Modeling Novel Ionenes for Electrochemical Devices with First Principles and Machine Learning Methods

by

Mehrdad Mokhtari

B.Sc. (Chemical Engineering), Sharif University of Technology, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Chemistry
Faculty of Science

SIMON FRASER UNIVERSITY
Spring 2020

Approval

Name:	**Mehrdad Mokhtari**
Degree:	**Master of Science**
Title:	**Modeling Novel Ionenes for Electrochemical Devices with First Principles and Machine Learning Methods**
Examining Committee:	**Chair:** **Tim Storr** Associate Professor
	Steven Holdcroft Senior Supervisor Professor
	Michael H. Eikerling Supervisor Professor
	Hua-Zhong Yu Supervisor Professor
	Jeffrey Warren Internal Examiner Associate Professor
Date Defended/Approved:	April 6, 2020

Abstract

Polybenzimidazole-based ionenes are being developed for use in both alkaline anion-exchange membrane fuel cells and alkaline polymer electrolysers. The first part of this work explores the impact of the degree of methylation on the conformations and electronic structure properties of poly-(hexamethyl-p-terphenylbenzimidazolium) (HMT-PMBI), the materials of interest in this thesis. For this purpose, HMT-PMBI oligomers, from monomer to pentamer, are studied with density functional theory calculations. Next, molecular dynamics simulations are used to calculate the trajectory paths of all atoms of the fully methylated HMT-PMBI tetramer. Lastly, recurrent neural networks are explored as a means to accelerate the statistical sampling of molecular conformations of polymeric systems, thereby providing complementary tools for molecular dynamics simulations. It is demonstrated that these types of artificial neural networks can be learned from the distribution of the coordinates of atoms over molecular dynamics simulations. As shown, the trained multivariate time series model enables forecasting trajectory paths of atoms accurately and in much reduced time with over 96% accuracy.

Keywords: Anion exchange membrane; density functional theory; recurrent neural network; time series forecasting; machine learning

Dedication

Dedicated to the memory of those who have lost their lives in a tragic plane crash in Iran on January 8, 2020.

Acknowledgements

I would like to thank my advisor Dr. Michael Eikerling for giving me the chance to work in his research group as a graduate student. It was and is always a pleasure to work with such a professional, supportive and caring supervisor.

I would also like to thank my committee members, Dr. Steven Holdcroft, and Dr. Hogan Yu for their comments and guidance during my graduate studies. Specifically, Dr. Holdcroft who was my senior supervisor for the last year. Also, a special thanks to Dr. Holdcroft's research group as all the computational studies in this project were investigated on materials that synthesized in his group as well as Eric Schibli for providing me with molecular dynamics results.

I would acknowledge the financial support provided by NiElectroCan (the pan-Canadian Engineered Nickel Catalysts for Electrochemical Clean Energy).

I would also acknowledge the computational resources provided by WestGrid (www.westgrid.ca) and Compute Canada Calcul Canada (www.computecanada.ca).

I would like to thank all my friends and colleagues, specifically, Dr. Eslamibidgoli, a previous postdoc in our research group who was always available to lend a helping hand from the beginning of my career.

At last, my deep gratitude to my family: my mother, father and brother for their continuous love and support in my entire life, and specifically my beloved partner, Faezeh, for her unconditional love and countless sacrifices to help me overcome tough times.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AEMFC	Anion-Exchange Membrane Fuel Cells
AFC	Alkaline Fuel Cells
ARIMA	Autoregressive Integrated Moving Average
BLSTM	Bidirectional Long Short-Term Memory
CM2	C2-position
CNN	Convolutional Neural Networks
DFT	Density Functional Theory
DLSTM	Deep Long Short-Term Memory
dm	degree of methylation
FF	Force Field
GD	Gradient Descent
GGA	Generalized Gradient Approximation
GRU	Gated Recurrent Units
HMT-PMBI	Poly-(hexamethyl-p-terphenylbenzimidazolium)
HOR	Hydrogen Oxidation Reaction
KS	Kohn-Sham
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator
LSTM	Long Short-Term Memory

MD	Molecular Dynamics
MDN	Mixture Density Network
ML	Machine Learning
MSE	Mean Square Error
NN	Neural Networks
OPLS	Optimized Potentials for Liquid Simulations
OPLS-AA	Optimized Potentials for Liquid Simulations–All Atom
ORR	Oxygen Reduction Reaction
PACF	Partial Autocorrelation Function
PBC	Periodic Boundary Conditions
PCM	Polarizable Continuum Model
PEM	Proton Exchange Membrane
PEMFC	Proton Exchange Membrane Fuel Cells
QM	Quantum Mechanics
RNN	Recurrent Neural Networks
VDW	Van der Waals

Chapter 1.

Introduction

1.1. Fuel cells

Fuel cells are electrochemical devices that convert chemical energy directly into electrical energy. The typical polymer-based fuel cell is comprised of two electrodes, an anode and a cathode, with a membrane sandwiched between them that acts as a medium for the transport of ions, viz. protons in proton exchange membranes (PEMs) or hydroxide anions in anion exchange membranes (AEMs). In a hydrogen PEM fuel cell (PEMFC), electricity is produced in spontaneous redox reactions.[10] The cell generates electricity for as long as the fuel (hydrogen gas) and the oxidant (oxygen or air) are provided. At the anode, hydrogen is oxidized in the hydrogen oxidation reaction (HOR) to produce protons and electrons,

$$2H_{2\,(g)} \rightarrow 4H^+ + 4e^- \qquad E^\circ = 0\,V. \qquad 1.1$$

Here, E° is the electrode potential of the reaction under standard state, i.e., at 1 atm and 25℃.[11] The electrons are conducted from anode to cathode through an external circuit to do electrical work. Protons are conducted through the PEM to the cathode. At the cathode side, electrons, protons and oxygen gas react with each other to form water, in a process called oxygen reduction reaction (ORR),

$$4H^+ + 4e^- + O_{2\,(g)} \rightarrow 2H_2O_{(l)} \qquad E^\circ = 1.23\,V. \qquad 1.2$$

Hence, the overall reaction for this cell is,

$$2H_{2\,(g)} + O_{2\,(g)} \rightarrow 2H_2O_{(l)} \qquad E^\circ = 1.23\,V. \qquad 1.3$$

PEMFCs are fraught with several challenges that impede their path to commercialization. A primary challenge is the use of platinum catalysts, as platinum exhibits high stability under acidic conditions and provides a high activity for the notoriously sluggish ORR.[12] To reduce the amount of platinum used in PEM fuel cells and the associated costs, numerous researches have been strived to minimize the use of Pt

required in catalysts of PEMFCs. The high costs of Pt-based catalysts are still the biggest drawback of PEMFCs.[13] As another drawback, Nafion that is employed as the membrane in PEMFCs has a complicated synthetic route and is costly in comparison to the anion-exchange membranes.[14] AEMs have received notable attention because of the possible improvement in the kinetics of the ORR and the higher flexibility in view of finding cathode catalysts based on non-precious metals, such as nickel-based catalysts.[15] Nickel-based catalysts used in AEMFCs are much cheaper than Pt-based catalysts in PEMFCs.[16]

As shown in Figure 1.1, at the anode catalyst layer of an AEMFC, hydrogen is oxidized as shown in the reaction equation below, releasing electrons and water,

$$2H_2 + 4OH^- \rightarrow 4H_2O + 4e^- \qquad E^\circ = -0.83\,V. \qquad 1.4$$

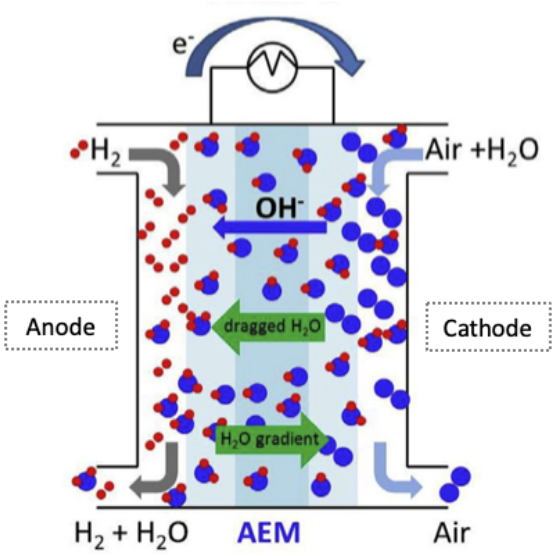


Figure 1.1. Schematic representation of anion-exchange membrane fuel cell (reproduced from Ref. [1]).[1]

At the cathode catalyst layer, oxygen from the air, water, and electrons from the external circuit react to form hydroxide ions,

[1*] https://creativecommons.org/licenses/by-nc-nd/4.0/
https://doi.org/10.1016/j.jpowsour.2017.07.117

$$O_2 + 2H_2O + 4e^- \rightarrow 4OH^- \qquad E^\circ = +0.40\,V, \qquad 1.5$$

with the same overall reaction as stated 1.3. In the next part, various AEM materials and the anion-exchange membrane of interest in this study will be discussed in detail.

1.2. AEM materials

Research on anion-exchange membranes (AEMs) is thriving, propelled by their promise for uses in alkaline electrochemical energy technologies such as fuel cells,[1,17,18] water electrolyzers,[19] redox flow batteries,[20] and waste-water treatment systems.[21] Alkaline conditions offer distinct advantages over acidic conditions. A major driver for the development of alkaline technologies is the possible replacement of Pt- by Ni-based materials as oxygen reduction catalysts.[22] AEMs that are already being tested in fuel cells possess relatively simple synthetic routes, which is another advantage over proton-conducting polymer electrolyte membranes (PEMs).[14,23] AEMs also show promise in reducing the membrane sensitivity to variations in the hydration level, which could reduce humidification requirements and electrode flooding, and thereby diminish system costs.[22,24]

Electro-osmotic drag is an important factor in fuel cells as it directly affects the overall water management and performance of the cell. The larger electro-osmotic drag coefficient (the number of water molecules transferred per proton when protons transport from the anode electrode to the cathode electrode) would result in higher water content within the anode electrode and can accelerate electrode kinetics.[25] The electro-osmotic drag coefficient of water for AEMs with Tokuyama A201 membrane[26] at 30-40°C is about 2.5-3.5.[27] Nevertheless, the electro-osmotic drag coefficient in PEMs with Nafion® 117 is ranging from 1.5 to 2.5 in the varying operating conditions, which are lower than the AEMs values.[28] The larger values of electro-osmotic drag coefficient in AEMs can lead to electrode flooding at the anode and dehydration at the cathode side, which lower cell performance.[25]

On the flip side, a decade ago, the ion conductivity of AEMs trailed that of PEMs by a significant factor. Nafion® 117, as the benchmark PEM,[29] has a proton conductivity of 78 $mS.cm^{-1}$, whereas anion conductivities of AEMs used to lie in the range of 5 to 20 $mS.cm^{-1}$;[30,31] however, over the past years, conductivities of AEMs have seen significant

improvement, with values reported recently in the range from 50 to 200 mS.cm^{-1}.[32–34] As a matter of fact, over the last 10 years the number of publications on polyaromatic AEMs that partially or fully contain benzene rings, including polybenzimidazoles, has increased significantly in comparison to the number of publications on polyolefin and perfluorinated PEMs, which had stagnating publication numbers[2] (see Figure 1.2).

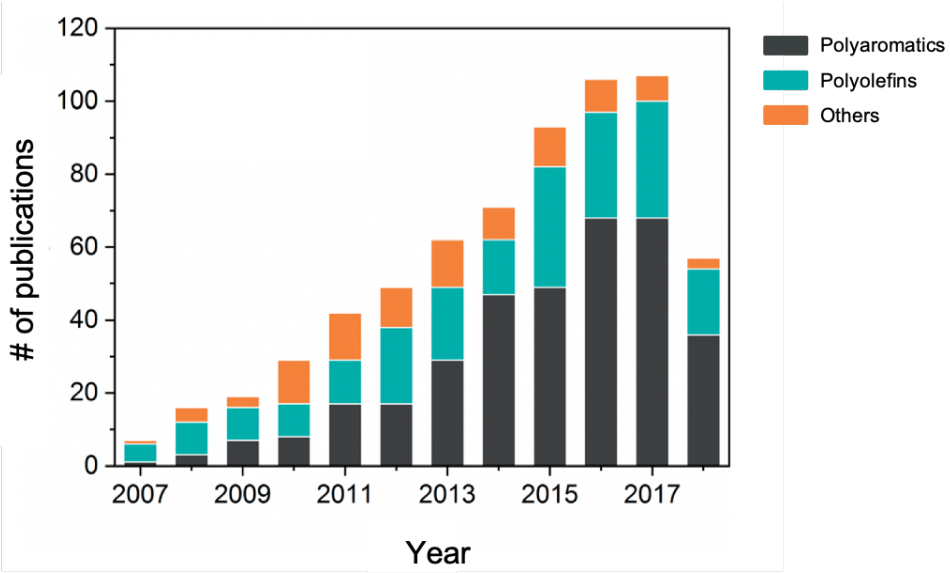


Figure 1.2. Number of research articles on polymer electrolyte membranes for AMFCs over the 10 years (as of May 8, 2018) (reproduced from Ref. [2]).[2]

Photo is reproduced by permission of The Royal Society of Chemistry.[2*]

Over the past two decades, materials chemists have tested strategies in chemical design and synthesis to overcome the issue of the poor chemical stability of AEMs.[35] The most common cationic moieties employed are phosphonium,[36] sulfonium,[37] pyridinium,[38] ammonium,[39] piperidinium[40], as well as imidazolium-based cations among which benzimidazolium is the most promising in terms of stability and synthetic route.[32,41,42] In a charged benzimidazole ring, the nitrogen cation stability is provided by steric protection via methyl groups.[42]

[2*] https://doi.org/10.1039/C8TA05428B

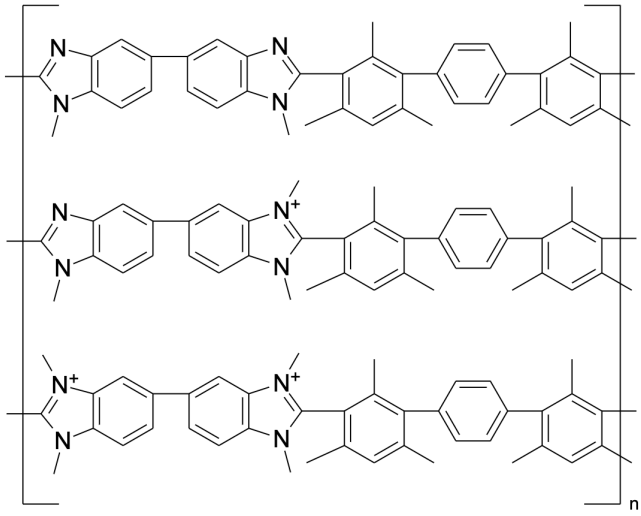


Figure 1.3. Chemical structure of 50% (one on top), 75%, and 100% dm HMT-PMBI. Structure by "ChemDraw JS".[3]

This work focuses on a sterically C2-protected poly(benzimidazole) material, called poly-(hexamethyl-p-terphenylbenzimidazolium), in short HMT-PMBI, that was developed by the Holdcroft Group, Simon Fraser University.[3] This compound is hydroxide-stable, methanol-soluble, and water-insoluble, which renders it highly suitable for uses in alkaline fuel cells and electrolyzers.[3,23] It exhibits unprecedented hydroxide stability and ion conductivity in the temperature range from 25 to 80 °C and for NaOH concentration from 1 to 6 M. The material was synthesized with varying degree of methylation (dm), as reported in Ref. [3] and shown in Figure 1.3.

1.3. Machine-learned algorithms: literature review

Although the primary methods were developed in the 1950s to 80s, breakthroughs in machine learning have occurred only recently, with many interesting applications such as in computer vision,[43] speech and language technology,[44] self-driving cars,[45] recommender systems,[46] financial predictions,[47] and many more.[48] This is mainly owed to recent advances in neural network architectures and algorithms,[49] the escalating growth of data for model training,[50] powerful parallel computer processing, enhanced frameworks for implementation[51] and, of course, larger involvement from the scientific community as well as industry in the field.

Machine learning is well-established in areas like bioinformatics, as a computational method to analyze and interpret biological data.[52,53] Deep neural network architectures, such as multi-layered networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs) and memory networks, are the most used architectures in this area.[54] Machine learning has also shown to be promising in theoretical chemistry[55] mainly to speed up the discovery of novel compounds or materials,[56–59] as well in terms of predicting the potential energy surface of chemical structures.[60,61]

Extensive research has been focusing on developing inter-atomic potentials[62] or performing AIMD simulations.[63,64] Brockherde *et al*. employed the kernel ridge regression training algorithm based on an external Gaussian potential to generate the machine learned potential-density map of a given molecule.[60] Their proposed approach starts with the construction of descriptors that encode the structure of each molecule from a training dataset. The kernel matrix is then developed from a kernel function to represent the correlation of descriptors to each other.[60] The dataset in their work is composed of 2000 DFT optimized structures.

Gomez-Bombarelli *et al*.[57] recently used variational autoencoders to transform the discrete representation of molecular structures into a continuous latent space, from which a decoder neural network converts these continuous representations back into a discrete molecular representation. This approach allows designing new molecules and enables efficient exploration and optimization of the chemical spaces of compounds.[57] Chmiela *et al*. advanced a symmetrized gradient-domain machine learning model, in which one can create a molecular force field of intermediate-sized organic compounds with the same accuracy as high-level *ab initio* calculations only using limited samples (of size 1000) of AIMD calculated trajectories.[65] In another work, Xie and Grossman proposed a crystal graph convolutional neural network framework to provide a universal representation of crystal materials and an accurate prediction of DFT calculated properties of various crystal structures and compositions.[66]

Time-series forecasting also has become a thriving research topic over the past couple of years; a significant example is the seq2seq model proposed by Google to make multi-step sequence predictions in order to solve the machine translation problem.[67] An important class of RNN architectures designed for sequence-to-sequence problems is called the encoder-decoder long short-term memory (LSTM), which comprises an encoder

that receives input signals and returns a fixed-length internal representation as a vector, and a decoder that interprets this vector and uses it to predict the output signal.[68] This approach is powerful because we can have an input of univariate time-series data, and we can encode the multi-variate features of a time-series dataset. Other types of RNNs have been employed for such predictions such as LSTM,[69] bidirectional long short-term memory (BLSTM) and mixture density network (MDN) approaches,[70,71] and gated recurrent units (GRU).[72]

Zhao *et al*.[71] proposed a BLSTM-MDN approach for three-point shot prediction in basketball games. They have utilized a Python library called Hyperopt[73] for hyperparameter self-tuning during model training. Faster convergence rates and more accurate predictions are superior features of their approach in comparison to CNNs and MDNs. In terms of predicting the trajectories, their proposed model can produce new trajectory samples beyond predicting those stemming from the real data. Another example reported is predicting amazon spot price using a three-layer LSTM-based network, composed of two LSTM layers and one dense neural network layer, by Baughman and his coworkers.[74] Their model has revealed a reduction in mean square error (MSE) of 60% up to 95% for different training sets relative to the well-known Autoregressive Integrated Moving Average (ARIMA) model.[75]

Sagheer and Kotb[76] employed deep LSTM (DLSTM) recurrent networks for univariate time series forecasting of petroleum production. To find the optimum architecture of DLSTM and optimal selection of hyperparameters, a genetic algorithm was used in their work. For evaluation purposes, the ARIMA model,[75] the Vanilla RNN model,[77] the deep GRU model,[78] and the Higher-Order Neural Network model[79] were employed; using root mean square error and root mean square percentage error[80] measures, the proposed deep LSTM approach outperforms the other standard models.

1.4. Thesis outline

Despite the promising attributes of HMT-PMBI, to the best of our knowledge only one DFT work has been performed on HMT-PMBI material,[23,81] but no comprehensive computational exploration of its molecular conformation and electronic structure has been reported so far. Since physical properties of HMT-PMBI are strongly affected by the degree of methylation, in this study we consider the degree of methylation as the main

parameter.[23,41] The primary objective of the second chapter of this thesis is to understand the effect of the degree of methylation on molecular conformation, electronic structure of HMT-PMBI.

In the third chapter, based on the outputs of molecular dynamics simulations and force field parameters provided by Eric Schibli,[23,81] PhD student in Physics Department at Simon Fraser University, a trajectory history of all atoms of a single HMT-PMBI^{2+} tetramer is created. This trajectory history, i.e., the positions of atoms in 3-dimensional space recorded as a function of time, is being used as the input data for the training of the proposed neural networks. We employ recurrent neural networks-based models for the prediction of trajectory path in polymeric materials over Molecular Dynamics simulations. As is known, the main purpose of MD simulation is to generate as many as possible statistically independent configurations of the system, which requires expensive simulation runs. RNN-based algorithms are shown to accelerate the statistical sampling of molecular conformations and generate the trajectory paths as accurate as MD simulations, but in much less time. We treat the interactions between atoms as temporary correlations, thus, the problem of predicting the trajectory of atoms reduces to a multi-variate time-series forecasting problem.

The overview graphic in Figure 1.4 illustrates the contributions provided in the main chapters of this thesis.

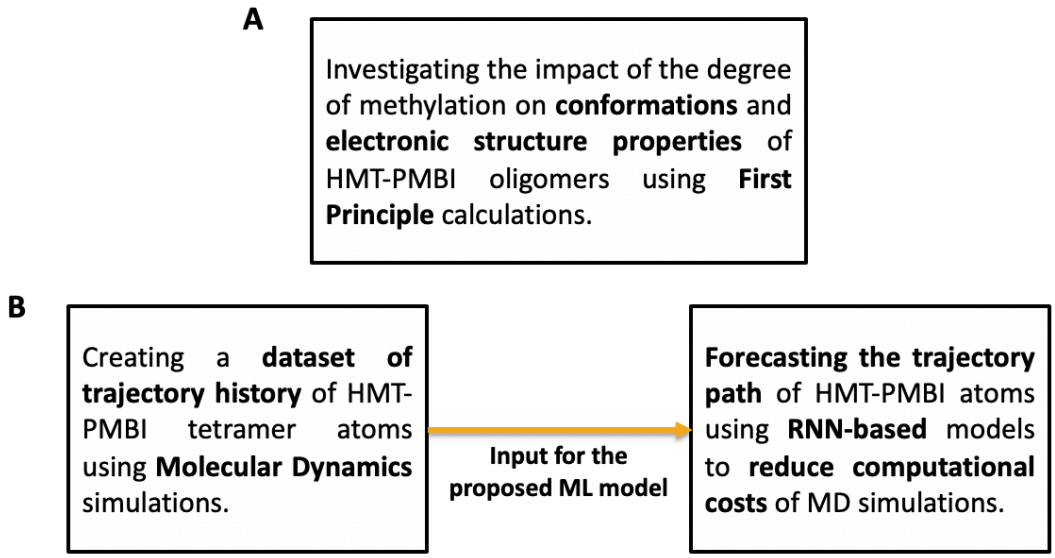


Figure 1.4. An overview graphic which summarizes thesis objectives. Part A demonstrates the second chapter and part B gives a precis of the third chapter of thesis.

Chapter 2.

DFT study of the conformational and electronic properties of HMT-PMBI

As for the very first step of this work, we[3,*] have performed calculations based on quantum mechanical density functional theory to study the influence of the ionic charge distribution on the conformation and electronic structure properties of polybenzimidazole-based ionenes.[82] Polybenzimidazole-based ionenes are highly suitable for uses in both alkaline anion-exchange membrane fuel cells and alkaline polymer electrolyzers. HMT-PMBI, the material of interest in this article, is exceptionally hydroxide-stable and water-insoluble. Optimization studies are presented for both the gas-phase and in the presence of implicit water. Results are insightful for experimentalists and theorists investigating the impact of synthetic and environmental conditions on the conformation and electronic properties of polybenzimidazole-based membranes.

2.1. Theoretical background

Quantum mechanics (QM) provides the physical-mathematical basis for studying the behavior of electrons, atoms, and molecules in chemistry. Mathematically exact solutions of the QM equations can only be obtained for one-electron systems.[83] Over the past half century, a vast number of methods including density functional theory (DFT) has been developed to obtain approximate solutions of the Schrödinger equation for many-body (multiple electron) systems.[84,85]

[3] This chapter reproduces in revised form material from Ref. [82] (https://pubs.acs.org/doi/full/10.1021/acsomega.9b03116), with permission from the American Chemical Society and any further permissions related to the material excerpted should be directed to the ACS.

* Mohammad J. Eslamibidgoli has also contributed to this chapter.

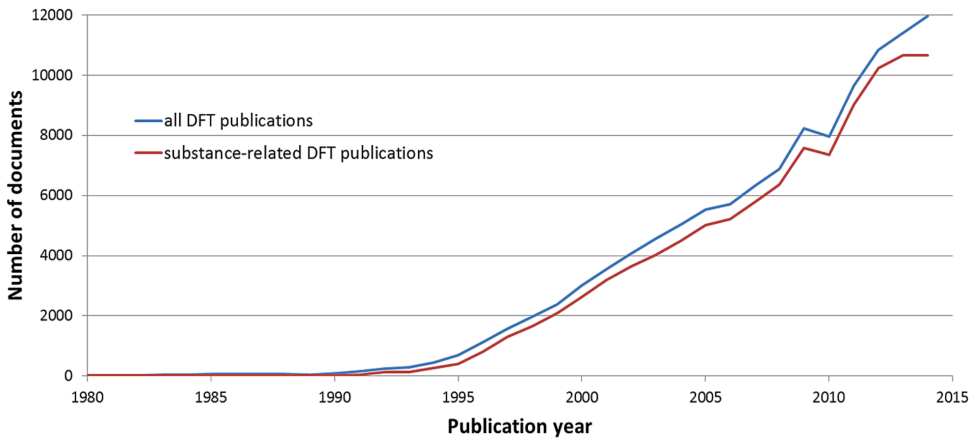


Figure 2.1. The number of DFT and substance related DFT publications annually from 1980-2015 (reproduced from Ref. [4]).[4]

Permission is not required for this photo under the terms of the Creative Commons Attribution 4.0 International License.[4*]

As shown in Figure 2.1, DFT is one of the most well-known and flourishing quantum mechanical approaches and widely employing method to calculate the electronic structure properties of matter.[4] From a quantum mechanical point of view, all information about a given system is contained in the system's wave function, $\Psi(r)$, which is a function of the electronic coordinates, r, as well as the coordinates of the nuclei, R. The wave function is calculated from time independent Schrödinger's equation, which for a many-electron system with potential energy of $V(r)$, is given by[86]

$$\hat{H}\,\Psi(r,R) = E\,\Psi(r,R). \tag{2.1}$$

The Hamiltonian operator, $\hat{H}$, for a system composed of M nuclei and N electrons is defined as

$$\hat{H}\,\Psi(r,R) = \left[\hat{T}_n + \hat{T}_e + \hat{V}_{nn} + \hat{V}_{ee} + \hat{V}_{ext}\right]\Psi(r,R). \tag{2.2}$$

In Equation 2.2, $\hat{T}_n$, states nuclear kinetic energy,

$$\hat{T}_n = -\sum_{i=1}^{M}\frac{\hbar^2}{2m_A}\nabla_A^2, \tag{2.3}$$

[4*] http://creativecommons.org/licenses/by/4.0/
https://doi.org/10.1186/s13321-016-0166-y

and term $\hat{T}_e$, denotes electronic kinetic energy,

$$\hat{T}_e = -\sum_{i=1}^{N} \frac{\hbar^2}{2m_e} \nabla_i^2, \tag{2.4}$$

Nucleus-nucleus, electron-electron, and nucleus-electron interaction energies are represented by $\hat{V}_{nn}$, $\hat{V}_{ee}$ and $\hat{V}_{ext}$,

$$\hat{V}_{nn} = -\sum_{A=1}^{M} \sum_{B>A}^{M} \frac{Z_A Z_B e^2}{4\pi\varepsilon_0 |R_A - R_B|}, \tag{2.5}$$

$$\hat{V}_{ee} = -\sum_{i=1}^{N} \sum_{j>i}^{N} \frac{e^2}{4\pi\varepsilon_0 |r_i - r_j|}, \tag{2.6}$$

$$\hat{V}_{ext} = -\sum_{i=1}^{N} \sum_{A=1}^{M} \frac{Z_A e^2}{4\pi\varepsilon_0 |R_A - r_i|}. \tag{2.7}$$

Here, $|R_A - R_B|$, $|R_A - r_i|$ and $|r_i - r_j|$ are the relative nuclear distance between the A^{th} and B^{th} nucleus, the distance between the i^{th} electron and A^{th} nucleus, and the relative distance between the i^{th} and j^{th} electrons, respectively. Z_A, m_A and m_e are the charge and mass of the A^{th} nucleus and the latter is the electron mass. Terms $\hbar$, ε_0 and e are the reduced Planck constant, the permittivity of free space, and the elementary charge (charge on the electron), respectively.[87]

The Schrödinger equation will become inseparable starting from the two-body Coulomb interactions, which makes solving the Schrödinger equation impossible. Therefore, to be able to solve this equation, approximations are required. In this study, the usual Born-Oppenheimer approximation was invoked.[88] The basis for this approximation is that the mass of a nucleus is over 1800 times higher than the mass of an electron so nuclei can be supposed as immobilized points in space. In the Born-Oppenheimer approximation, the many-body electronic wave functions can also be defined as product of single-particle wave functions. This fact separates nuclear and electronic degrees of freedom and factorizes the wave function Ψ into a nuclear wave function Ψ_{nuclei} and an electronic wave function $\Psi_{electrons}$, where the total wave function will be shown as $\Psi = \Psi_{nuclei} \times \Psi_{electrons}$.[87]

The Hamiltonian operator in Equation 2.2 by assuming the Born-Oppenheimer approximation will be written as $\hat{H} = \hat{T}_e + \hat{V}_{ee} + \hat{V}_{ext}$, where $\hat{T}_e$, $\hat{V}_{ee}$ and $\hat{V}_{ext}$ are electronic

kinetic energy, electron-electron interaction energy and nucleus-electron interaction energy, respectively.

Density functional theory is another and perhaps the best approach to solve the Schrödinger equation and is widely being used in the theoretical chemistry community.[89] DFT employs the electron density, $\rho(\boldsymbol{r})$, instead of the electronic wave function.[90] In 1964, Hohenberg and Kohn[91] introduced the basis of density functional theory. Assuming the ground state of electrons is nondegenerate, the first Hohenberg and Kohn theorem states that the ground state electronic density $\rho_0(\boldsymbol{r})$ is a unique functional of an external potential $v(\boldsymbol{r})$, i.e., potential energy due to electron-nucleus interactions,

$$E_0 = \int \rho_0(r) v(\boldsymbol{r}) \mathrm{dr} + < \mathrm{T}[\rho_0] > + < V_{ee}[\rho_0] >, \tag{2.8}$$

where E is the electronic energy of the system. The first term on the right hand-side of Equation 2.8 represents the potential energy due to electron-nuclei interactions and the next two terms are the kinetic and potential energy of electrons, respectively.

The second Hohenberg-Kohn theorem expresses that any trial electron density offers an energy higher than the true ground state energy calculated with E_0,

$$E[\rho_t(r)] \geq E\,[\rho_0(\boldsymbol{r})]. \tag{2.9}$$

Kohn–Sham (KS) equation (Eq. 2.10) is the one-electron Schrödinger equation of a fabricated system of non-interacting particles (normally orbitals), which produce the same density as any given system of interacting particles. KS equation was first proposed by Walter Kohn and Lu Sham[92] in 1965, where $v_{eff}(r)$ and $v_{xc}(r)$ in this equation are the Kohn–Sham potential and the exchange-correlation potential, respectively,

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + v_{eff}(r)\right]\Psi_i(r) = \varepsilon_i\,\Psi_i(r), \tag{2.10}$$

$$v_{eff}(r) = v(r) + \int \frac{\rho(r')}{|r-r'|}\,dr' + \,v_{xc}(r), \tag{2.11}$$

where ε_i is the orbital energy of the corresponding KS orbital, $\Psi_i(r)$. The electron density for the many-body system and exchange-correlation potential can be calculated by

$$\rho(r) = \sum_i^N |\Psi_i(r)|^2, \tag{2.12}$$

$$v_{xc}(r) = \frac{\delta E_{xc}[\rho]}{\delta \rho}. \quad (2.13)$$

In Equation 2.13, exchange-correlation potential is equal to derivative of the exchange-correlation energy functional, E_{xc}, with respect to the electron density, ρ. The exact form of the exchange-correlation energy functional is not known; however, it is universal, which means the functional is not dependant on the substance being investigated. The most prominent approximations for E_{xc} are local density approximation (LDA), the generalized gradient approximation (GGA) and hybrid functionals.[93] The generalized gradient approximation which is used in our study has the general form,

$$E_{xc}[\rho] = \int \rho(r)\, \varepsilon_{xc}[\rho, \nabla\rho]\, \mathrm{dr}, \quad (2.14)$$

where ε_{xc} is the exchange-correlation energy to consider the non-homogeneity of the electron density.[94] Using the GGA approximation in DFT calculations, gives accurate results for molecular geometries and ground-state energies.[83]

In Figure 2.2, a schematic representation of an iterative method for solving Kohn-Sham equations is shown. First, an initial guess for the electron density, $\rho(r)$, is taken. Then from the initial guess Kohn–Sham potential, $v_{eff}(r)$, is computed. Now, the Kohn–Sham equation (see Equation 2.10) can be solved and using Equations 2.8 and 2.12 electron density and electronic energy of the system will be evaluated. If the convergence criterion is not satisfied the loop should be iterated with the updated value of electron density instead of the initial guess of that. As soon as the criterion is satisfied, various outputs such as the ground state electron density, $\rho_0(r)$, and the total ground state electronic energy, $E_{tot}[\rho_0(r)]$, can be calculated.[6]

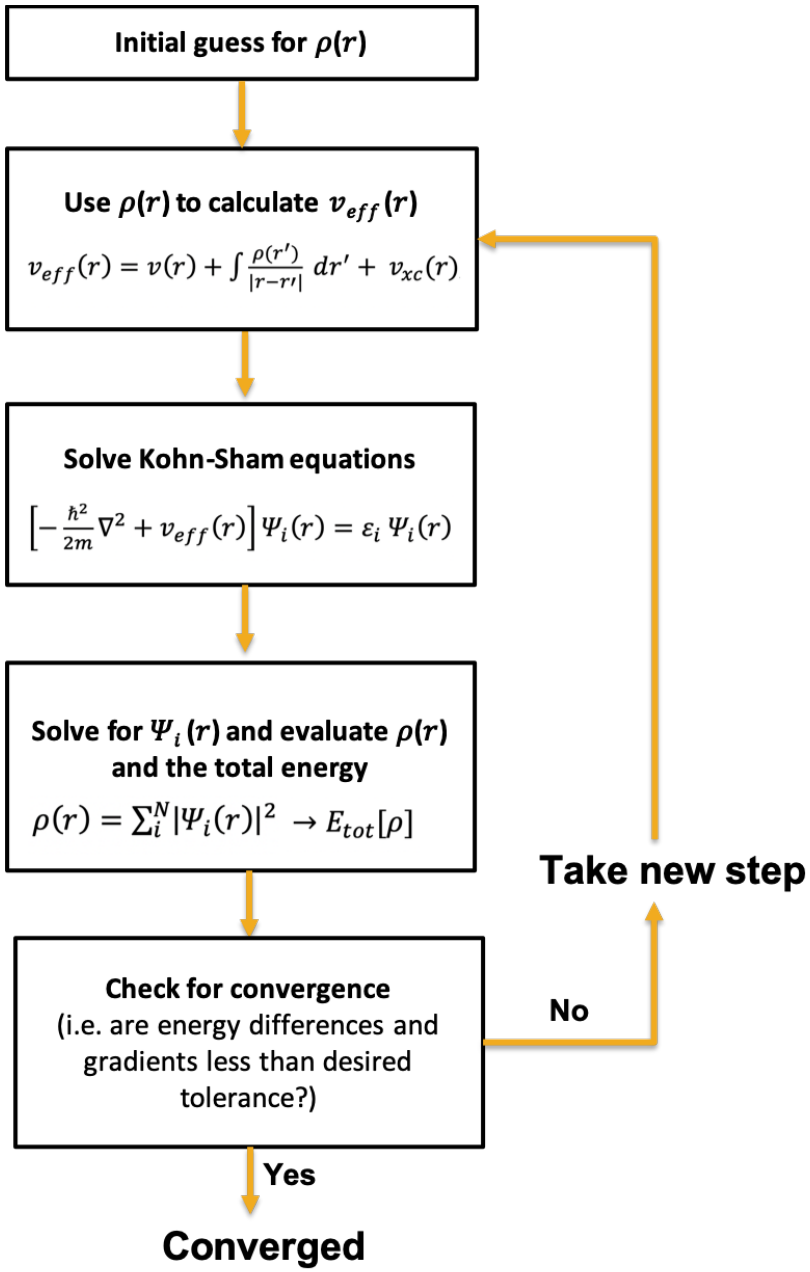


Figure 2.2. Basic schematic of solving the Kohn-Sham equations.[5,6]

2.2. Computational details

Calculations in this work have been performed with Gaussian 16.[95] Gaussian 16 includes an extensive suite of DFT tools.[91,92] In comparison to Gaussian 9, the default integration accuracy in Gaussian 16 has been enhanced from 10^{-10} to 10^{-12}.[96] We have performed DFT calculations from monomer to pentamer of HMT-PMBI. HMT-PMBI repeating unit with 50% degree of methylation is electroneutral, while and 75% and 100% degree of methylation correspond to a charge +1 and +2, respectively.[3] We calculated the electronic band gap, E_{gap}, which is defined as the minimal energy difference between HOMO and LUMO levels.

All structures were fully optimized to find the ground state energy. The condition for attaining the ground state is satisfied when the gradient of the total energy with respect to the nuclear coordinates is zero. The convergence criterion for the calculation is as follows:[95] the maximum component of the total force must be below a cut-off value 0.00045 N; the root-mean-square of the force below 0.0003 N; the calculated displacement for the next step below the cut-off value 0.0018 Å; and the root-mean-square of the displacement for the next step below 0.0012 Å. We employed the 6-31G(d) basis set to stay consistent with the results produced in Ref. [23].

We tested various functionals for calculating HOMO, LUMO and E_{gap}, including B97D, Grimme's functional with dispersion correction;[97] PBE functional, which is at the generalized gradient approximation (GGA) level.[98] TPSS functional, which is a non-empirical meta-GGA;[99] O3LYP[100] and B3LYP[101] hybrid functionals, which are very similar but have slightly different mixing coefficients; PBE1 which is also a hybrid exchange–correlation functional based on the GGA PBE;[102] wB97XD,[103] which is known as a range-separated functional that includes Grimme's D2 dispersion model[97] to capture both short-range interactions and long-range corrections. Our main motivation for testing these seven DFT functionals was to explore how well they can describe the electronic properties of the ionene system in comparison to the B3LYP functional, which is the only functional used to date for this system.[81] Solvent effects were investigated by optimizing the tetramer structures in the presence of implicit water using the polarizable continuum model (PCM).[104,105] The PCM model generates a solute cavity thorough a set of overlapping

spheres.[106] GaussView 6.0 visualization software was used to generate the input structures and display the output geometries and orbitals.[107]

2.3. Results and discussion

Optimized structures of repeating unit of PBI, HMT-PMBI, HMT-PMBI^{+}, HMT-PMBI^{2+} obtained using the B3LYP functional under vacuum conditions in gas phase are shown in Figure 2.3. For PBI, the phenyl ring and the two adjoining benzimidazole rings are in-plane. The torsional angle between two adjoining benzimidazole rings was found to be 38.9° and the one between phenyl and benzimidazole groups was 7.6°, as also shown in Table 2.1. For the HMT-PMBI repeating unit, however, due to the steric interaction between the mesitylene ring and benzimidazole group, a torsional angle of about 104° was observed.

Considering the fully methylated HMT-PMBI repeating unit, the angles between the adjacent benzimidazole and mesitylene planes is 81.9°, and 40.4° between two consecutive benzimidazole groups. By increasing the degree of methylation, the angle between the adjacent mesitylene and benzimidazole planes decreases from 106.6° to about 82°, as shown in Table 2.1. The reason of this change is the additional methyl group in the benzimidazole unit, which provides steric protection to the cationic imidazole rings.

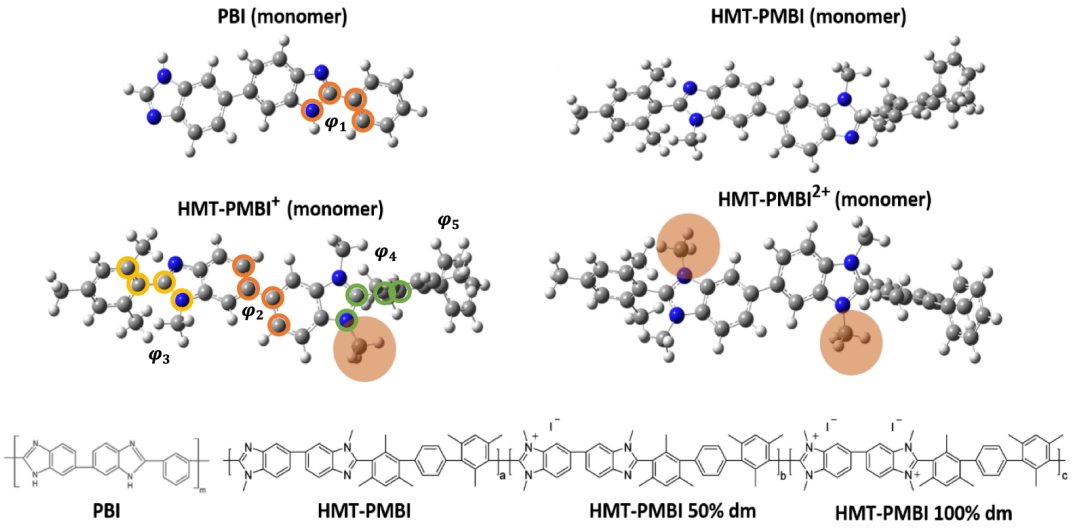


Figure 2.3. **Chemical structure of the monomer of PBI, HMT-PMBI, HMT-PMBI^{+}, HMT-PMBI^{2+}. Torsional angles $\varphi_1 - \varphi_5$ shown in orange, yellow, and green colors represent angles between phenyl, benzimidazole, and mesitylene groups.**

Table 2.1. **Comparison of torsional angles between phenyl, benzimidazole, and mesitylene groups for various repeating unit studies systems. (Torsional angles are shown in Figure 2.3)**

	$\varphi_{\text{phenyl\_benz}}\ (\varphi_1)$	$\varphi_{\text{benz\_benz}}\ (\varphi_2)$	$\varphi_{\text{mes\_benz}}\ (\varphi_3)$	$\varphi_{\text{benz\_mes}}(\varphi_4)$	$\varphi_{\text{mes\_phenyl}}\ (\varphi_5)$
PBI	7.60	38.9	-	-	-
HMT-PMBI	-	37.9	103.7	106.6	83.2
HMT-PMBI^{+}	-	35.9	103.3	86.3	89.8
HMT-PMBI^{2+}	-	40.4	97.8	81.9	92.3

Figure 2.4 shows the optimized structures from monomer to pentamer of the neutral and the charged HMT-PMBI polymers along with their end-to-end distances, which are normalized to the number of repeating units. As can be seen, results reveal a trend in

the chain conformation of HMT-PMBI as a function of the degree of methylation. As the charge increases, the normalized end-to-end distance becomes larger from around 16 Å for HMT-PMBI to around 19 Å for HMT-PMBI^{+}, to around 21 Å for HMT-PMBI^{2+}. The observed chain stretching is caused by the electrostatic repulsion along the backbone. Both gas phase and solvent studies are relevant in real systems as the hydration level is not uniform across the membrane. The membrane could exist in fully hydrated condition as well as in partially or completely dry conditions. Hence, we investigated the solvent effects by optimizing the tetramer structures in the presence of implicit water using the PCM as discussed in computational details section.[104,105] As expected, the charge screening provided by the solvent suppresses the electrostatic repulsion among charged backbone units rendering the normalized end-to-end distance almost unchanged from the one obtained for the corresponding neutral polymer. The DFT-optimized structure of 100% dm HMT-PMBI trimer in gas-phase, as presented in References [23] and [81], reveals the angle between the adjacent mesitylene and benzimidazole groups and two adjacent benzimidazoles as 77° and 47°, respectively, which is consistent with our findings.

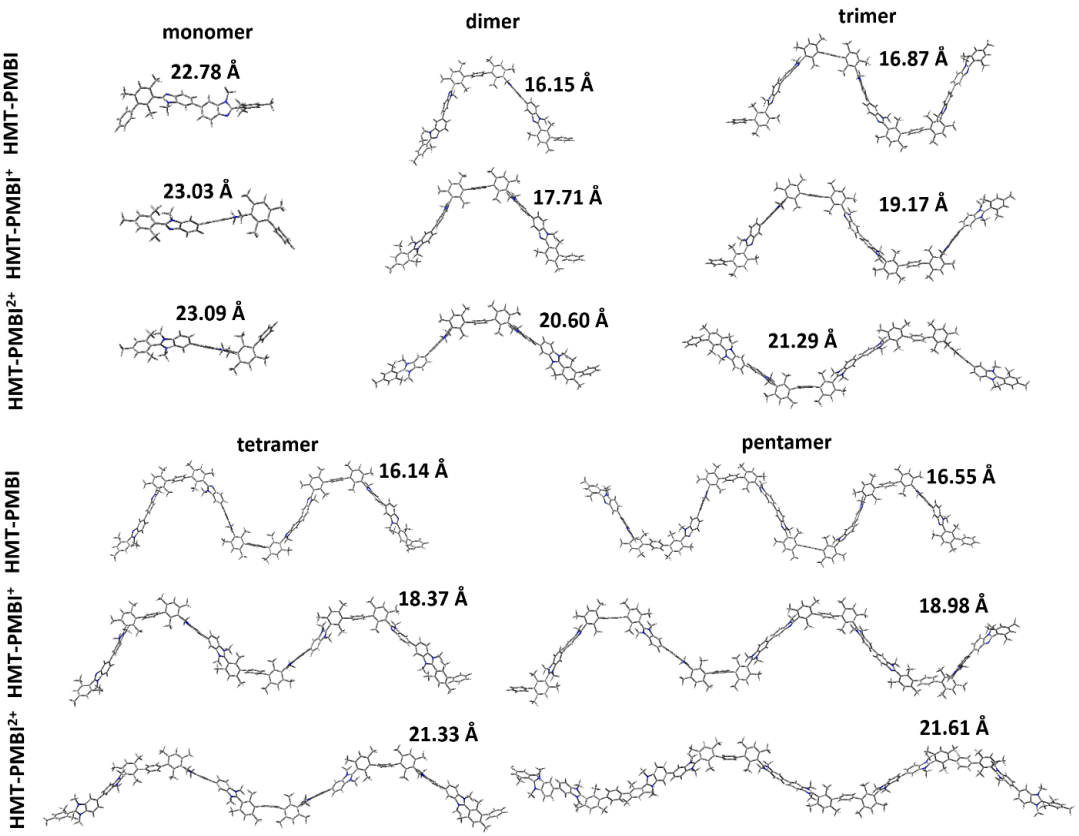


Figure 2.4. Effect of charge on the conformations of various studied systems in gas-phase. Numbers represent the normalized end-to-end distance values.

Figure 2.5 shows the electrostatic potential energy map. On neutral HMT-PMBI, negative charge is accumulated on the nitrogen atoms of the benzimidazole ring, indicated in red. As the degree of methylation increases, more positive charge is distributed on benzimidazole groups of the backbone, as shown in darker blue for HMT-PMBI^{+} and HMT-PMBI^{2+}. This is in agreement with the observation of stretching and the change in the persistence length value, caused by electrostatic repulsion.[100,108]

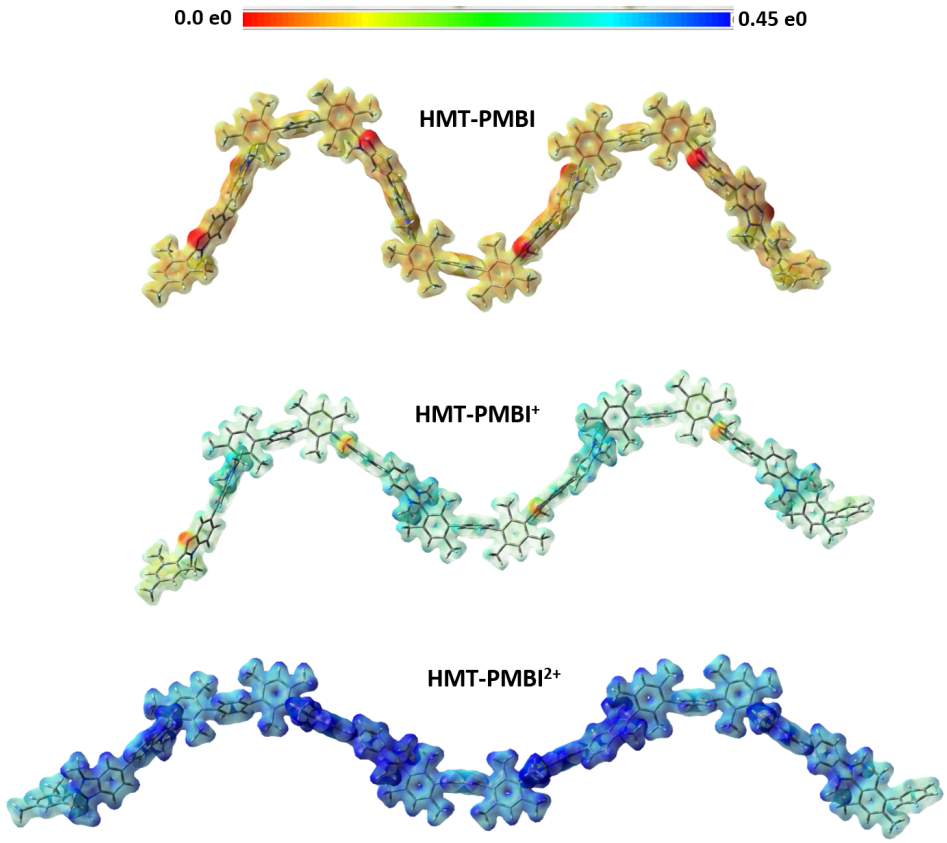


Figure 2.5. Electrostatic potential energy map of the tetramer of HMT-PMBIs.

In Figure 2.6, HOMO, LUMO and E_{gap}, obtained with the B3LYP functional, are plotted as functions of oligomer length for the various degrees of methylation. As the length of oligomer increases the structure becomes more stabilized and HOMO, LUMO and E_{gap} energy levels converge to specific values corresponding to the infinite (polymer) structure. For instance, for HMT-PMBI$^+$, HOMO, LUMO and band gap energy level of around -9.0, -6.5, and 2.5 eV, respectively. It should be noted that for charged oligomers the drop in the energy levels from monomer to trimer is significantly larger (over 20 times larger) than that of the neutral HMT-PMBI and PBI oligomers. Moreover, the increasing degree of methylation causes a decrease in HOMO and LUMO energy levels as well as in the band gap of HMT-PMBI. The band gap for HMT-PMBI$^+$ and HMT-PMBI^{2+} is decreased by approximately 2.0 eV and 3.0 eV relative to neutral HMT-PMBI, respectively.

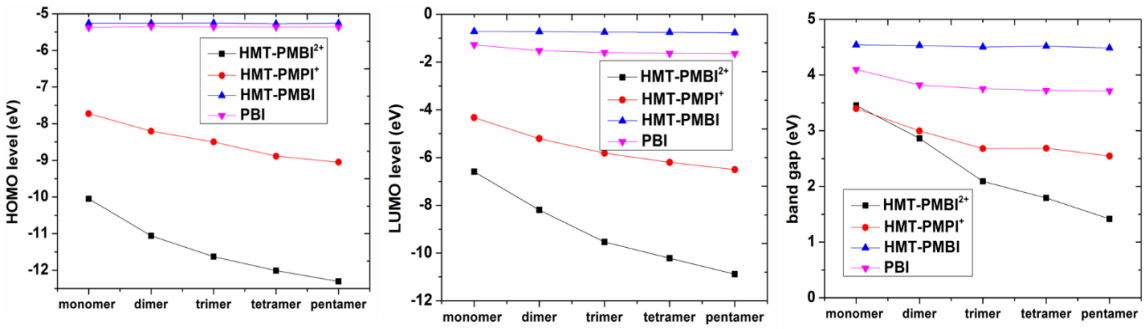


Figure 2.6. The HOMO, LUMO and band gap of the studied polybenzimidazole-based ionenes.

The values of HOMO, LUMO and E_{gap} are shown in Table 2.2. HOMO and LUMO levels of neutral HMT-PMBI lie slightly above those of PBI as shown in Figure 2.6, consequently the band gap of PBI is lower than that of HMT-PMBI by about 0.5 eV. In general, the relatively large range of E_{gap}, varying from 1.4 to 4.5 eV, implies that the electronic and optical properties of HMT-PMBI-based polymers depend strongly on the degree of methylation and the oligomer length. Depending on these structural properties the charged polymers could have either insulating or semiconducting properties. Since the results of the band gap energy for neutral HMT-PMBI (blue line in Fig. 2.6) display a consistent value of about 4.5 eV for various oligomer lengths, it is expected that neutral HMT-PMBI polymers have insulator properties.[109]

Table 2.2. Comparison of HOMO, LUMO and band gap data of different oligomers of PBI and HMT-PMBIs using B3LYP functional

Material	PBI			HMT-PMBI		
Functional	HOMO	LUMO	E_{gap}	HOMO	LUMO	E_{gap}
monomer	-5.38	-1.28	4.10	-5.26	-0.72	4.54
dimer	-5.35	-1.52	3.82	-5.26	-0.73	4.53
trimer	-5.36	-1.60	3.75	-5.25	-0.75	4.50
tetramer	-5.36	-1.64	3.72	-5.27	-0.75	4.52
pentamer	-5.36	-1.65	3.71	-5.26	-0.77	4.49
Material	$HMT\text{-}PMBI^{+}$			$HMT\text{-}PMBI^{2+}$		
Functional	HOMO	LUMO	E_{gap}	HOMO	LUMO	E_{gap}
monomer	-7.72	-4.33	3.40	-10.05	-6.60	3.45
dimer	-8.21	-5.21	2.99	-11.06	-8.20	2.86
trimer	-8.50	-5.82	2.68	-11.63	-9.54	2.09
tetramer	-8.89	-6.20	2.68	-12.01	-10.22	1.79
pentamer	-9.05	-6.50	2.55	-12.31	-10.89	1.42

Figure 2.7 shows the molecular orbital densities for the repeating units of the studied systems in the gas phase and in the presence of implicit water. Evidently, introducing an electron diminishing group onto the polymer backbone, i.e., a methyl group, has significant influence on the HOMO and LUMO energy levels. For the PBI repeating unit the HOMO level is delocalized along the backbone indicating a strong electronic coupling between the subunits. On the neutral HMT-PMBI repeating unit, both in the absence and presence of solvent, the HOMO level is predominantly localized on the benzimidazole groups and the contributions from phenyl and mesitylene rings are almost negligible. Weaker electronic coupling is expected in this case due to the relatively large torsional angles between these groups. On 75% dm HMT-PMBI, the HOMO level is further localized with negligible contribution from the charged nitrogen of the benzimidazole unit. This trend is more significant on fully methylated HMT-PMBI, where the HOMO is strongly localized on the phenyl unit. In the presence of implicit water, however, the HOMO is more delocalized which is due to the charge screening effect exerted by the solvent. As also

shown in Figure 2.7, the electron densities of the LUMO of PBI and HMT-PMBI are rather delocalized, while those for HMT-PMBI^{+} and HMT-PMBI^{2+} are more localized on the charged units both in the gas-phase and in the presence of implicit water. In Table 2.3, we report the HOMO, LUMO and E_{gap} values for the tetramer of PBI and HMT-PMBIs with various degree of methylations obtained with different DFT functionals.

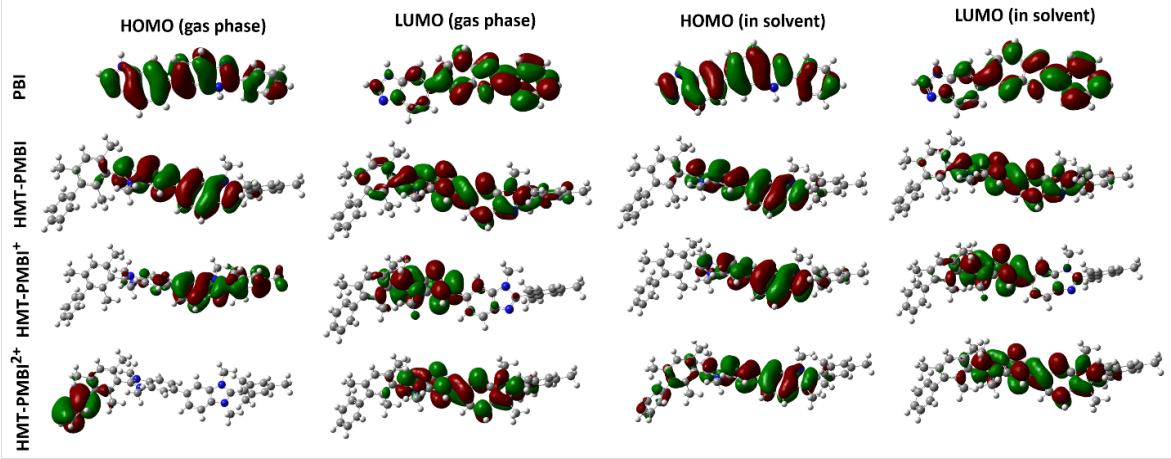


Figure 2.7. The HOMO and LUMO of the studied systems in gas-phase and in the presence of water solvent.

Table 2.3. Comparison of HOMO, LUMO and band gap data of tetramer of PBI and HMT-PMBIs using various functionals

Material	PBI			HMT-PMBI		
Functional	HOMO	LUMO	E_{gap}	HOMO	LUMO	E_{gap}
B97D	-4.66	-2.21	2.45	-4.50	-1.48	3.02
PBE	-4.77	-2.31	2.45	-4.61	-1.53	3.08
TPSS	-4.75	-2.20	2.55	-4.60	-1.40	3.20
O3LYP	-5.08	-1.81	3.27	-4.96	-0.84	4.11
B3LYP	-5.36	-1.64	3.72	-5.27	-0.75	4.52
PBE1	-5.62	-1.55	4.07	-5.52	-0.69	4.82
wB97XD	-7.23	-0.12	7.11	-7.17	1.01	8.17
Material	HMT-PMBI^{+}			HMT-PMBI^{2+}		
Functional	HOMO	LUMO	E_{gap}	HOMO	LUMO	E_{gap}
B97D	-7.95	-6.81	1.14	-11.17	-10.78	0.39
PBE	-8.10	-6.90	1.19	-11.27	-10.89	0.39
TPSS	-8.12	-6.78	1.34	-11.27	-10.76	0.51
O3LYP	-8.58	-6.39	2.19	-11.66	-10.41	1.25
B3LYP	-8.88	-6.20	2.68	-12.01	-10.22	1.79
PBE1	-9.11	-6.11	3.00	-12.33	-10.12	2.22
wB97XD	-10.81	-4.47	6.34	-14.06	-8.44	5.62

As shown in Figure 2.8, the choice of the DFT functional results in similar trends for all systems; Grimme's functional including dispersion, B97D, as well as PBE and TPSS functionals, which are at GGA and meta-GGA level, respectively, predict higher values for HOMO as compared to the hybrid functionals, namely, O3LYP, B3LYP, PBE1 and wB97XD. HOMO level predicted by wB97XD is significantly lower than for the other functionals. Likewise, the LUMO predicted by B97D, PBE and TPSS is smaller than those predicted by hybrid functionals, while the LUMO obtained with the wB97XD functional is significantly larger than for other functionals. Therefore, wB97XD gives the largest value of E_{gap} for all systems by a difference in the range between 3-5 eV. E_{gap} calculated by PBE1 is larger than that found with B3LYP by about 0.3-0.4 eV; that for B3LYP is larger

than the value for O3LYP by about 0.4-0.5 eV. In turn, E_{gap} values predicted by PBE and TPSS are smaller than that obtained with B3LYP by 1.0-1.5 eV. B97D, on the other hand, predicts similar values to PBE functional with a difference smaller than 0.05 eV.

Similar calculations on organic photovoltaics based on π-conjugated polymers reported in Ref. [110] suggests the B3LYP is in better agreement with experimental values.[110] In another work, McCormick and coworkers used multiple DFT calculations to model the orbital energies of conjugated polymers. They have shown that for the B3LYP level of DFT, outputs are reasonably well correlated with experimental HOMO energy values (cyclic voltammetry results).[111] Also, as mentioned in Ref. [3], 89% dm HMT-PMBI is a yellow-colored polymer, implying that its HOMO-LUMO gap is expected to lie in the range of 1.9 to 2.3 eV.[112] This means that the result for the HOMO-LUMO gap of HMT-PMBI^{2+} obtained with the B3LYP functional, which is seen to be ~2 eV, as shown in Fig. 2.8, represents a reasonable range for this compound. Moreover, cyclic voltammetry can be used as an experimental approach to determine the HOMO and LUMO energies of the various HMT-PMBI ionenes.[113]

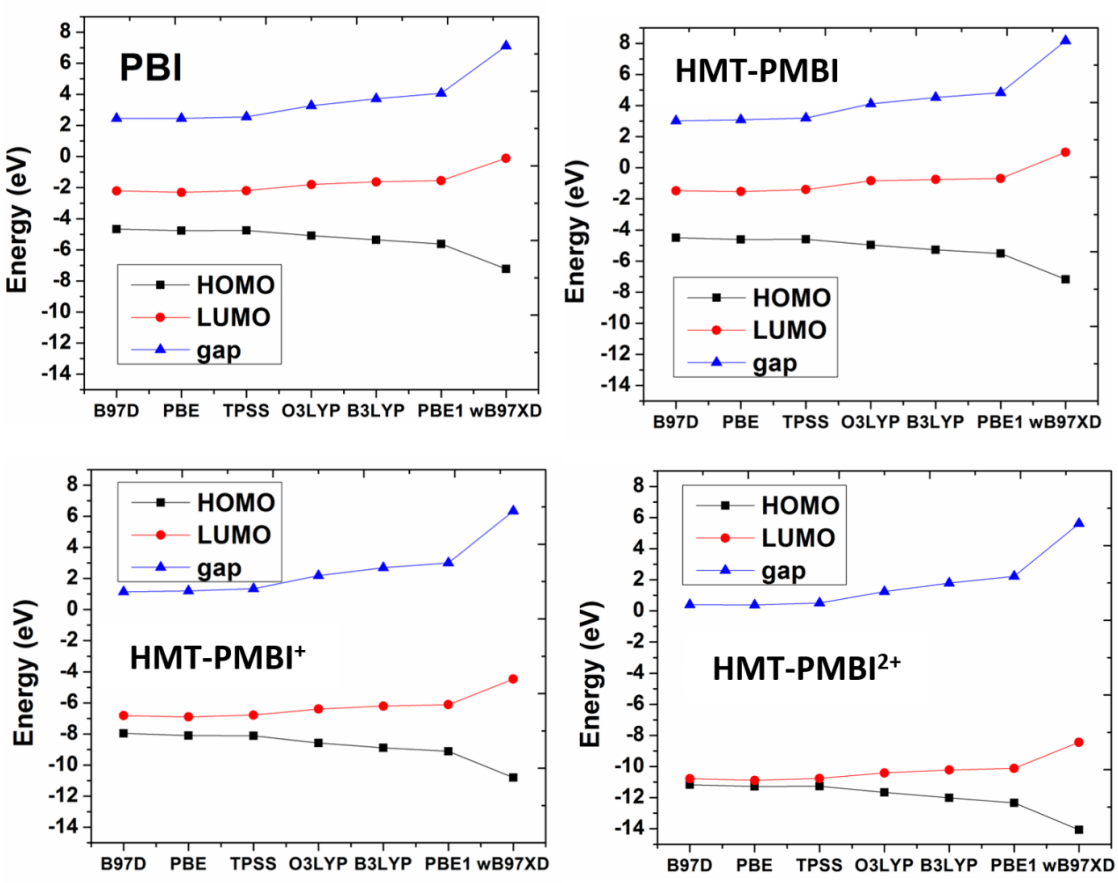


Figure 2.8. The effect of different density functionals on the HOMO, LUMO and band gap of PBI and HMT-PMBI tetramers.

In this chapter, we demonstrated that the ionic charge distribution and electronic structure properties of single ionene moieties can be calculated well with density functional theory. Results indicate an increase of the electrostatic repulsion of ionene moieties in the gas phase with increasing degree of methylation, leading to a stretching of the chains. The presence of water, however, suppresses the electrostatic repulsion among ionene charges. We calculated HOMO, LUMO and electronic band gap of various HMT-PMBIs. More positive charge on the backbone leads to a decrease, by about 2.0 and 3.0 eV, of the band gaps of HMT-PMBI^{+} and HMT-PMBI^{2+} relative to that for the neutral polymer, respectively.

Chapter 3.

Recurrent neural network-based model for accelerated trajectory analysis of MD simulations

This chapter demonstrates the training of recurrent neural networks (RNNs) from distributions of HTM-PMBI atom coordinates in polymer structures that were obtained using Molecular Dynamics simulations. Molecular Dynamics trajectory output data, obtained from Ref. [23], on the HMT-PMBI^{2+} tetramer structure was used as a machine learning training input data for RNNs. This problem is treated as a multi-variate time-series problem. By referring interactions between atoms over the simulation time to temporary correlations among them, RNNs find patterns in the multi-variate time-dependent data, which enable forecasting trajectory paths. Two types of RNNs, namely gated recurrent unit and long short-term memory networks, are considered. The model is described and compared against a baseline MD simulation on a single tetramer of fully methylated HMT-PMBI. Findings demonstrate that both networks can potentially be harnessed for accelerated statistical sampling in computational materials research.

3.1. Theoretical background

3.1.1. Regression fit

Curve fitting is the process of stating the model which offers the best fit to the curves in described by a given dataset. The dataset is decomposed into two variables, first the observed data *X* and second a target variable *Y* which is going to be predicted. The purpose of regression fit is to find the relationship between these two independent variables and thereafter predict the unseen target variable. Linear Regression is considered as a machine learning (ML) method based on supervised learning. In supervised learning, the mapping function is learned from the input to the output data, using a specific algorithm.[114,115]

Given the training set $\{(x_1,t_1), \ldots ,(x_N,t_N)\}$, with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^D$, the simplest linear model for regression can be written as

$$y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D, \quad 3.1$$

where x_i is input training data and y is the dependent output. Target variable is denoted by t_i. The goal here is to learn and find the best set of coefficients *w* so that $y(x, w)$ aligns with target value in training data.[116]

Since having a model that is linear in parameter *w* is what will be important for simple algorithms, Equation 3.1 can be extended to include non-linear functions, $\phi_i(x)$, of data,

$$y(x, w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_{M-1} \phi_{M-1}(x). \quad 3.2$$

Linear combinations of these basis functions, $\phi_i(x)$, are also linear in parameters w_i. Various non-linear basis function such as polynomial, gaussian, sigmoidal, logarithmic, and sinusoidal/cosinusoidal could be used. Linear Regression fits a model to the dataset by tuning the set of parameters *w* to minimize the loss function (also called cost function), $E(w)$, which is defined as the sum of the squared residuals or errors (observed minus fitted value),[116]

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2, \quad 3.3$$

$$\nabla E(w) = 0. \quad 3.4$$

To solve Eq. 3.4, and get the best set of *w* parameters, the gradient descent (GD) method is employed. The following is a simple algorithm of the GD method:[117]

1. Initialize the parameters *w* with guessed values (all zeros or random values)

2. Update the parameters: $w_{new} = w - \eta \nabla E(w)$

3. Update the learning rate η (gradually decrease over time)

4. Repeat steps 2 and 3 until $\nabla E(w)$ becomes small enough (10^{-3} – 10^{-4}).

3.1.2. Neural networks

Neural networks (NNs) are a specific set of methods that has dramatically altered the machine learning area. The artificial neural networks are inspired by biological neural networks. The primary works of Rosenblatt and McCulloch *et al*. in the mid 20th century led to the creation of the perceptron or an artificial neuron.[118,119]

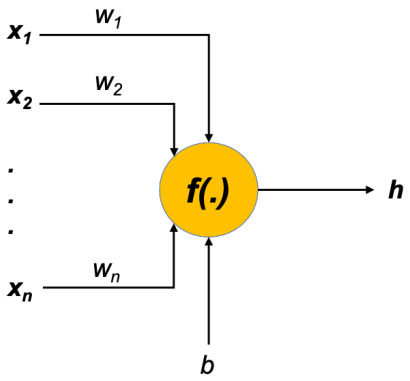


Figure 3.1. A single neuron. x_i, w_i, b, h, and f denote the inputs, weights, bias term, hidden state (or the output of neuron) and activation function, respectively.

The neurons (Fig. 3.1) are the basic unit of the neural network architectures. The neuron is a placeholder for a mathematical function which takes all the inputs (x_i), multiplies them by the corresponding weights (w_i), and adds them up ($\sum_{i=1}^{n} x_i * w_i$); a bias term (b) is often added (Equation 3.5). The weight parameters define the strength of the connection between neural network nodes (see Figure 3.2). For instance, if w_1 has greater magnitude than w_2, that means the neuron 1 has greater impact over the neuron 2.[120] To generate the output value of the neuron, also called hidden state (h), a non-linear activation function (f) is applied. The equation below represents the process of calculating the hidden state,

$$h = f(b + \sum_{i=1}^{n} w_i x_i). \qquad 3.5$$

The set of weights $W = \{w_1, w_2, \ldots, w_n\}$ is going to be trained to detect and learn a pattern among the inputs. The output of one neuron could be the input for another, which generates a network of neurons, or a “neural network."

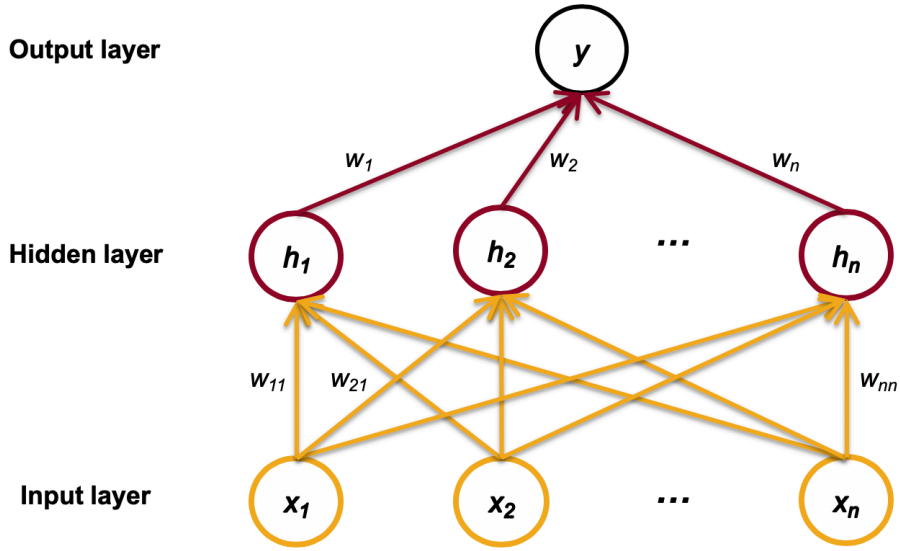


Figure 3.2. A fully connected neural network diagram with one hidden layer corresponding to Eq. 3.5. The input, hidden and output states are represented by nodes, and the weight parameters, w_i, are depicted by links between the nodes.[7]

As it is shown in Figure 3.2, input, hidden and output nodes are organized into layers and these layers make up a connected network. The value of the output, *y*, in this figure can be calculated using Equation 3.5:

$$y = f_2(\sum_{i=1}^{n} w_i\, h_i) = f_2\big(\sum_{i=1}^{n} w_i\, f_1(\sum_{j=1}^{n} w_{ij}\, x_j)\big), \qquad 3.6$$

where f_1 and f_2 are activation functions. The sigmoid function, $f(a) = \frac{1}{1+\exp(-a)}$, is a common example of activation function for binary classification problems. For Recurrent Neural Networks (see section 3.1.3), however, the hyperbolic tangent function, $tanh(a) = \frac{1-exp(-2a)}{1+exp(-2a)}$, is preferred as a standard activation function.[7] Here, parameters {w_{11}, w_{21}, …, w_{nn}} represent the connection between input and hidden layer nodes and set of parameters {w_1, w_2, …, w_n} define the connection between hidden and output layer nodes.

To optimize a neural network, the gradient descent algorithm (Equations 3.3 and 3.4) is usually employed. As discussed in the previous section, a cost function, also called

loss function, $E(w)$, measures how incorrect the output $y(x_n, w)$ of the network is compared to the target output t_n. It is defined by

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2. \quad 3.7$$

For every loop on the training data, each weight is updated as

$$w_i^{new} = w_i - \eta * \nabla w_i, \quad 3.8$$

where η is a dimensionless factor called learning rate. The partial derivative of the loss function with respect to the corresponding weights is shown by

$$\nabla w_i = \frac{\partial E(w)}{\partial w_i}. \quad 3.9$$

To train a machine learning model the aforementioned loss function must be minimized to the desired error. The best-known technique to calculate the partial derivative of the loss function with respect to the corresponding weights (Equation 3.9) and updating weights (Equation 3.8) is the backpropagation method proposed by Rumelhart *et al.*.[121] A common problem occurring with backpropagation and updating weights is that the gradient (∇w_i) becomes vanishingly small[122], thus, preventing the weight from changing its value. To address this problem, which specifically happens in recurrent neural networks, the Long Short-Term Memory (LSTM) model[123] was proposed which will be elaborated in the next section.

Overfitting refers to an outcome that predicts the training data with high precision, however, has a low accuracy on a test dataset. To avoid overfitting, various regularization techniques can be used. One of the most common methods which has been employed in this study is L2 regularization,

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\| w \|^2, \quad 3.10$$

where $\| w \|^2 = \sum_{i=1}^{n} w_i^2$ and λ is the regularization parameter. L2 regularization adds a penalty term, $\frac{\lambda}{2}\| w \|^2$, equal to the square of the magnitude of weights. The hyperparameter λ, adjusts the trade-off between obtaining low training loss and obtaining less complex model which is often selected by trial and error work. The neural network

structure is determined by hyperparameter variables. These variables must be defined before training the algorithm and optimizing the weights. Here I have introduced all hyperparameters that have been employed in our study:

The **learning rate** (Equation 3.8) is the most important hyperparameter, which must be set first. The learning rate describes how fast a network updates its parameters. Small learning rate would lead to a more reliable training; however, the optimization process (satisfying Eq. 3.4) would take longer as the steps towards the minimum of the loss function are small (~10^{-5}).

The **number of epochs** is the number of times the entire training examples are passed through the network. The number of epochs needs be as large as possible, but the training accuracy should be inspected to prevent a possible overfitting.[7,117]

The **batch size** is the number of subsamples propagated through the networks after which neural network weights get updated. If the size of samples is equal to 2000 and the batch size is equal to 200, then the model grabs the first 200 samples and trains the network with them. Following that, the next 200 samples (from 201st to 400th) will be taken and the network gets updated. This process will be continued until all samples in the training dataset broadcast to the network.[117]

The **sequence length** is length of the sequence that will be learned by the model and is being used for recurrent neural networks (Table 3.1). The default value of the sequence length is equal to the total length of the dataset dived by the batch size. A change in the sequence length will not affect the trained model considerably.[7]

3.1.3. Recurrent neural networks for time series forecasting

The time series forecasting problem is a salient area in machine learning, which attempts to model stochastic processes as well as to forecast the future subsequent values of sequences based on the previous records of those sequences. As it is shown in Figure 3.3, the time series forecasting/prediction problem is reflected as a supervised learning problem (see section 3.1.1).

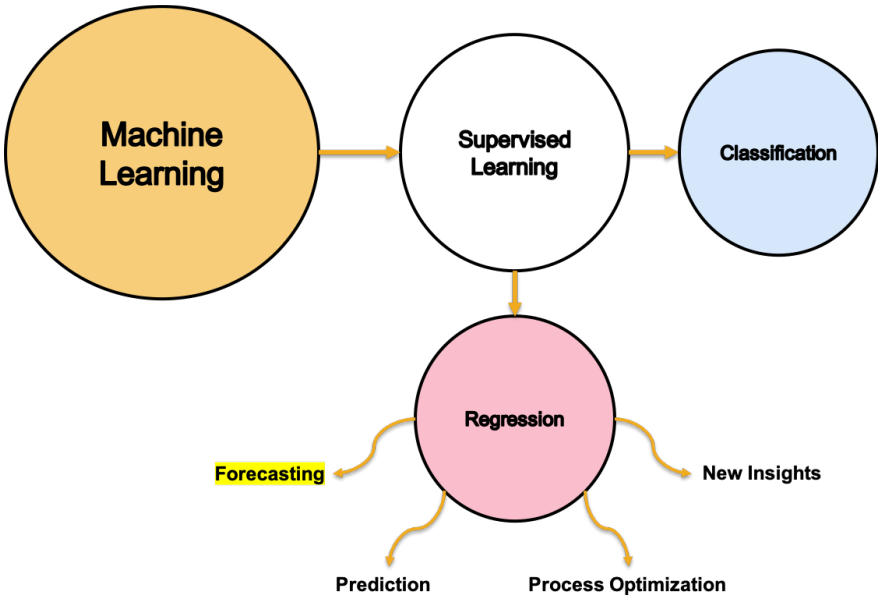


Figure 3.3. A simple schematic of the supervised machine learning categories and applications.[8]

A robust type of algorithms that can handle the time component complexity are Recurrent Neural Networks, which are designed to manage sequence dependency of input data.[117] RNNs receive as input a sequence of signals with no predetermined size limit and utilize their internal states (so-called as memories) for processing the input. In other words, RNNs iterate over the timesteps of a sequence, while preserving an internal state that keeps information about the timesteps it has already seen. The LSTM model, as a special kind of RNN, was proposed by Sepp Hochreiter *et al*. in 1997,[123] to solve the vanishing gradient issue associated with RNNs.[124,125] The problem is that the gradient or partial derivative of the cost function with respect to the layer's weight, w_i, (Equation 3.9) becomes vanishingly small, preventing the weights from updating their values.[122]

To address this problem, LSTMs employ a gating mechanism, composed of three gates, which give the model permission to pass or forget data, as shown in Figure 3.4.[123] The proposed gates establish a new relationship between data and forget previous relationships over a particular time span. Thus, the gating mechanism enables the previous input signals to affect the current signal at a given time, while remaining unaffected by signals that are far apart from the current signal. Gated Recurrent Unit (GRU) is a form of LSTMs introduced in 2014.[78] In what follows, we only present the architecture of a GRU, which is similar to that of a LSTM model.

Figure 3.4 represents a unit of a GRU model. In this figure *r* and *u* are called reset gate and update gate, respectively, which are the main gates in a GRU unit. Concatenating the new input with the preceding memory is enabled by the reset gate. The update gate, on the other hand, determines the portion of the preceding memory to be kept. We denote the hidden state of the GRU at time step *i* by h_i'. In a GRU structure, the activation h_i' at time *i* is a linear interpolation between the previous activation h_{i-1}' and the candidate activation $\widetilde{h_\iota}$ (see Equation 3.5). An update gate u_i governs this linear interpolation, and the candidate activation $\widetilde{h_\iota}$ depends on a reset gate r_i. The exact formulation of this process is

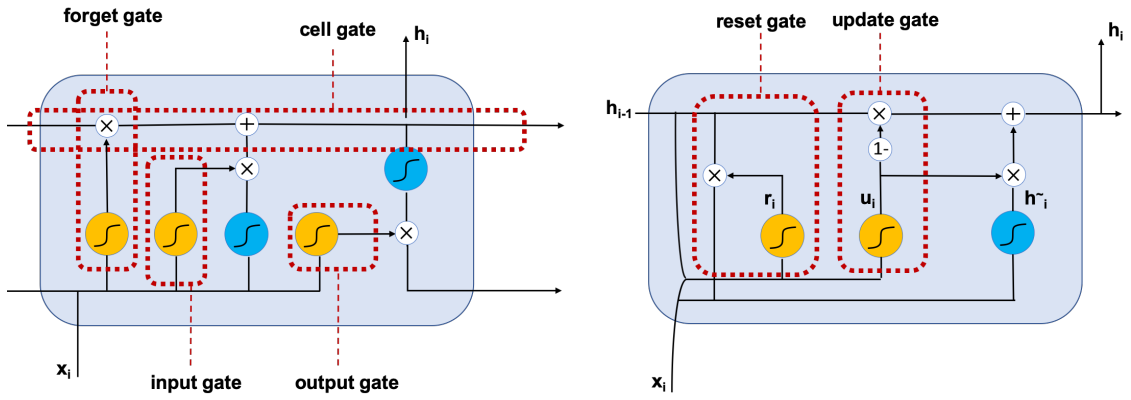
$$h_i' = (1 - u_i)h_{i-1}' + u_i\widetilde{h_\iota}, \qquad 3.11$$

$$u_i = \sigma(W_u x_i + U_u h_{i-1}'), \qquad 3.12$$

$$\widetilde{h_\iota} = \tanh\big(\mathrm{W}x_i + U(r_i \odot h_{i-1}')\big), \qquad 3.13$$

$$r_i = \sigma(W_r x_i + U_r h_{i-1}'). \qquad 3.14$$

Here, vector x_i represents the input. The update gate u_i in Equation 3.12 decides how much of each dimension of the hidden state is retained and how much should be updated with the transformed input x_i from the current time step.



Long Short-Term Memory (LSTM) **Gated Recurrent Unit (GRU)**

Figure 3.4. The LSTM and GRU units. Yellow and blue circles represent sigmoid and tanh functions (see 3.1.2 section), respectively. White circles show point-wise operations.

3.1.4. Molecular dynamics simulation

Molecular dynamics (MD) is one of the most powerful tools utilized for the simulation of complex molecular systems and nowadays routinely applied to various systems such as biomolecules, polymers, and solid-state materials.[93] Prediction of the trajectory of atoms, finding the structure and dynamics of macromolecules, and calculating the thermodynamic properties of novel compounds without the need to conduct experiments, can all be done using atomistic molecular dynamics simulations. Figure 3.5 represents a basic scheme of the all-atom MD algorithm.[126]

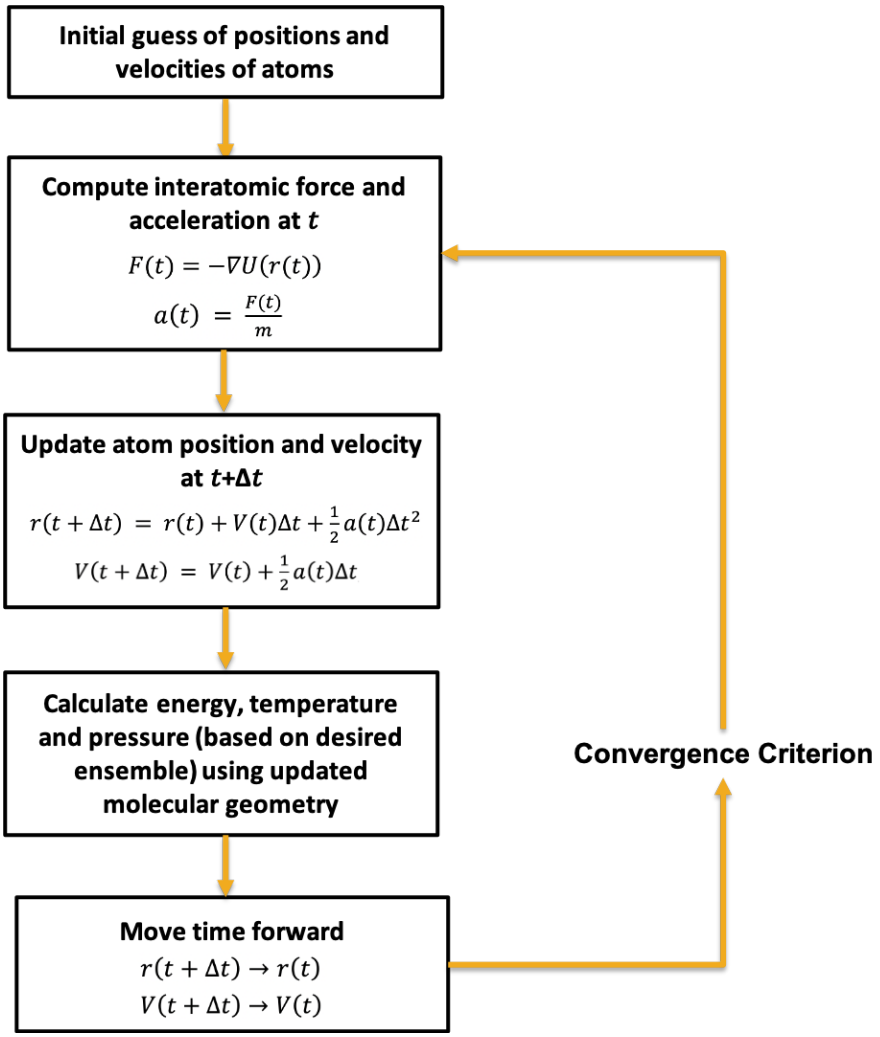


Figure 3.5. Schematic of a standard Molecular Dynamics simulation algorithm.

The loop in Figure 3.5 above will be continued until equilibrium is reached (equilibration time). Thereafter, simulations will be continued in the so-called production

run to sample uncorrelated system configurations for the calculation and analyses of thermodynamic-statistical properties and generating dynamical trajectories of molecular processes in the system.

Force field parameters used in computing interatomic potential energies (see Fig. 3.5) are taken either from *ab initio* or semi-empirical quantum mechanical calculations. Parameters could be obtained by fitting against calculations of electronic structure or experimental properties gained from elastic constants, lattice parameters and spectroscopic assessments.[127]

Below are the energy terms for the OPLS-all atom (OPLS-AA) force field,[128,129]

$$E = E_{bonds} + E_{angles} + E_{dihedrals} + E_{non-bonded} \quad 3.15$$

$$E_{bonds} = \sum_{bonds} k_r (r - r_0)^2, \quad 3.16$$

$$E_{angles} = \sum_{angles} k_\theta (\theta - \theta_0)^2, \quad 3.17$$

$$E_{dihedrals} = \sum_{dihedrals} \left(\frac{K_1}{2} [1 + cos(\phi)] + \frac{K_2}{2} [1 - cos(2\phi)] + \frac{K_3}{2} [1 + cos(3\phi)] + \frac{K_4}{2} [1 - cos(4\phi)] \right), \quad 3.18$$

$$E_{non-bonded} = \sum_{i>j} f_{ij} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right), \quad 3.19$$

where $A_{ij} = \sqrt{A_{ii} A_{jj}}$ and $C_{ij} = \sqrt{C_{ii} C_{jj}}$.

The OPLS (optimized potentials for liquid simulations) force field[128] is a widely employed force field (FF) for Molecular Dynamics (MD) simulations of liquids and polymer solutions. The energy between two covalently bonded atoms is described by E_{bonds}. The harmonic approximation of E_{bonds} works well close to the equilibrium bond length. However, the error of this expression grows with the separation distance between atoms. Changing the geometry of electron orbitals involved in covalent bonding will change the E_{angles}, which describes interactions between 3 bonded atoms. Equation 3.18 shows the dihedral angle energy term. A dihedral angle (also called torsion angle) for four bonded atoms (A-B-C-D) is the angle between planes through two sets of three bonded atoms

(composing of the two sets of atoms A-B-C and B-C-D), having two atoms (atoms B and C) shared. $E_{non-bonded}$ is the energy term for the non-bonded interactions between atom pairs, which consist of two separate terms. The first two terms under the sum on the right-hand side of Equation 3.19 correspond to the van der Waals (vdW) attraction and a strong repulsion at close distance, described with the 12-6 Lennard-Jones potential. The third term represents the Coulomb energy due to electrostatic interaction between charged atoms.[130] As Molecular Dynamics is only used for finite systems, periodic boundary conditions (PBCs) are employed to approximate an infinite system that simulates the bulk properties. PBCs imply that any atom, which leaves the simulation box by one face, will re-enter the simulation box instantaneously by the opposite face.[131]

MD results reported in Ref. [23] and [81] were obtained using the OPLS-AA force field (Equations 3.15-3.19) and the simulations were conducted for an NVT ensemble,[131] using Nose-Hoover thermostat, where the number of particles (N), the volume of the simulation cell (V), and the temperature of the system (T) kept constant. A Nose-Hoover thermostat can be represented by modifying equations of motion,[132]

$$m_i \frac{d^2 r_i}{dt^2} = f_i - m_i \gamma(t) \frac{dr_i}{dt}, \quad 3.20$$

$$f_i = -\frac{\partial}{\partial r_i} U_i(r_1, r_2, \ldots, r_N), \quad 3.21$$

where N denotes the number of particles. m_i , r_i , U_i and f_i are the mass, position, total potential energy and the total force acting on atom *i*.

Term $\gamma(t)$ on the right hand-side of Equation 3.20 demonstrates the thermodynamic friction coefficient and is given by

$$\frac{d\gamma(t)}{dt} = \frac{1}{Q}\left[\sum_i \frac{m_i}{2}\left(\frac{dr_i}{dt}\right)^2 - \frac{(X+1)}{2} k_B T\right], \quad 3.22$$

where k_B and Q constants are the Boltzmann constant and the thermostat mass (the artificial mass of the thermostat particle). X represents the number of degree of freedom of the system and is equal to $3N$, where N is the number of particles in the system.[133]

3.2. Computational details

As explained in the previous part, all-atom MD simulations in Ref. [23] were carried out to study a single tetramer of 100% dm HMT-PMBI, using the LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) program. LAMMPS is an open source software written in C++ language provided by Sandia National Laboratories.[134] Trajectory history comprised of 10 ns of X, Y and Z coordinates for all atoms were extracted from the LAMMPS trajectory output file to prepare the input file for neural network training. No missing data points were present among XYZ coordinates as expected from MD simulations.

We used Python 3.6 programming language as well as Keras-TensorFlow 2.1 framework[135] in Python to implement RNNs. TensorFlow is an open source library to support developing and training ML models in Python. TensorFlow was created and released by a Google team in 2015. The full Python code was uploaded in my GitHub and also attached in the Appendix A (GitHub link: https://github.com/Mehrdad93/Trajectory-path-prediction/blob/master/gru_trj_path-2.py). All Python codes were run on Google Colaboratory. Google Colaboratory, in short Colab,[136] is an open source product from Google Research, which enables all users to write and execute python scripts using the browser. Colab also allows everyone to take advantage of free access to computing resources including GPUs. Employing Tesla T4 GPUs on Colab platform takes about 40 mins of continuous run time to train both GRU and LSTM models, which is much less than a typical MD simulation for this system.

Before training the neural network, hyperparameters must be determined (see section 3.1.2 and Table 3.1). Our MD simulation trajectory dataset is split into two parts: a train and a test set. 80% of the dataset will be used for the train set. C2-position carbon, CM2 atom, also highlighted in Figure 3.6 in HMT-PMBI^{2+} tetramer, is chosen as target atom for prediction purposes. The entire dataset has been normalized using Normalizer function to scale individual samples to have unit norm. In this study, L2 normalization (see section 3.1.2) was applied to each sample (if each sample squared and summed with others, the total would be equal to one). The main purpose of normalizing the data is to maximize the performance of the neural network, by smoothing the optimization process.[137]

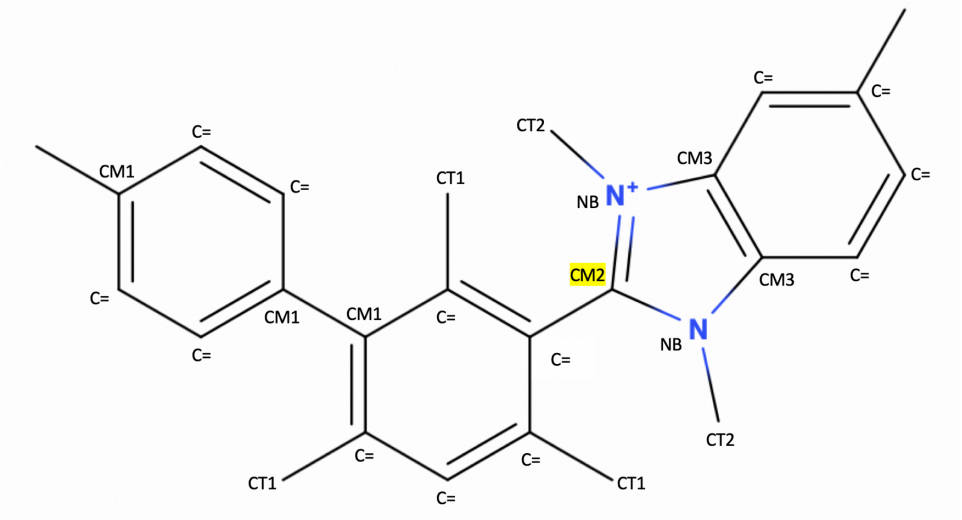


Figure 3.6. Half monomer of HMT-PMBI^{2+}. C2-position, highlighted CM2 atom type, is chosen as target atom for prediction. Structure by "molview" labeled with the atom types based on OPLS-AA force field.[9]

Various batch sizes were tried and 256 is selected as the default value. As discussed in 3.1.2 the frequency of updating the neural net weights can be controlled with the number of batch sizes. The sequence length of 150 was employed, which means that each random sequence of samples is comprised of 150 timesteps of each atom. The number of GRU and LSTM units was set to 250, which resulted in 250 output vectors (one output vector for each of GRU/LSTM units). Therefore, a fully connected layer, also called dense layer, was added to reduce 250 outputs down to 3 output vectors, which are the predicted X, Y, and Z coordinates of the target atom (highlighted CM2 atom in Figure 3.6). To compile the model and train the network, the learning rate began with 10^{-3} and decreased to 10^{-4} if needed. The epoch number was fixed at 500, which shows the number of complete passes through the training dataset. To avoid overfitting, lowering the training error and increasing the test error, L2 regularization (Equation 3.10) with the regularization parameter of 10^{-3} was utilized.

Table 3.1. List of hyperparameters

Hyperparameters	Min	Max	Default Value
Learning rate	10^{-4}	5×10^{-3}	10^{-3}
Batch size	128	512	256
# units for GRU and LSTM	200	300	250
Sequence length	50	200	150
Number of epochs	500	500	500
L2 regularization constant	0.0001	0.001	0.001
Train-test split	80%-20%	90%-10%	80%-20%

3.3. Results and discussion

Figure 3.7 shows the proposed GRU/LSTM-based model for MD trajectory path predictions. RNN-based models are chosen because in general their input can involve sequences of arbitrary lengths, which refers in this case to the varying number of atoms. The model consists of a GRU or LSTM layer and a subsequent dense layer to combine output from the previous layer. As input, the RNN-model receives trajectory data for all atoms in the unit-cell. As output, we aim to predict the X, Y, Z coordinates of the considered atom *i*. The predictions are then compared with the MD calculated outputs.

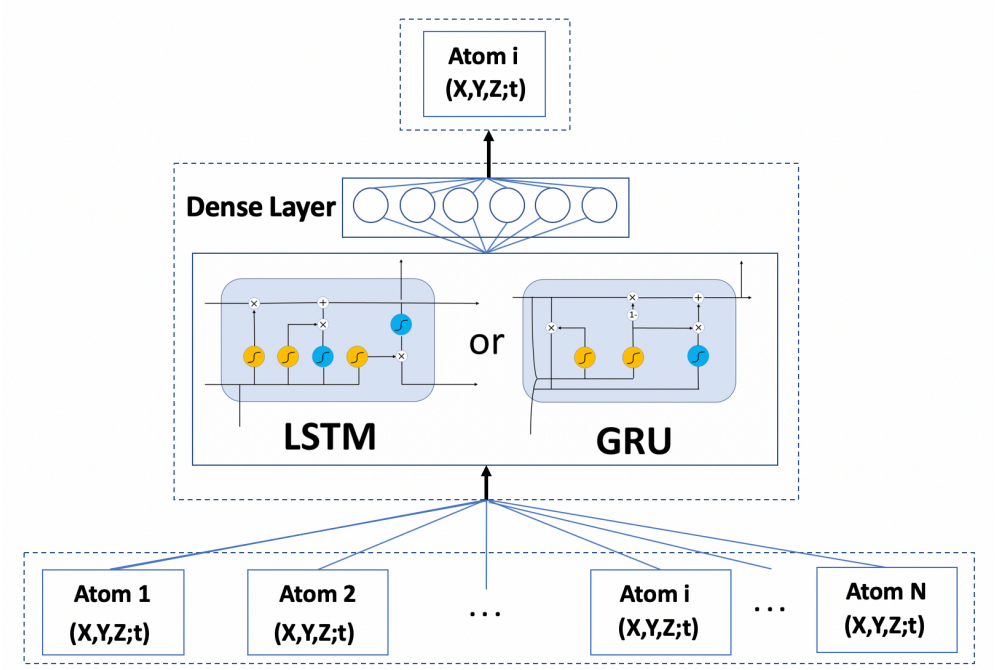


Figure 3.7. The proposed GRU/LSTM-based model for MD trajectory prediction.

We resampled the target data for the highlighted CM2 atom in Figure 3.6 by shifting the results of X,Y and Z coordinates of the target atom negatively by 500 ps (around 5% of the train set); meaning that we shifted the target data points to a configuration 500 ps in the past to let the model predict the future trajectory of the test atom. We used 80% of the MD time steps for the training set and the rest for the test set. The range of feature values was scaled in the data preprocessing step. To train the RNN, we created batches of shorter sequences of input data selected randomly from the train set. RNN was created in Keras-TensorFlow library (see section 3.2) and GRU or LSTM were added to the first layer. This layer returns a sequence of 250 values as input for the dense layer which in turn outputs 3 vectors (see Figure 3.7). The sigmoid activation function was used to output the values between 0 and 1 which is justified as we expect the output values to be in the same range as the training set. A gradient descent optimizer (Equations 3.3-3.4) was used to compile the model. The values for the hyperparameters used are shown in Table 3.1.

As for the loss function, we used the mean squared error (MSE) that must be minimized during the model training,

$$MSE = \frac{\sum_{i=1}^{n}(y_i - y_i^p)^2}{n}. \qquad 3.23$$

Figure 3.8 depicts the predicted output sequences, namely X, Y, Z coordinates of the GRU- and LSTM-based models compared with the ground truth from MD simulation. As shown for the train set, both models were able to predict the oscillations of the coordinates with reasonable accuracy which was expected as this data was seen by the models during the training. However, the prediction is not accurate for the first few time steps as the models have not received enough input data to be trained. The overall fluctuations were also projected reasonably well, although the peaks were sometimes inaccurate, especially for X and Y coordinates of the LSTM model. As for the test set, the data were not seen by the models during training process. In this case, the GRU model performs reasonably well. However, the LSTM model could not completely capture the peaks and the peaks were not matched with the MD data and there seems to be a deviation with maximum relative error of about 3-4% (see Figure 3.10). The relative error in this study is equal to the absolute error (MD results – ML outputs) divided by the known value (MD results).

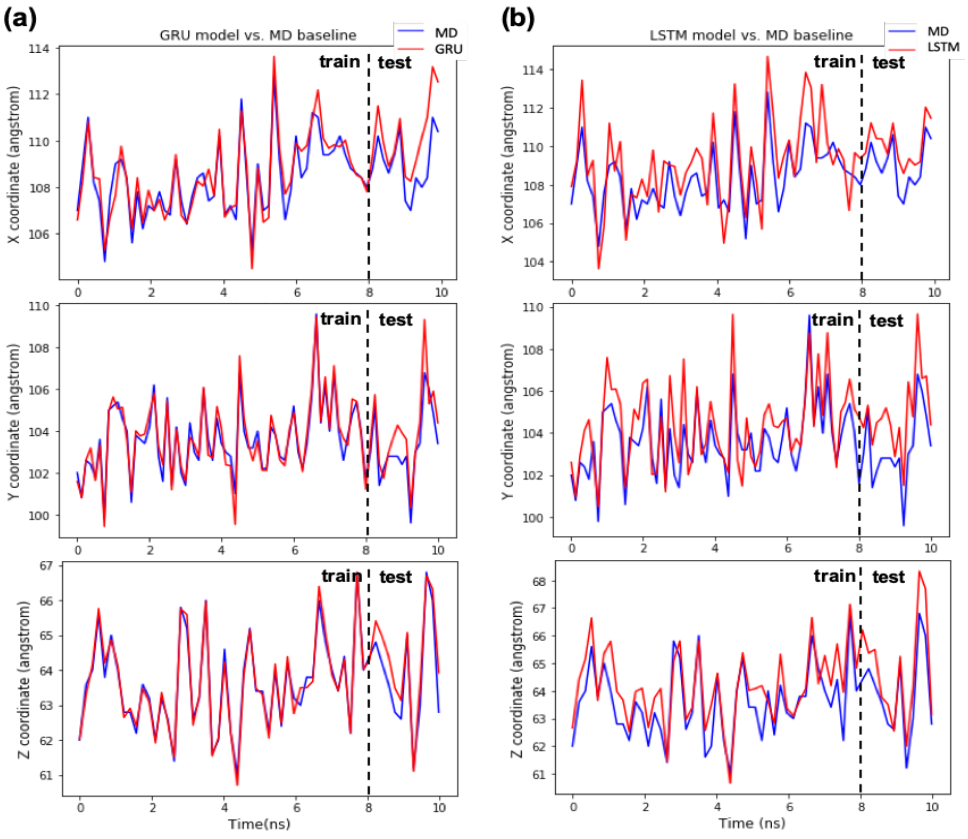


Figure 3.8. Predictions of the trajectory of CM2 atom, by (a) GRU and (b) LSTM compared to MD outputs.

Figure 3.9 compares the generated partial autocorrelation function (PACF) of the X coordinate of C2-position (target) atom calculated with MD simulations with those predicted by GRU and LSTM models. In the time series analysis, the partial autocorrelation function indicates the correlation between current and previous series values and measures which of the previous series values are most helpful for forecasting future behaviours. If a dependent variable Y has 3 independent variables {X_1, X_2, X_3}, Equation 3.24, then the partial autocorrelation function between Y and X_3 will be calculated as follows:[138]

1. Perform regression, in which variable Y is predicted from X_1, X_2 and then calculate the residuals of this regression (residual for each sample is the difference between the predicted value and the observed value for that sample).
2. Perform regression in which variable X_3 is predicted from X_1, X_2 and then calculate the residuals of this regression.
3. Use Equation 3.25 to determine the value of partial correlation between Y and X_3.

$$Y = a_0 + a_1X_1 + a_2X_2 + a_3X_3, \tag{3.24}$$

$$PACF(Y, X_3) = \frac{Covariance(Y, X_3|X_1, X_2)}{\sqrt{Variance(Y|X_1, X_2) \times Variance(X_3|X_1, X_2)}}, \tag{3.25}$$

where Variance is the difference between expectation of a squared variable and the expectation of that variable squared, $E(XX) - E(X)E(X)$; and Covariance is the same concept as Variance but compares two variables, $E(XY) - E(X)E(Y)$, which implies the relationship between two random variables X and Y.[139]

As shown in Figure 3.9, the PACF outputs from MD simulation and the proposed RNN-based models show that above around 30 ps (0.03 ns) the values become uncorrelated since all the correlation values lie in the 95%-confidence interval (blue boxes shown in Fig. 3.9). The 95%-confidence interval is equal to $\frac{1.96}{\sqrt{n}}$, where n is the number of samples. This figure displays the ability of GRU and LSTM models to produce uncorrelated configurations which can be used directly for the calculation of thermodynamic properties.[140]

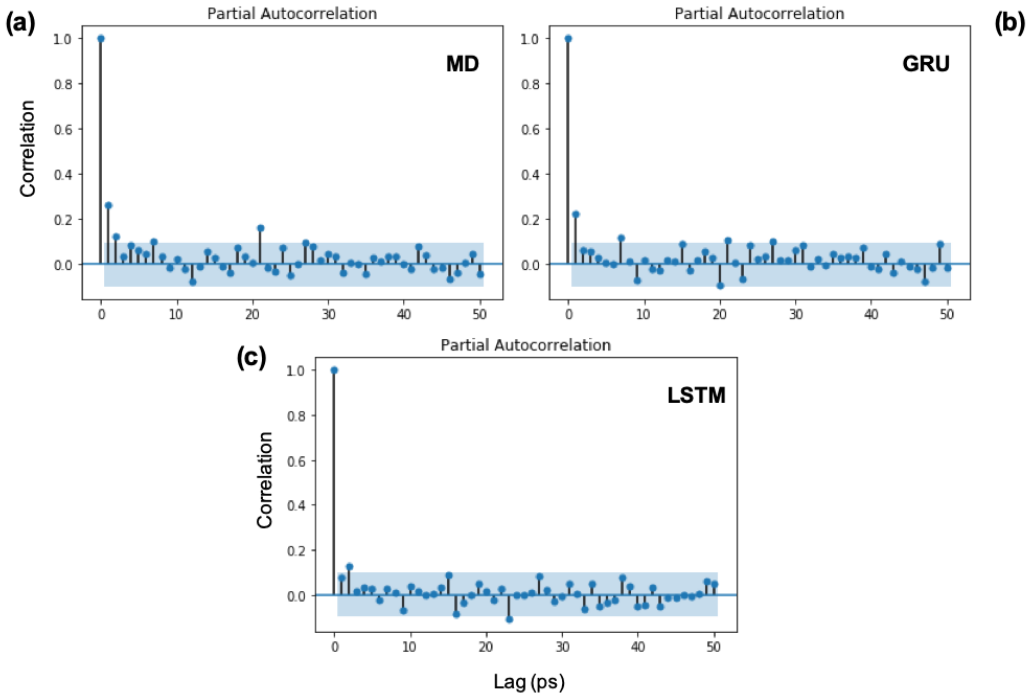


Figure 3.9. Partial autocorrelation function for the X coordinate of the target atom obtained from (a) MD, (b) GRU and (c) LSTM results with 95%-confidence interval shown in blue boxes.

In general, the performance of GRU model seems to be superior to the LSTM model. This is shown in Figure 3.10, by calculating the point-by-point relative percent error $(\frac{|y_i - y_i^p|}{y_i} \times 100)$ of GRU and LSTM models against the MD simulation. For both the train and test sets, the percent error of the GRU model for X, Y, Z coordinates was found to be less than 2.5%, with a mean error of around 1.1%. Regarding the LSTM model, the maximum percent error for the coordinates was found as around 3.5% with the mean error of around 1.6%. Even though the calculated error implies a slightly more accurate prediction for the GRU model, LSTM seems to be more robust in terms of dealing with the overfitting issue as the error values for both train and test sets lie in a similar range.

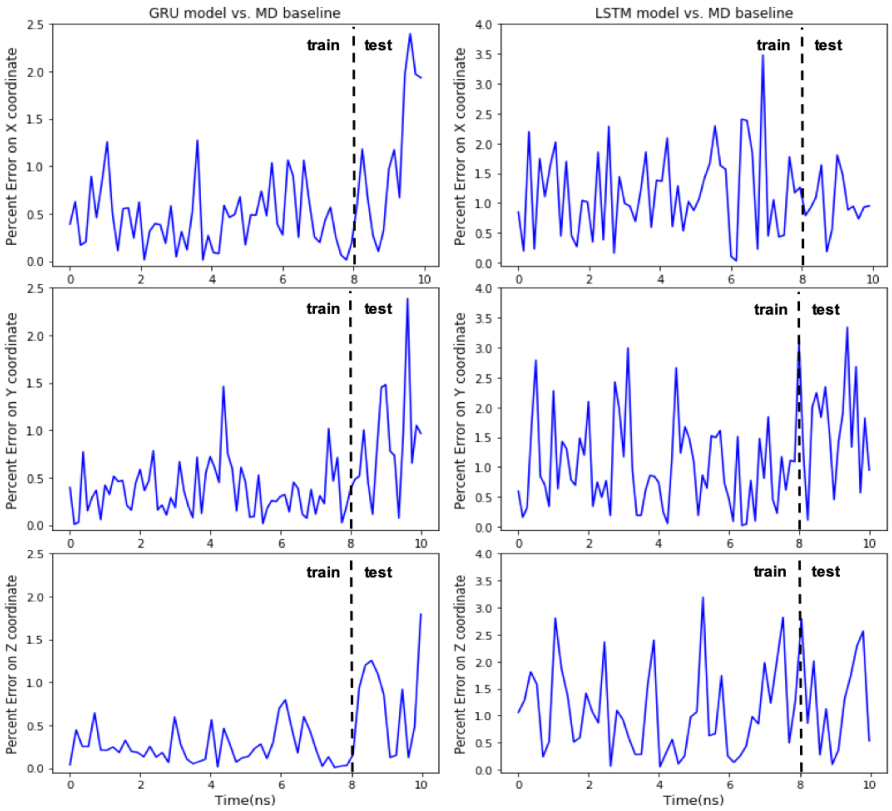


Figure 3.10. Relative percent error of the two studied models.

Overall, employing either GRU or LSTM models result in over 96% accuracy. This means that RNN-based algorithms including “Long Short-Term Memory” and “Gated Recurrent Units” models are able to forecast the trajectory paths of HMT-PMBI tetramer with a similar accuracy as molecular dynamics simulations, but in a much shorter time. Using Tesla T4 GPUs, it takes about 40 mins of continuous run time to train both GRU and LSTM models. Using recurrent neural networks can reduce the time for simulation of a molecular trajectory by a factor of 50-100, depending on the model parameters and the size of the system. In this study, two important hyperparameters mentioned in Table 3.1, the number of units for GRU and LSTM networks and the number of epochs, are playing a huge role in determining the overall runtime. Default values for the number of units for GRU and LSTM networks and the number of epochs was set to 250 and 500, respectively, which took 30-45 mins to train the networks and forecast the trajectory path of atoms. However, simulating a single HMT-PMBI tetramer using MD simulation can take up to 2 days of continuous run, which is significantly higher than machine learning methods such as recurrent neural networks.

Chapter 4.

Conclusion and future work

The first part of this work focused on the conformational properties of HMT-PMBI oligomers. We demonstrated that the ionic charge distribution and electronic structure properties of single ionene moieties can be calculated with density functional theory. Results indicate an increase of the electrostatic repulsion of ionene moieties in the gas phase with increasing degree of methylation, leading to a stretching of the chains. The presence of water, however, suppresses the electrostatic repulsion among ionene charges. We calculated HOMO, LUMO and electronic band gap energies of various HMT-PMBIs. More positive charge on the backbone leads to a decrease, by about 2.0 and 3.0 eV, of the band gaps of HMT-PMBI^{+} and HMT-PMBI^{2+} relative to that for the neutral polymer, respectively. In the next part, this study suggests that accelerated statistical sampling of a polymeric system can be performed with “Long Short-Term Memory” and “Gated Recurrent Units” models. That means the RNN-based models can be employed as complementary tools for molecular dynamics simulations in order to markedly reduce computational costs.

Studying the single chain behavior is a prerequisite for studying self-organization in highly concentrated solutions of polybenzimidazole ionenes. Studies performed and reported here reveal important trends in this regard and they could thus form the basis for further modeling and simulation of ionene self-aggregation, network formation, ion and solvent transport, and the development of a statistical model of fracture formation in ionene-based membranes. Insights on the conformational properties of ionenes with varying degree of methylation are important for better understanding the charge transport behavior in ionene solutions. We hope this study will prompt further fundamental investigations on conformational properties of benzimidazolium compounds.

Moreover, in future studies the proposed RNN-based models could be tested on more complicated systems, e.g., predicting the bundle formation or self-aggregation of multiple oligomers. Furthermore, the architecture of the proposed models can be improved by employing the attention mechanism in the recurrent neural network models. Attention-based RNNs emphasize on specific fragments of the input sequence when forecasting the

output sequence, facilitating simpler learning and higher performance. Attention will be considered as another component of the architecture of the RNN model which maps the relevant and important segments of the input data to their output values.[141] In our study, atoms that are neighbor to the target atom will be the most important and relevant segments of the input data.

References

(1) Dekel, D. R. Review of Cell Performance in Anion Exchange Membrane Fuel Cells. *Journal of Power Sources* **2018**, *375*, 158–169.

(2) Park, E. J.; Kim, Y. S. Quaternized Aryl Ether-Free Polyaromatics for Alkaline Membrane Fuel Cells: Synthesis, Properties, and Performance--a Topical Review. *Journal of Materials Chemistry A* **2018**, *6* (32), 15456–15477.

(3) Wright, A. G.; Fan, J.; Britton, B.; Weissbach, T.; Lee, H.-F.; Kitching, E. A.; Peckham, T. J.; Holdcroft, S. Hexamethyl-p-Terphenyl Poly (Benzimidazolium): A Universal Hydroxide-Conducting Polymer for Energy Conversion Devices. *Energy & Environmental Science* **2016**, *9* (6), 2130–2142.

(4) Haunschild, R.; Barth, A.; Marx, W. Evolution of DFT Studies in View of a Scientometric Perspective. *Journal of cheminformatics* **2016**, *8* (1), 52.

(5) Shan, N.; Zhou, M.; Hanchett, M. K.; Chen, J.; Liu, B. Practical Principles of Density Functional Theory for Catalytic Reaction Simulations on Metal Surfaces--from Theory to Applications. *Molecular Simulation* **2017**, *43* (10–11), 861–885.

(6) Payne, M. C.; Teter, M. P.; Allan, D. C.; Arias, T. A.; Joannopoulos, ad J. D. Iterative Minimization Techniques for Ab Initio Total-Energy Calculations: Molecular Dynamics and Conjugate Gradients. *Reviews of modern physics* **1992**, *64* (4), 1045.

(7) Bishop, C. M. *Pattern Recognition and Machine Learning*; springer, 2006.

(8) COGNUB Decision Solutions. COGNITIVE COMPUTING AND MACHINE LEARNING http://www.cognub.com/index.php/cognitive-platform/.

(9) Bergwerf, H. MolView: An Attempt to Get the Cloud into Chemistry Classrooms.

(10) R. S. Khurmi and R. S. Sedha. *Materials Science*, 5th ed.; S. Chand & company, New Delhi, 2014.

(11) Gold, V. *Compendium of Chemical Terminology*; Blackwell scientific publications, 1987.

(12) Wang, Y.-J.; Long, W.; Wang, L.; Yuan, R.; Ignaszak, A.; Fang, B.; Wilkinson, D. P. Unlocking the Door to Highly Active ORR Catalysts for PEMFC Applications: Polyhedron-Engineered Pt-Based Nanocrystals. *Energy & Environmental Science* **2018**, *11* (2), 258–275.

(13) Chong, L.; Wen, J.; Kubal, J.; Sen, F. G.; Zou, J.; Greeley, J.; Chan, M.; Barkholtz, H.; Ding, W.; Liu, D.-J. Ultralow-Loading Platinum-Cobalt Fuel Cell Catalysts Derived from Imidazolate Frameworks. *Science* **2018**, *362* (6420), 1276–1281.

(14) Lufrano, F.; Gatto, I.; Staiti, P.; Antonucci, V.; Passalacqua, E. Sulfonated Polysulfone Ionomer Membranes for Fuel Cells. *Solid State Ionics* **2001**, *145* (1–4), 47–51.

(15) Ran, J.; Wu, L.; Wei, B.; Chen, Y.; Xu, T. Simultaneous Enhancements of Conductivity and Stability for Anion Exchange Membranes (AEMs) through Precise Structure Design. *Scientific reports* **2014**, *4*, 6486.

(16) Gottesfeld, S.; Dekel, D. R.; Page, M.; Bae, C.; Yan, Y.; Zelenay, P.; Kim, Y. S. Anion Exchange Membrane Fuel Cells: Current Status and Remaining Challenges. *Journal of Power Sources* **2018**, *375*, 170–184.

(17) Varcoe, J. R.; Slade, R. C. T. Prospects for Alkaline Anion-Exchange Membranes in Low Temperature Fuel Cells. *Fuel cells* **2005**, *5* (2), 187–200.

(18) Lufrano, F.; Gatto, I.; Staiti, P.; Antonucci, V.; Passalacqua, E. Sulfonated Polysulfone Ionomer Membranes for Fuel Cells. *Solid State Ionics* **2001**, *145* (1–4), 47–51.

(19) Marini, S.; Salvi, P.; Nelli, P.; Pesenti, R.; Villa, M.; Berrettoni, M.; Zangari, G.; Kiros, Y. Advanced Alkaline Water Electrolysis. *Electrochimica Acta* **2012**, *82*, 384–391.

(20) Lin, K.; Chen, Q.; Gerhardt, M. R.; Tong, L.; Kim, S. B.; Eisenach, L.; Valle, A. W.; Hardee, D.; Gordon, R. G.; Aziz, M. J.; others. Alkaline Quinone Flow Battery. *Science* **2015**, *349* (6255), 1529–1532.

(21) Liao, Q.; Zhang, J.; Li, J.; Ye, D.; Zhu, X.; Zheng, J.; Zhang, B. Electricity Generation and COD Removal of Microbial Fuel Cells (MFCs) Operated with Alkaline Substrates. *International Journal of Hydrogen Energy* **2014**, *39* (33), 19349–19354.

(22) An, L.; Zhao, T. S.; Shen, S. Y.; Wu, Q. X.; Chen, R. Alkaline Direct Oxidation Fuel Cell with Non-Platinum Catalysts Capable of Converting Glucose to Electricity at High Power Output. *Journal of Power Sources* **2011**, *196* (1), 186–190.

(23) Schibli, E. M.; Wright, A. G.; Holdcroft, S.; Frisken, B. J. Morphology of Anion-Conducting Ionenes Investigated by X-Ray Scattering and Simulation. *The Journal of Physical Chemistry B* **2018**, *122* (5), 1730–1737.

(24) Weber, A. Z.; Newman, J. Transport in Polymer-Electrolyte Membranes I. Physical Model. *Journal of the Electrochemical Society* **2003**, *150* (7), A1008--A1015.

(25) Pivovar, B. S. An Overview of Electro-Osmosis in Fuel Cell Polymer Electrolytes. *Polymer* **2006**, *47* (11), 4194–4202.

(26) Pushkareva, I. v; Pushkarev, A. S.; Grigoriev, S. A.; Modisha, P.; Bessarabov, D. G. Comparative Study of Anion Exchange Membranes for Low-Cost Water Electrolysis. *International Journal of Hydrogen Energy* **2019**.

(27) Li, Y. S.; Zhao, T. S.; Yang, W. W. Measurements of Water Uptake and Transport Properties in Anion-Exchange Membranes. *international journal of hydrogen energy* **2010**, *35* (11), 5656–5665.

(28) Yan, Q.; Toghiani, H.; Wu, J. Investigation of Water Transport through Membrane in a PEM Fuel Cell by Water Balance Experiments. *Journal of Power Sources* **2006**, *158* (1), 316–325.

(29) Zawodzinski, T. A.; Derouin, C.; Radzinski, S.; Sherman, R. J.; Smith, V. T.; Springer, T. E.; Gottesfeld, S. Water Uptake by and Transport through Nafion®117 Membranes. *Journal of the electrochemical society* **1993**, *140* (4), 1041–1047.

(30) Sone, Y.; Ekdunge, P.; Simonsson, D. Proton Conductivity of Nafion 117 as Measured by a Four-Electrode AC Impedance Method. *Journal of the Electrochemical Society* **1996**, *143* (4), 1254–1259.

(31) Merle, G.; Wessling, M.; Nijmeijer, K. Anion Exchange Membranes for Alkaline Fuel Cells: A Review. *Journal of Membrane Science* **2011**, *377* (1–2), 1–35.

(32) Hagesteijn, K. F. L.; Jiang, S.; Ladewig, B. P. A Review of the Synthesis and Characterization of Anion Exchange Membranes. *Journal of materials science* **2018**, *53* (16), 11131–11150.

(33) Zheng, Y.; Ash, U.; Pandey, R. P.; Ozioko, A. G.; Ponce-González, J.; Handl, M.; Weissbach, T.; Varcoe, J. R.; Holdcroft, S.; Liberatore, M. W.; others. Water Uptake Study of Anion Exchange Membranes. *Macromolecules* **2018**, *51* (9), 3264–3278.

(34) Li, Z.; He, X.; Jiang, Z.; Yin, Y.; Zhang, B.; He, G.; Tong, Z.; Wu, H.; Jiao, K. Enhancing Hydroxide Conductivity and Stability of Anion Exchange Membrane by Blending Quaternary Ammonium Functionalized Polymers. *Electrochimica Acta* **2017**, *240*, 486–494.

(35) Neagu, V.; Bunia, I.; Plesca, I. Ionic Polymers VI. Chemical Stability of Strong Base Anion Exchangers in Aggressive Media. *Polymer degradation and Stability* **2000**, *70* (3), 463–468.

(36) Noonan, K. J. T.; Hugar, K. M.; Kostalik IV, H. A.; Lobkovsky, E. B.; Abruña, H. D.; Coates, G. W. Phosphonium-Functionalized Polyethylene: A New Class of Base-Stable Alkaline Anion Exchange Membranes. *Journal of the American Chemical Society* **2012**, *134* (44), 18161–18164.

(37) Zhang, B.; Gu, S.; Wang, J.; Liu, Y.; Herring, A. M.; Yan, Y. Tertiary Sulfonium as a Cationic Functional Group for Hydroxide Exchange Membranes. *Rsc Advances* **2012**, *2* (33), 12683–12685.

(38) Huang, A.; Xia, C.; Xiao, C.; Zhuang, L. Composite Anion Exchange Membrane for Alkaline Direct Methanol Fuel Cell: Structural and Electrochemical Characterization. *Journal of applied polymer science* **2006**, *100* (3), 2248–2251.

(39) Wang, J.; Wang, J.; Zhang, S. Synthesis and Characterization of Cross-Linked Poly (Arylene Ether Ketone) Containing Pendant Quaternary Ammonium Groups for Anion-Exchange Membranes. *Journal of membrane science* **2012**, *415*, 205–212.

(40) Chu, X.; Shi, Y.; Liu, L.; Huang, Y.; Li, N. Piperidinium-Functionalized Anion Exchange Membranes and Their Application in Alkaline Fuel Cells and Water Electrolysis. *Journal of materials chemistry A* **2019**, *7* (13), 7717–7727.

(41) Wright, A. G.; Holdcroft, S. Hydroxide-Stable Ionenes. *ACS Macro Letters* **2014**, *3* (5), 444–447.

(42) Wright, A. G.; Weissbach, T.; Holdcroft, S. Poly (Phenylene) and m-Terphenyl as Powerful Protecting Groups for the Preparation of Stable Organic Hydroxides. *Angewandte Chemie International Edition* **2016**, *55* (15), 4818–4821.

(43) Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Computational intelligence and neuroscience* **2018**, *2018*.

(44) Gaikwad, S. K.; Gawali, B. W.; Yannawar, P. A Review on Speech Recognition Technique. *International Journal of Computer Applications* **2010**, *10* (3), 16–24.

(45) Bojarski, M.; del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L. D.; Monfort, M.; Muller, U.; Zhang, J.; others. End to End Learning for Self-Driving Cars. *arXiv preprint arXiv:1604.07316* **2016**.

(46) Fan, J.; Willdorf-Cohen, S.; Schibli, E. M.; Paula, Z.; Li, W.; Skalski, T. J. G.; Sergeenko, A. T.; Hohenadel, A.; Frisken, B. J.; Magliocca, E.; Mustain, W. E.; Diesendruck, C. E.; Dekel, D. R.; Holdcroft, S. Poly(Bis-Arylimidazoliums) Possessing High Hydroxide Ion Exchange Capacity and High Alkaline Stability. *Nature Communications* **2019**, *10* (1), 2306. https://doi.org/10.1038/s41467-019-10292-z.

(47) Wilson, R. L.; Sharda, R. Bankruptcy Prediction Using Neural Networks. *Decision support systems* **1994**, *11* (5), 545–557.

(48) Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* **2014**.

(49) Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural networks* **2015**, *61*, 85–117.

(50) Zikopoulos, P.; Eaton, C.; others. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*; McGraw-Hill Osborne Media, 2011.

(51) Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in Pytorch. **2017**.

(52) Chicco, D. Ten Quick Tips for Machine Learning in Computational Biology. *BioData mining* **2017**, *10* (1), 35.

(53) Wei, L.; Ding, Y.; Su, R.; Tang, J.; Zou, Q. Prediction of Human Protein Subcellular Localization Using Deep Learning. *Journal of Parallel and Distributed Computing* **2018**, *117*, 212–217.

(54) Hou, J.; Adhikari, B.; Cheng, J. DeepSF: Deep Convolutional Neural Network for Mapping Protein Sequences to Folds. *Bioinformatics* **2017**, *34* (8), 1295–1303.

(55) Aspuru-Guzik, A.; Lindh, R.; Reiher, M. The Matter Simulation (r) Evolution. *ACS central science* **2018**, *4* (2), 144–152.

(56) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse Molecular Design Using Machine Learning: Generative Models for Matter Engineering. *Science* **2018**, *361* (6400), 360–365.

(57) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS central science* **2018**, *4* (2), 268–276.

(58) Huan, T. D.; Batra, R.; Chapman, J.; Krishnan, S.; Chen, L.; Ramprasad, R. A Universal Strategy for the Creation of Machine Learning-Based Atomistic Force Fields. *NPJ Computational Materials* **2017**, *3* (1), 37.

(59) Li, Y.; Li, H.; Pickard IV, F. C.; Narayanan, B.; Sen, F. G.; Chan, M. K. Y.; Sankaranarayanan, S. K. R. S.; Brooks, B. R.; Roux, B. Machine Learning Force Field Parameters from Ab Initio Data. *Journal of chemical theory and computation* **2017**, *13* (9), 4492–4503.

(60) Brockherde, F.; Vogt, L.; Li, L.; Tuckerman, M. E.; Burke, K.; Müller, K.-R. Bypassing the Kohn-Sham Equations with Machine Learning. *Nature communications* **2017**, *8* (1), 872.

(61) Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R. Machine Learning of Accurate Energy-Conserving Molecular Force Fields. *Science advances* **2017**, *3* (5), e1603015.

(62) Kassal, I.; Jordan, S. P.; Love, P. J.; Mohseni, M.; Aspuru-Guzik, A. Polynomial-Time Quantum Algorithm for the Simulation of Chemical Dynamics. *Proceedings of the National Academy of Sciences* **2008**, *105* (48), 18681–18686.

(63) Deringer, V. L.; Bernstein, N.; Bartók, A. P.; Cliffe, M. J.; Kerber, R. N.; Marbella, L. E.; Grey, C. P.; Elliott, S. R.; Csányi, G. Realistic Atomistic Structure of Amorphous Silicon from Machine-Learning-Driven Molecular Dynamics. *The journal of physical chemistry letters* **2018**, *9* (11), 2879–2885.

(64) Gastegger, M.; Behler, J.; Marquetand, P. Machine Learning Molecular Dynamics for the Simulation of Infrared Spectra. *Chemical science* **2017**, *8* (10), 6924–6935.

(65) Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. SGDML: Constructing Accurate and Data Efficient Molecular Force Fields Using Machine Learning. *Computer Physics Communications* **2019**.

(66) Xie, T.; Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical review letters* **2018**, *120* (14), 145301.

(67) Sutskever, I.; Vinyals, O.; Le, Q. v. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*; 2014; pp 3104–3112.

(68) Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. v; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; others. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* **2016**.

(69) Schultz, M.; Reitmann, S. Prediction of Aircraft Boarding Time Using LSTM Network. In *2018 Winter Simulation Conference (WSC)*; 2018; pp 2330–2341.

(70) Zhang, T.; Song, S.; Li, S.; Ma, L.; Pan, S.; Han, L. Research on Gas Concentration Prediction Models Based on LSTM Multidimensional Time Series. *Energies* **2019**, *12* (1), 161.

(71) Zhao, Y.; Yang, R.; Chevalier, G.; Shah, R. C.; Romijnders, R. Applying Deep Bidirectional LSTM and Mixture Density Network for Basketball Trajectory Prediction. *Optik* **2018**, *158*, 266–272.

(72) Wolski, R.; Brevik, J.; Chard, R.; Chard, K. Probabilistic Guarantees of Execution Duration for Amazon Spot Instances. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; 2017; p 18.

(73) Bergstra, J.; Komer, B.; Eliasmith, C.; Yamins, D.; Cox, D. D. Hyperopt: A Python Library for Model Selection and Hyperparameter Optimization. *Computational Science & Discovery* **2015**, *8* (1), 14008.

(74) Baughman, M.; Haas, C.; Wolski, R.; Foster, I.; Chard, K. Predicting Amazon Spot Prices with LSTM Networks. In *Proceedings of the 9th Workshop on Scientific Cloud Computing*; 2018; p 1.

(75) Contreras, J.; Espinola, R.; Nogales, F. J.; Conejo, A. J. ARIMA Models to Predict Next-Day Electricity Prices. *IEEE Transactions on Power Systems* **2003**, *18* (3), 1014–1020. https://doi.org/10.1109/TPWRS.2002.804943.

(76) Sagheer, A.; Kotb, M. Time Series Forecasting of Petroleum Production Using Deep LSTM Recurrent Networks. *Neurocomputing* **2019**, *323*, 203–213.

(77) Gulcehre, C.; Cho, K.; Pascanu, R.; Bengio, Y. Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; 2014; pp 530–546.

(78) Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555* **2014**.

(79) Chakra, N. C.; Song, K.-Y.; Gupta, M. M.; Saraf, D. N. An Innovative Neural Forecast of Cumulative Oil Production from a Petroleum Reservoir Employing Higher-Order Neural Networks (HONNs). *Journal of Petroleum Science and Engineering* **2013**, *106*, 18–33.

(80) Hyndman, R. J.; Koehler, A. B. Another Look at Measures of Forecast Accuracy. *International journal of forecasting* **2006**, *22* (4), 679–688.

(81) Schibli, E. M. Analysis of Novel Ionenes via X-Ray and Neutron Scattering, 2016.

(82) Mokhtari, M.; Eslamibidgoli, M. J.; Eikerling, M. H. Electronic Structure and Conformational Properties of Polybenzimidazole-Based Ionenes—A Density Functional Theory Investigation. *ACS Omega* **2020**, *5* (3), 1472–1478. https://doi.org/10.1021/acsomega.9b03116.

(83) Parr, R. G. Density Functional Theory of Atoms and Molecules. In *Horizons of Quantum Chemistry*; Springer, 1980; pp 5–15.

(84) Jones, R. O.; Gunnarsson, O. The Density Functional Formalism, Its Applications and Prospects. *Reviews of Modern Physics* **1989**, *61* (3), 689.

(85) Kohn, W. Nobel Lecture: Electronic Structure of Matter—Wave Functions and Density Functionals. *Reviews of Modern Physics* **1999**, *71* (5), 1253.

(86) Jensen, F. *Introduction to Computational Chemistry*; John wiley & sons, 2017.

(87) Eslamibidgoli, M. J. Platinum Electrocatalysis: Novel Insights into the Dissolution Mechanism and Oxygen Reduction Reaction, 2016.

(88) Born, M.; Oppenheimer, R. Zur Quantentheorie Der Molekeln. *Annalen der physik* **1927**, *389* (20), 457–484.

(89) Onishi, T. Theoretical Chemistry for Advanced Nanomaterials: Functional Analysis by Computation and Experiment. Springer.

(90) Bader, R. F. W. Atoms in Molecules. *Accounts of Chemical Research* **1985**, *18* (1), 9–15.

(91) Hohenberg, P.; Kohn, W. Inhomogeneous Electron Gas. *Physical review* **1964**, *136* (3B), B864.

(92) Kohn, W.; Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical review* **1965**, *140* (4A), A1133.

(93) Lewars, E. Computational Chemistry. *Introduction to the theory and applications of molecular and quantum mechanics* **2003**, 318.

(94) Vosko, S. H.; Wilk, L.; Nusair, M. Accurate Spin-Dependent Electron Liquid Correlation Energies for Local Spin Density Calculations: A Critical Analysis. *Canadian Journal of physics* **1980**, *58* (8), 1200–1211.

(95) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; others. Gaussian 16 Revision B. 01. 2016; Gaussian Inc. *Wallingford CT* **46AD**.

(96) WEBMASTER. Gaussian16 Benchmark https://computational-chemistry.com/en/benchmark/gaussian16-benchmark/.

(97) Grimme, S. Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction. *Journal of computational chemistry* **2006**, *27* (15), 1787–1799.

(98) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Physical review letters* **1996**, *77* (18), 3865.

(99) Tao, J.; Perdew, J. P.; Staroverov, V. N.; Scuseria, G. E. Climbing the Density Functional Ladder: Nonempirical Meta--Generalized Gradient Approximation Designed for Molecules and Solids. *Physical Review Letters* **2003**, *91* (14), 146401.

(100) COHEN, A. J.; HANDY, N. C. Dynamic Correlation. *Molecular Physics* **2001**, *99* (7), 607–615.

(101) Becke, A. D. Density-Functional Thermochemistry. I. The Effect of the Exchange-Only Gradient Correction. *The Journal of chemical physics* **1992**, *96* (3), 2155–2160.

(102) Adamo, C.; Barone, V. Toward Reliable Density Functional Methods without Adjustable Parameters: The PBE0 Model. *The Journal of chemical physics* **1999**, *110* (13), 6158–6170.

(103) Chai, J.-D.; Head-Gordon, M. Long-Range Corrected Hybrid Density Functionals with Damped Atom--Atom Dispersion Corrections. *Physical Chemistry Chemical Physics* **2008**, *10* (44), 6615–6620.

(104) Miertuš, S.; Scrocco, E.; Tomasi, J. Electrostatic Interaction of a Solute with a Continuum. A Direct Utilizaion of AB Initio Molecular Potentials for the Prevision of Solvent Effects. *Chemical Physics* **1981**, *55* (1), 117–129.

(105) Pascual-ahuir, J.-L.; Silla, E.; Tunon, I. GEPOL: An Improved Description of Molecular Surfaces. III. A New Algorithm for the Computation of a Solvent-Excluding Surface. *Journal of Computational Chemistry* **1994**, *15* (10), 1127–1138.

(106) Mennucci, B.; Cammi, R. *Continuum Solvation Models in Chemical Physics: From Theory to Applications*; John Wiley & Sons, 2008.

(107) Roy Dennington, T. A. K.; Millam, J. M. GaussView, Version 6. Semichem Inc., Shawnee Mission, KS 2016.

(108) Tegenfeldt, J. O.; Prinz, C.; Cao, H.; Chou, S.; Reisner, W. W.; Riehn, R.; Wang, Y. M.; Cox, E. C.; Sturm, J. C.; Silberzan, P.; others. The Dynamics of Genomic-Length DNA Molecules in 100-Nm Channels. *Proceedings of the National Academy of Sciences* **2004**, *101* (30), 10979–10983.

(109) Hadziioannou, G.; van Hutten, P. *Semiconducting Polymers*; Wiley VCH Weinheim, New York, 2000.

(110) Boschetto, G.; Xue, H.-T.; Dziedzic, J.; Krompiec, M.; Skylaris, C.-K. Effect of Polymerization Statistics on the Electronic Properties of Copolymers for Organic Photovoltaics. *The Journal of Physical Chemistry C* **2017**, *121* (5), 2529–2538.

(111) McCormick, T. M.; Bridges, C. R.; Carrera, E. I.; DiCarmine, P. M.; Gibson, G. L.; Hollinger, J.; Kozycz, L. M.; Seferos, D. S. Conjugated Polymers: Evaluating DFT Methods for More Accurate Orbital Energy Modeling. *Macromolecules* **2013**, *46* (10), 3879–3886.

(112) White, M. A. *Properties of Materials*; 1999.

(113) Leonat, L.; Sbarcea, G.; Branzoi, I. V. Cyclic Voltammetry for Energy Levels Estimation of Organic Materials. *UPB Sci Bull Ser B* **2013**, *75*, 111–118.

(114) Lai, T. L.; Robbins, H.; Wei, C. Z. Strong Consistency of Least Squares Estimates in Multiple Regression II. *Journal of Multivariate Analysis* **1979**, *9* (3), 343–361.

(115) Devroye, L.; Györfi, L.; Krzyżak, A. The Hilbert Kernel Regression Estimate. *Journal of Multivariate Analysis* **1998**, *65* (2), 209–227.

(116) Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer series in statistics New York, 2001; Vol. 1.

(117) Burkov, A. *The Hundred-Page Machine Learning Book*; Andriy Burkov Quebec City, Can., 2019.

(118) McCulloch, W. S.; Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The bulletin of mathematical biophysics* **1943**, *5* (4), 115–133.

(119) Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review* **1958**, *65* (6), 386.

(120) Kailash Ahirwar. Everything you need to know about Neural Networks https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491.

(121) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning Representations by Back-Propagating Errors. *nature* **1986**, *323* (6088), 533–536.

(122) Hochreiter, S. The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **1998**, *6* (02), 107–116.

(123) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural computation* **1997**, *9* (8), 1735–1780.

(124) Mandic, D. P.; Chambers, J. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*; John Wiley & Sons, Inc., 2001.

(125) Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using {RNN} Encoder-Decoder for Statistical Machine Translation. *CoRR* **2014**, *abs/1406.1*.

(126) Hospital, A.; Goñi, J. R.; Orozco, M.; Gelp\'\i, J. L. Molecular Dynamics Simulations: Advances and Applications. *Advances and applications in bioinformatics and chemistry: AABC* **2015**, *8*, 37.

(127) Moon, J.; Kim, S.; Akgul, C. M.; Bae, S.-C.; Clark, S. M. Experimental and Theoretical Study on Elastic Properties of Crystalline Alkali Silicate Hydrate. *Materials & Design* **2020**, *185*, 108240.

(128) Jorgensen, W. L.; Tirado-Rives, J. The OPLS [Optimized Potentials for Liquid Simulations] Potential Functions for Proteins, Energy Minimizations for Crystals of Cyclic Peptides and Crambin. *Journal of the American Chemical Society* **1988**, *110* (6), 1657–1666.

(129) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of the American Chemical Society* **1996**, *118* (45), 11225–11236.

(130) Sharma, S. *Molecular Dynamics Simulation of Nanocomposites Using BIOVIA Materials Studio, Lammps and Gromacs*; Elsevier, 2019.

(131) Frenkel, D.; Smit, B. Understanding Molecular Simulation: From Algorithms to Applications. *Computational sciences series*. Elsevier (formerly published by Academic Press) 2002, pp 1–638.

(132) Hoover, W. G. Canonical Dynamics: Equilibrium Phase-Space Distributions. *Physical review A* **1985**, *31* (3), 1695.

(133) Kleinerman, D. S.; Czaplewski, C.; Liwo, A.; Scheraga, H. A. Implementations of Nosé--Hoover and Nosé--Poincaré Thermostats in Mesoscopic Dynamic Simulations with the United-Residue Model of a Polypeptide Chain. *The Journal of chemical physics* **2008**, *128* (24), 06B621.

(134) Plimpton, S. *Fast Parallel Algorithms for Short-Range Molecular Dynamics*; 1993.

(135) Chollet, F.; others. Keras. 2015.

(136) Bisong, E. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Springer, 2019; pp 59–64.

(137) LeCun, Y. A.; Bottou, L.; Orr, G. B.; Müller, K.-R. Efficient Backprop. In *Neural networks: Tricks of the trade*; Springer, 2012; pp 9–48.

(138) Ramsey, F. L.; others. Characterization of the Partial Autocorrelation Function. *The Annals of Statistics* **1974**, *2* (6), 1296–1301.

(139) Kurt, W. *Bayesian Statistics the Fun Way: Understanding Statistics and Probability with Star Wars, LEGO, and Rubber Ducks*; No Starch Press, 2019.

(140) Pal, A.; Prakash, P. K. S. *Practical Time Series Analysis: Master Time Series Data Processing, Visualization, and Modeling Using Python*; Packt Publishing Ltd, 2017.

(141) 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*; 2017; pp 5998–6008.

Appendix A.

Python code; Recurrent Neural Network (GRU and LSTM) models for multivariant time series forecasting

```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import seaborn as sns
import glob
from scipy import stats
from statsmodels.tsa import stattools
from statsmodels.tsa.stattools import acf, pacf
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer

from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, GRU, LSTM, Embedding
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard, ReduceLROnPlateau
```

```
df = pd.read_csv('input_trj.csv', header=0)
df.head()

# Raw data
# Coordinates of atom 100 will be selected as target!
ax = plt.gca()
df.plot(kind='line', y='x100', ax=ax)
df.plot(kind='line', y='y100', color='red', ax=ax)
df.plot(kind='line', y='z100', color='green', ax=ax)

plt.show()

# Data generation: Increase dataset size from 1e3 to 10e4

import numpy as np
from random import randrange, choice
from sklearn.neighbors import NearestNeighbors


def SMOTE(T, N, k):
  """
  Returns (N/100) * n_minority_samples synthetic minority samples.

  Parameters
  ----------
  T : array-like, shape = [n_minority_samples, n_features]
      Holds the minority samples
  N : percetange of new synthetic samples:
      n_synthetic_samples = N/100 * n_minority_samples. Can be < 100.
  k : int. Number of nearest neighbours.

  Returns
  -------
  S : array, shape = [(N/100) * n_minority_samples, n_features]
  """
  n_minority_samples, n_features = T.shape

  if (N % 100) != 0:
      raise ValueError("N must be < 100 or multiple of 100")
```

```
N = int(N/100)
n_synthetic_samples = int(N * n_minority_samples)
S = np.zeros(shape=(n_synthetic_samples, n_features))

#Learn nearest neighbours
neigh = NearestNeighbors(n_neighbors = k)
neigh.fit(T)

#Calculate synthetic samples
for i in range(n_minority_samples):
    nn = neigh.kneighbors(T[i].reshape(1, -1), return_distance=False)
    for n in range(N):
        nn_index = choice(nn[0])
        #NOTE: nn includes T[i], we don't want to select it
        while nn_index == i:
            nn_index = choice(nn[0])

        dif = T[nn_index] - T[i]
        gap = np.random.random()
        S[n + i * N, :] = T[i,:] + gap * dif[:]

return S
```

```
df_modified = df.iloc[:,1:]
col = list(df_modified.columns)
T = df_modified.to_numpy()

new_data = SMOTE(T, N = 1e4, k = 2)
new_data.shape

df_new = pd.DataFrame(data = new_data,
                      index = [i for i in range(new_data.shape[0])],
                      columns = col)
df_new.head()

# After increasing dataset
ax = plt.gca()
df_new.plot(kind='line', y='x100', ax=ax)
df_new.plot(kind='line', y='y100', color='red', ax=ax)
df_new.plot(kind='line', y='z100', color='green', ax=ax)

plt.show()


# Other transformers also can be used!
transformer = Normalizer().fit(df_new)
# Normalize all values between 0 and 0.5 as NN works better w/ them!
array_norm = transformer.transform(df_new)*10
df_norm = pd.DataFrame(data = array_norm,
                      index = [i for i in range(new_data.shape[0])],
                      columns = col)

# Add timeStep column back
df_norm.insert(0, "TimeStep", range(1,len(df_norm)+1))
#df_norm.to_csv("normal_input_trj.csv")
df_norm.head()

ax = plt.gca()
df_norm.plot(kind='line', y='x100', ax=ax)
df_norm.plot(kind='line', y='y100', color='red', ax=ax)
df_norm.plot(kind='line', y='z100', color='green', ax=ax)
```

```
plt.show()

sns.jointplot("TimeStep", "z100", df_norm, kind="kde", space=0, color="g")

target = ['x100', 'y100', 'z100'] # Atom 100
shift_steps = 400      # Lag
df_targets = df_norm[target].shift(-shift_steps)
n1 = df_norm.columns.get_loc("x100")

# Use 10 neighbour atoms (33 col in total) as the feature vectors
feature_col = df_norm.columns[n1-15:n1+18]
df_feature = df_norm[feature_col]

x_data = df_feature.values[0:-shift_steps]
y_data = df_targets.values[0:-shift_steps]


num_data = len(x_data)
train_split = 0.85       # fraction of the dataset for the training

num_train = int(train_split * num_data)
num_test = num_data - num_train

x_train = x_data[0:num_train]
x_test = x_data[num_train:]

y_train = y_data[0:num_train]
y_test = y_data[num_train:]

num_x_signals = x_data.shape[1]
num_y_signals = y_data.shape[1]

validation_data = (np.expand_dims(x_test, axis=0),
                   np.expand_dims(y_test, axis=0))

print("num_x_signals: ", num_x_signals)
```

```
# Batch generator
def batch_generator(batch_size):
    """
    Generator function for creating random batches of training data.
    """
    batch_num = int(num_train/batch_size) + 1
    # Infinite loop
    while True:
        # Allocate a new array for the batch of inputs.
        x_shape = (batch_num, batch_size, num_x_signals)
        x_batch = np.zeros(shape = x_shape, dtype = np.float64)
        print(x_shape)
        # Allocate a new array for the batch of outputs.
        y_shape = (batch_num, batch_size, num_y_signals)
        y_batch = np.zeros(shape = y_shape, dtype = np.float64)

        # Fill the batch with random sequences of data.
        for i in range(batch_num):
            # Get a random start-index.
            # This points somewhere into the training data.
            idx = np.random.randint(num_train - batch_size)
            idx = np.random.randint(num_train - batch_size)


             x_batch[i] = x_train[idx:idx+batch_size]
             y_batch[i] = y_train[idx:idx+batch_size]

         yield (x_batch, y_batch)

batch_size = 128  # 2^n like!
generator = batch_generator(batch_size = batch_size)
x_batch, y_batch = next(generator)
```

```
# Create the Recurrent Neural Network
model = Sequential()
model.add(GRU(units = 256,
              return_sequences = True,
              input_shape = (None, num_x_signals,)))

# To predict 3 output signals, we add a dense layer,
# which maps output values down to only 3 values.
model.add(Dense(num_y_signals, activation = 'sigmoid'))

if False:
    from tensorflow.keras.initializers import RandomUniform

    # Maybe use lower init-ranges.
    init = RandomUniform(minval=-0.02, maxval=0.02)
    model.add(Dense(num_y_signals,
                    activation = 'linear',
                    kernel_initializer = init))


# Loss Function: MSE
warmup_steps = 500

def loss_mse_warmup(y_true, y_pred):
    """
    Calculate the Mean Squared Error between y_true and y_pred,
    but ignore the beginning "warmup" part of the sequences.

    y_true is the desired output.
    y_pred is the model's output.
    """
```

```
    y_true_slice = y_true[:, warmup_steps:, :]
    y_pred_slice = y_pred[:, warmup_steps:, :]

    # Calculate the MSE loss for each value in these tensors.
    # Be aware of upgraded version of TensorFlow in colab!
    loss = tf.compat.v1.losses.mean_squared_error(y_true_slice, y_pred_slice)
    loss_mean = tf.reduce_mean(input_tensor = loss)

    return loss_mean


# Compile Model
optimizer = RMSprop(lr = 1e-3)
model.compile(loss = loss_mse_warmup, optimizer = optimizer, metrics = ['accuracy'])
model.summary()

# Callback Functions
path_checkpoint = '23_checkpoint.keras'
callback_checkpoint = ModelCheckpoint(filepath = path_checkpoint,
                                      monitor = 'val_loss',
                                      verbose = 1,
                                      save_weights_only = True,
                                      save_best_only = True)

callback_early_stopping = EarlyStopping(monitor = 'val_loss',
                                        patience = 8, verbose=1)



callback_tensorboard = TensorBoard(log_dir='./23_logs/',
                                   histogram_freq = 0,
                                   write_graph = False)
# Min learning rate 1e-5
callback_reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                                       factor = 0.10,
                                       min_lr = 1e-5,
                                       patience = 2,
                                       verbose = 1)

callbacks = [callback_early_stopping,
             callback_checkpoint,
             callback_tensorboard,
             callback_reduce_lr]
```

```
# Train the Recurrent Neural Network
model.fit_generator(generator = generator,
                    epochs = 30,             # least 20 - keep it
                    steps_per_epoch = int(num_train/batch_size),
                    validation_data = validation_data,
                    callbacks = callbacks)

# Load Checkpoint
try:
    model.load_weights(path_checkpoint)
except Exception as error:
    print("Error trying to load checkpoint.")
    print(error)

print("batch_size = ", batch_size)


# Generate Predictions
def plot_comparison(start_idx, length):
    """
    Plot the predicted and true output-signals.

    :par start_idx: Start-index for the time-series.
    :par length: Sequence length to process and plot.
    """
    # Entire dataset
    x = df_feature.values[:]
    y_true = df_targets.values[:]

    end_idx = start_idx + length
    # Pridict the future: the same amount as "shift" value!
    x = x[start_idx:end_idx]
    y_true = y_true[start_idx:end_idx]

    # Input signals for the model.
    x = np.expand_dims(x, axis=0)
    y_pred = model.predict(x)    # GRU model
```

```
    # print(y_pred[0].shape)
    # print(y_true.shape)

    for signal in range(len(target)):
        # Get the output-signal predicted by the model.
        signal_pred = y_pred[0][:, signal]

        # Get the true output-signal from the data-set.
        signal_true = y_true[:, signal]

        # Make the plotting-canvas bigger.
        plt.figure(figsize=(10,5))

        # Plot and compare the two signals.
        plt.plot(signal_true, label='true')
        plt.plot(signal_pred, label='pred')

        # Plot grey box for warmup period.
        p = plt.axvspan(0, warmup_steps, facecolor='black', alpha=0.3)


        # Plot labels
        plt.ylabel(target[signal])
        plt.legend()
        plt.show()

# Plot
plot_comparison(start_idx = 0, length = 10000)
```

Appendix B.

Effect of charge on the conformations of various HMT-PMBIs in gas phase (Figure 2.4)

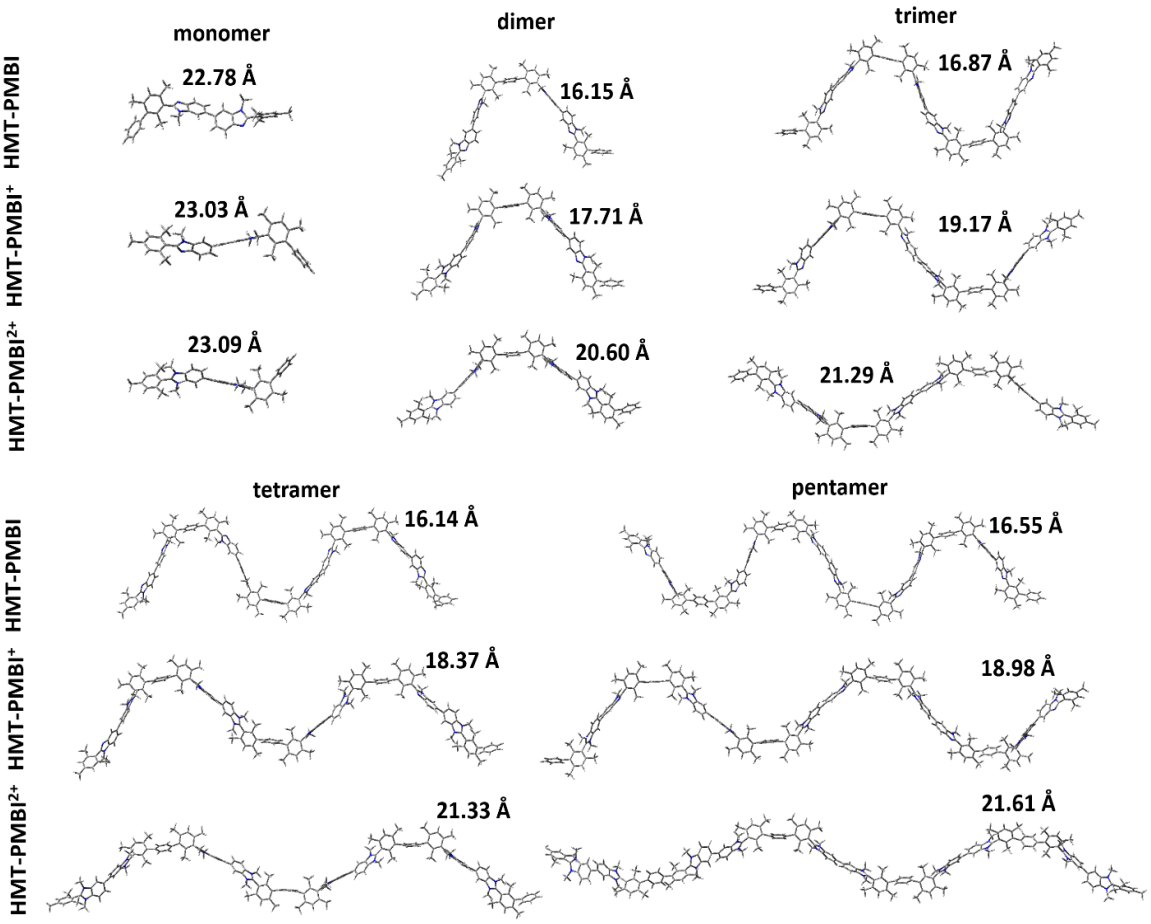


Figure B.1. **Effect of charge on the conformations of various studied systems in gas-phase. Numbers represent the normalized end-to-end distance values.**