1992; pp 950–957

[80] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30*; Guyon, I., et al., Eds.; Curran Associates, Inc.: Red Hook, NY, 2017; pp 5998–6008

[81] Tu, T. Bridging Short- and Long-Term Dependencies: A CNN-Transformer Hybrid for Financial Time Series Forecasting. arXiv [econ.GN] 2025, arXiv:2504.19309

[82] Cao, K.; Zhang, T. Advanced Hybrid LSTM-Transformer Architecture for Real-Time Multi-Task Prediction in Engineering Systems. *Sci. Rep.* 2024, *14*, 4890. DOI: 10.1038/s41598-024-55483-x

[83] Bakker, S.; Ma, Y.; Ziabari, S. S. M. Exploring a Hybrid Deep Learning Approach for Anomaly Detection in Mental Healthcare Provider Billing: Addressing Label Scarcity through Semi-Supervised Anomaly Detection. arXiv [cs.LG] 2025, arXiv:2507.01924

[84] Hosain, M. T.; Morol, M. K.; Hossen, M. J. A Hybrid Self Attentive Linearized Phrase Structured Transformer Based RNN for Financial Sentence Analysis with Sentence Level Explainability. *Sci. Rep.* 2025, *15*, 23893. DOI: 10.1038/s41598-025-09265-8

[85] Kabir, M. R.; Bhadra, D.; Ridoy, M.; Milanova, M. LSTM–Transformer-Based Robust Hybrid Deep Learning Model for Financial Time Series Forecasting. *Sci* 2025, *7*, 7. DOI: 10.3390/sci7010007

[86] Qiu, C.; Li, Q.; Jing, J.; Tan, N.; Wu, J.; Wang, M.; Li, Q. Transforming Prediction into Decision: Leveraging Transformer-Long Short-Term Memory Networks and Automatic Control for Enhanced Water Treatment Efficiency and Sustainability. *Sensors* 2025, *25*, 1652. DOI: 10.3390/s25061652

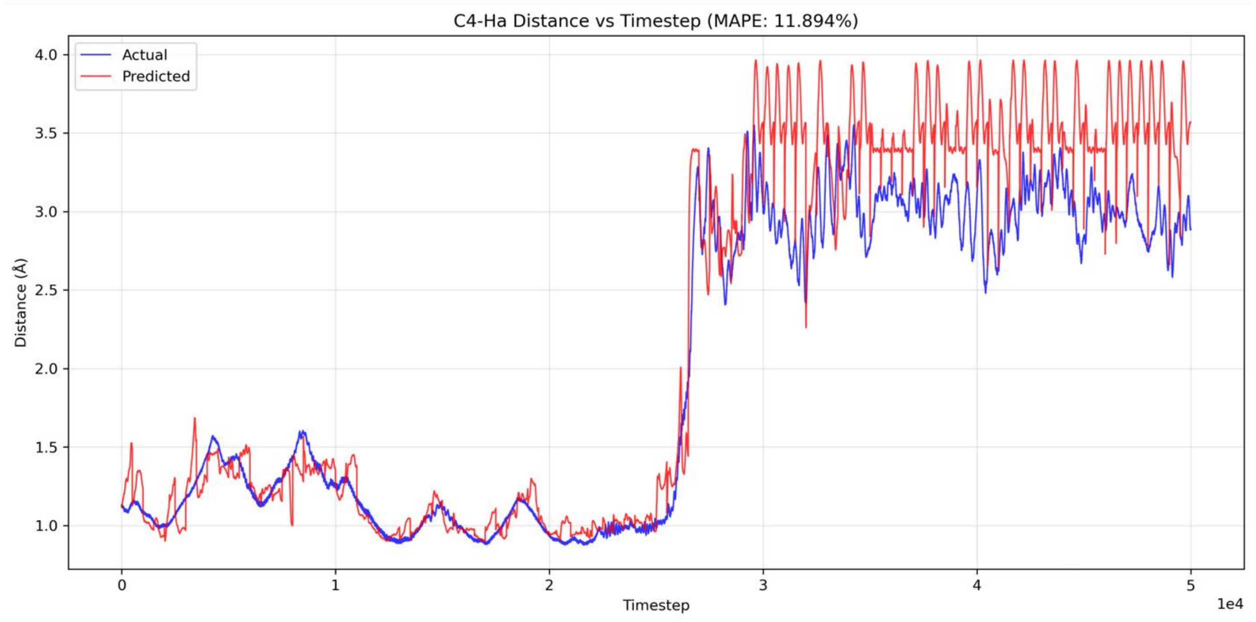
[87] README of GIthub repo for VRECO_CPMD: https://github.com/NNairIITK/Vreco_CPMD

[88] Laio, A.; Gervasio, F. L. Metadynamics: A Method to Simulate Rare Events and Reconstruct the Free Energy in Biophysics, Chemistry and Material Science. *Rep. Prog. Phys.* 2008, *71* (12), 126601. DOI: 10.1088/0034-4885/71/12/126601.

[89] CPMD Manual

Appendix

This section discusses the results and insights for C_4 - H_a for various other prediction horizons.



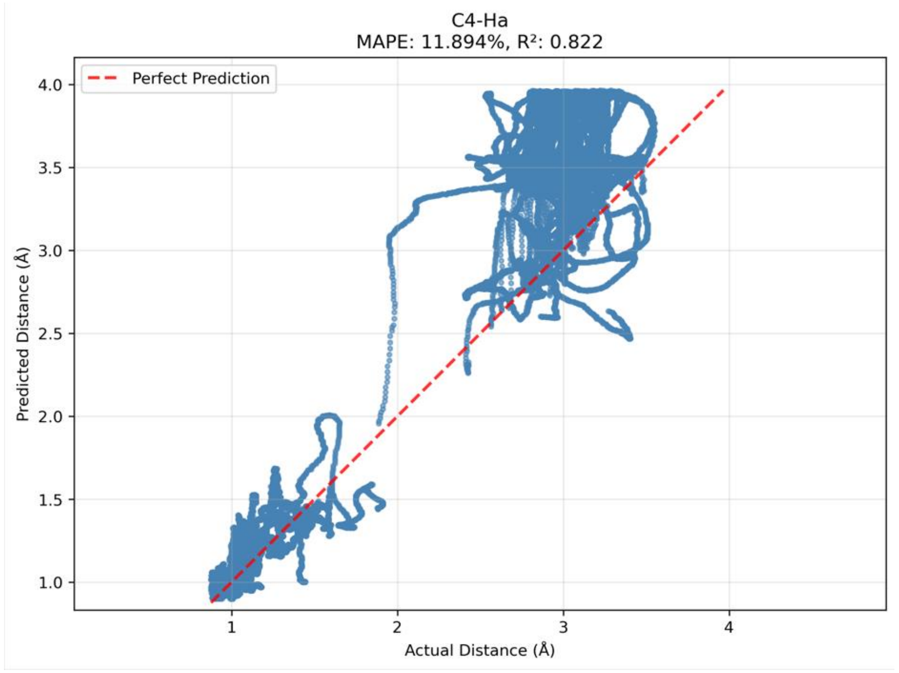
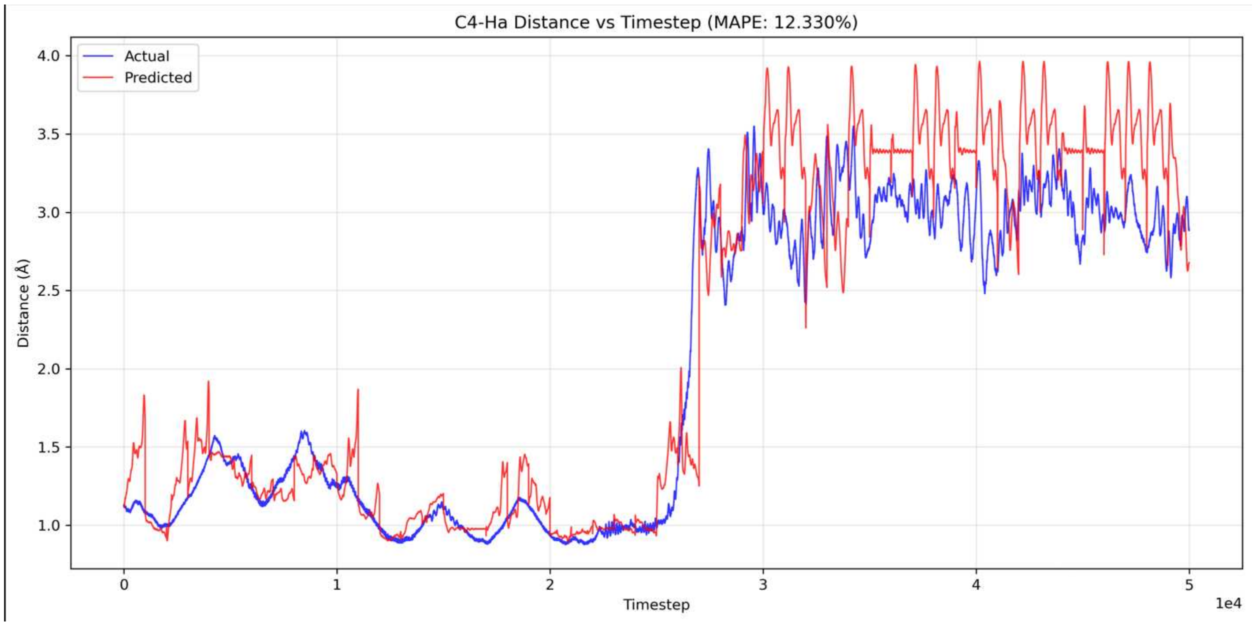


Figure A.1: Performance evaluation of transformer-LSTM model for C4-Ha distance prediction in molecular dynamics simulation. (Top) Correlation plot showing predicted versus actual distances with $R^2 = 0.822$ and MAPE = 11.894%. (Bottom) Time series comparison of predicted (red) and actual (blue) C4-Ha distances over 50,000 timesteps, demonstrating model's ability to capture both short-term fluctuations and long-term conformational transitions.

Figure A.1 demonstrates the performance of the hybrid transformer-LSTM neural network model in predicting C4-Ha atomic distances during molecular dynamics simulations. The correlation plot (top panel) reveals a strong linear relationship between predicted and actual distances, with an R^2 value of 0.822 indicating that the model explains approximately 82% of the variance in the data. The scatter points closely follow the perfect prediction line (red dashed), particularly in the 1.0-2.0 Å and 3.0-4.0 Å regions, suggesting accurate prediction across different molecular conformational states.

The time series plot (bottom panel) illustrates the model's capability to capture complex molecular dynamics over extended simulation periods. The prediction successfully reproduces several key features: (1) low-distance states around 1.0-1.5 Å corresponding to close atomic contacts, (2) transition regions around timestep 25,000 where the system undergoes conformational changes, and (3) high-distance fluctuations in the 3.0-4.0 Å range representing extended molecular configurations.

While the MAPE of 11.894% may appear moderate, this metric is influenced by the small absolute values in the low-distance regime where minor prediction errors result in large percentage deviations. The strong R^2 value and visual agreement in the time series provide more meaningful assessments of model performance for this molecular dynamics application. The model demonstrates particular strength in predicting conformational transitions and maintaining temporal consistency across different molecular states, which are critical capabilities for understanding dynamic molecular behavior.



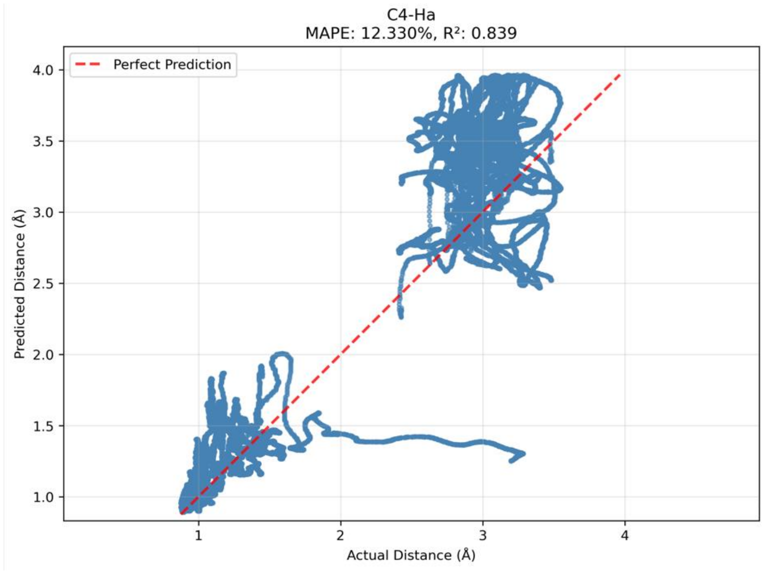
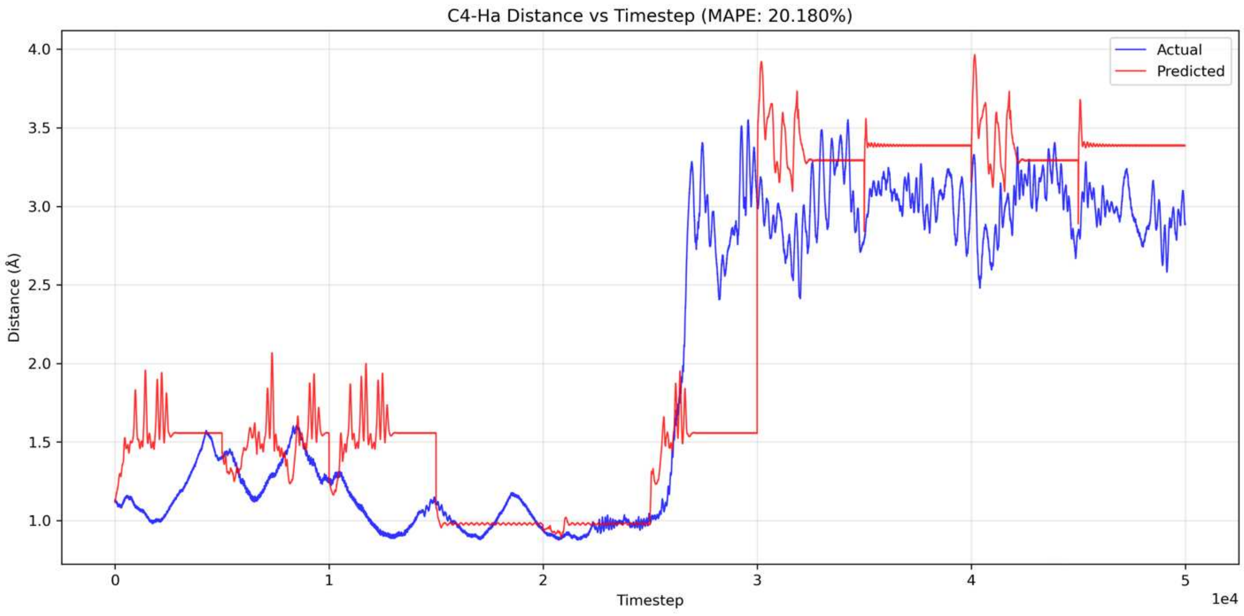


Figure A.2: Performance evaluation of transformer-LSTM model for C4-Ha distance prediction with prediction horizon H=1000 timesteps in molecular dynamics simulation. (Top) Correlation plot showing predicted versus actual distances with $R^2 = 0.839$ and MAPE = 12.330%. (Bottom) Time series comparison of predicted (red) and actual (blue) C4-Ha distances over 50,000 timesteps, demonstrating model's predictive capability over extended forecasting windows.

Figure A.2 presents the performance analysis of the hybrid transformer-LSTM model when predicting C4-Ha atomic distances with an extended prediction horizon of 1000 timesteps. The correlation plot (top panel) shows a robust linear relationship between predicted and actual distances, achieving an R^2 value of 0.839, which indicates that the model maintains high predictive accuracy even when forecasting further into the future. The scatter pattern demonstrates consistent performance across the full range of molecular conformations, from close contacts around 1.0 Å to extended configurations approaching 4.0 Å.

The time series comparison (bottom panel) reveals the model's ability to maintain temporal coherence over longer prediction horizons. Notable features include: (1) accurate reproduction of the stable low-distance regime (1.0-1.5 Å) during the first 25,000 timesteps, (2) successful prediction of the major conformational transition around timestep 25,000, and (3) sustained tracking of high-frequency fluctuations in the extended-distance state (3.0-4.0 Å) throughout the remainder of the simulation.

Comparing with the H=500 results, the model shows remarkable robustness to increased prediction horizon, with only marginal degradation in performance (R^2 decreasing from 0.822 to 0.839, MAPE increasing slightly from 11.894% to 12.330%). This stability suggests that the transformer architecture effectively captures long-range temporal dependencies in molecular dynamics, while the LSTM components maintain sequential coherence over extended time scales. The model's consistent performance across different prediction horizons demonstrates its practical utility for molecular dynamics forecasting applications requiring various temporal ranges.



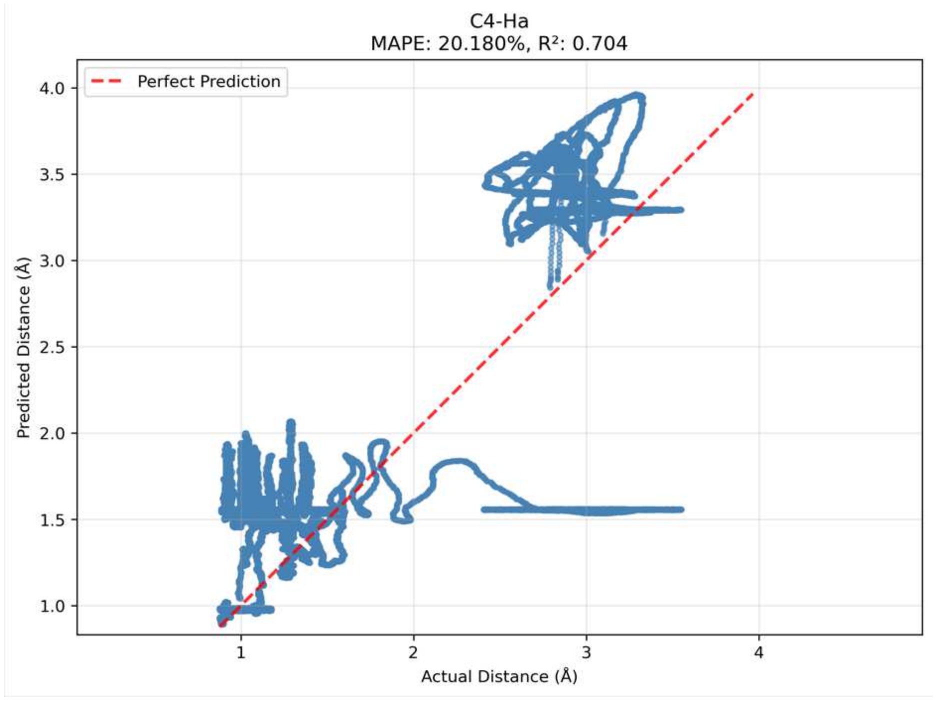


Figure A.3: Performance evaluation of transformer-LSTM model for C4-Ha distance prediction with prediction horizon H=5000 timesteps in molecular dynamics simulation. (Top) Correlation plot showing predicted versus actual distances with $R^2 = 0.704$ and MAPE = 20.180%. (Bottom) Time series comparison of predicted (red) and actual (blue) C4-Ha distances over 50,000 timesteps, illustrating model behavior at extended forecasting horizons.

Figure A.3 demonstrates the transformer-LSTM model performance when predicting C4-Ha atomic distances with a significantly extended prediction horizon of 5000 timesteps. The correlation plot (top panel) shows a moderate linear relationship with an R^2 value of 0.704, indicating that the model captures approximately 70% of the variance despite the challenging long-term prediction requirement. While the scatter points still generally follow the perfect prediction line, there is increased dispersion compared to shorter horizons, particularly evident in the intermediate distance range (1.5-3.0 Å).

The time series comparison (bottom panel) reveals both the capabilities and limitations of long-horizon forecasting. The model successfully identifies major conformational states and transitions, including the stable low-distance period (1.0-1.5 Å) and the transition to extended configurations around timestep 25,000. However, several characteristic behaviors emerge at this extended horizon: (1) the predicted trajectory shows less high-frequency variation than the actual dynamics, suggesting some temporal smoothing, (2) occasional step-like artifacts appear in the predictions, particularly in the later portion of the simulation, and (3) while the model captures the general amplitude of fluctuations in the extended state, it sometimes exhibits reduced responsiveness to rapid conformational changes.

The performance degradation from shorter horizons is expected, with R^2 decreasing from 0.839 (H=1000) to 0.704 (H=5000) and MAPE increasing from 12.330% to 20.180%. This decline illustrates the fundamental challenge of long-term molecular dynamics prediction, where small uncertainties accumulate over time. Nevertheless, the model maintains reasonable accuracy in capturing major conformational trends and state transitions, demonstrating its utility for understanding long-term molecular behavior patterns, albeit with reduced precision in fine-scale temporal dynamics.