Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Kunal Ghosh

Deep Learning for Predicting Molecular Electronic Properties

Master's Thesis
Espoo, August 7, 2017

Supervisor: Professor Aki Vehtari
Advisor: Professor Patrick Rinke

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Kunal Ghosh		
Title:	Deep Learning for Predicting Molecular Electronic Properties		
Date:	August 7, 2017	**Pages:**	iv + 42
Major:	Machine Learning and Data Mining	**Code:**	SCI3044
Supervisor:	Professor Aki Vehtari		
Advisor:	Professor Patrick Rinke		

Applications of novel materials have a significant positive impact on our lives. To search for such novel materials, material scientists traverse massive datasets of prospective materials identifying ones with favourable properties. Prospective materials are screened by studying a suitable spectra of these materials. Contemporary methods like high-throughput screening are very time consuming for moderately sized datasets.

Recently, deep learning algorithms have proven to be successful in modelling very complex functions like the mapping from image to text, use for image captioning and the mapping from text in one language to another, used for machine translation.

In this thesis, we propose deep learning methods which are able to predict molecular orbital energies and spectra, from only the charges and coordinates of constituent atoms of test molecules. Our proposed machine learning (ML) model surpassed the state-of-the-art in prediction accuracy of the molecular orbital energies and based on our literature review it is the first ML model to predict molecular spectra.

Keywords:	Deep Learning, Material Science, Machine Learning, Neural Networks
Language:	English

Preface

This work was carried out jointly, in the Probabilistic Machine Learning (PML) research group at the Department of Computer Science and the Computational Electronic Structure Theory (CEST) research group, at the Department of Applied Physics, at Aalto University.

I would like to thank Professor Aki Vehtari for his supervision and trusting me with this project. I would also like to thank my advisor, Professor Patrick Rinke, for being an excellent guide into the world of material science, for the regular meetings. I also thank Patrick for organizing the *Machine learning for Material Science* workshop during the start of my thesis work which provided a very good overview of the cutting edge research in machine learning for material science.

I would like to thank Dr. Milica Todorovic for her guidance and physics lessons, and Annika Stuke for creating the datasets used in this work. I express my gratitude to Patrick, Milica and Suhas Muniyappa for taking the time to review drafts of this thesis and providing constructive comments and feedback. I thank Peter Bjørn Jørgensen from DTU for his implementation of the DTNN model on which I built upon. I would also like to thank Jussi, Shishir and Srikanth, for their company, without which my learning experience as a Masters' student and my life in Finland might have been entirely different.

This thesis has significantly benefited from the computing resources and guidance made available by Aalto Science-IT, and I thank them for their support. A lot of free and open-source software packages have been used in this thesis and I thank all the volunteers for their contributions.

Finally, I would like to thank my parents for being my constant source of motivation and support.

Espoo, August 7, 2017

Kunal Ghosh

Contents

Chapter 1

Introduction

The applications of novel materials pervade our lives. Prominent examples being the lithium polymer battery in smart phones which have transformed the way people communicate and consume digital content, Teflon^{TM} coating on non-stick frying pans being used in kitchens worldwide. Humans have achieved space-worthiness, thanks to the high temperature withstanding coatings which prevent the space shuttle from burning out when re-entering earth's atmosphere. However, each one of these materials had to be engineered for the particular purpose by multiple teams of scientists often searching through many possible materials with desirable properties, altering them to synthesize new ones and re-iterating.

Apart from providing conveniences, the discovery of a novel material can also have an enormous societal impact. Global warming and climate change are major threats to life on earth. One of the main contributors global warming is the use of fossil fuel. Although the ill effects of the use of fossil fuels for meeting the energy demands of an ever growing human population are evident, they are one of the dominant sources of energy and the switch to renewable sources cannot be made yet.

Among the renewable energy sources, the wind and hydroelectric power require massive infrastructures to harness sufficient energy to meet the demands of a modern household. Solar power doesn't have this limitation. However, in 2009 only 1% of the worlds energy needs were met by solar power, and this is likely to increase to only 2-3% if there are no new material breakthroughs in solar cell materials research, according to a study done by Chu and Majumdar (2012). However, a recently discovered class of novel photovoltaic materials called the hybrid perovskites could be that breakthrough. Within only a few years of their discovery, the hybrid perovskites are approaching[1] the efficiency

[1] As per NREL (2017)

of conventional inorganic-based photovoltaic materials. This might be a step, towards making solar power a viable option for global energy needs.

Broadly, materials science comprises of materials synthesis and materials characterization. In this thesis, we will focus on the latter, and specifically spectroscopy. In spectroscopy, the material of interest is perturbed, and the response of the system is recorded. The response is almost always a *spectrum*. By analyzing these spectra, scientists can better understand the properties of the materials and consequently design better ones. Photoemission spectroscopy, in which a material is excited by light, and the spectra of emitted electrons are recorded, is one such example. The photoemission spectra have a close relationship with the *energies* corresponding to the electronic states of these substances.

Complementary to the experimental photoemission spectroscopy is theoretical spectroscopy, in which the electronic energy levels are calculated from quantum mechanical first principles. Search for novel materials by contemporary material scientists can be accelerated by high-throughput screening enabled by theoretical spectroscopy algorithms which take as input a *molecular representation* and run simulations[2] on them to calculate the energies corresponding to their molecular orbitals or the desired spectra. With access to a cluster of fast computers, many such simulations can be run in parallel. However, even for a moderately sized dataset of molecules, these simulations can take months[3].

Rupp (2015) noted that these quantum mechanical simulations have some redundancy, i.e. similar molecules go through similar computations during the simulations, resulting in correlated outputs. This is the motivation to use machine learning (ML) techniques in the field of material science, i.e. in a supervised learning setting, we could model the problem as using ML models to learn properties of (training) molecules and predict the properties of corresponding similar (test) molecules.

ML techniques have already been applied to quite a few problems in material science. Rupp et al. (2012) have compared the prediction accuracy of kernel methods and feed-forward neural networks when predicting the atomization energy of the molecules. Montavon et al. (2013) predict atomization energy and thirteen other molecular properties simultaneously using a multi-task neural network. More recently, Faber et al. (2017) have published a comparative study of using different regressors and input representations for predicting thirteen electronic properties of organic molecules. Schütt et al. (2017)

[2]These simulations are typically based on density functional theory (DFT).

[3]The computation of spectra and energies of the 132K molecules, used in our experiments, took two months (wall clock time) to compute on a cluster.

have proposed the deep tensor neural network (DTNN) to predict molecular free energies starting from a very generic molecular representation, which contains only the atom types and their spatial location in three dimensional space.

Although a lot of research has been done on predicting a few real valued target properties at a time, based on our literature survey, no one has yet investigated predicting a target function like a spectrum. Therefore in this thesis, *we study the use of ML techniques to predict the photoemission spectra, and energies corresponding to the orbitals of molecules* from datasets commonly employed by the material science community. Specifically, we try to adapt three neural network models, a feed-forward multilayer perceptron (MLP), a convolutional neural network (CNN) and the deep tensor neural network (DTNN). The models are trained on molecular representations to predict two real-valued target vectors, a sixteen-dimensional vector corresponding to energies of sixteen molecular orbitals and a 300-dimensional vector obtained by discretizing computed photoemission spectra.

Structure of the thesis

In Chapter 2 we introduce the different representations of molecules which are eventually used as input to the ML models. We also describe how the target vectors were generated and the molecular datasets used in our experiments. Thereafter, in Chapter 3 we describe the different ML models used in our experiments. This is followed by a discussion of hyperparameter choices of the ML models and the experimental results in Chapter 4. Finally, the discussion in Chapter 5 concludes the thesis.

Chapter 2

Datasets and data representation

In this chapter, we describe the data sets which we use for the experiments in Chapter 4. The different representations of the molecule, which form the input data, are described in Section 2.1. Thereafter, in Section 2.2, we discuss the two target vectors which are the output of the machine learning (ML) models discussed in Chapter 3.

2.1 Input representations

In this thesis, we propose ML models to predict molecular properties. Thinking abstractly, we want the models to accept a molecule as an input and return a desired molecular property. But a molecule is a physical entity. Before it can be input to a model, we must represent it in a suitable format. Many such representations have already been proposed in the literature. For example, Weininger (1988) have proposed the SMILES (Simplified Molecular Input Line Entry System) representation, which is a string based representation. The XYZ representation encodes the molecule as a list of its constituent atom types and their corresponding three-dimensional coordinates with respect to an arbitrary basis. The Coulomb matrix (CM) representation proposed by Rupp et al. (2012) described in Section 2.1.2 is derived from the XYZ data. The many-body tensor representation (MBTR) proposed by Hu et al. (2017) which is derived from the Coulomb matrix and includes information about angles between atoms in three-dimensional space. Figure 2.1 illustrates three different representations of an ethene molecule.

The models used in this thesis take as input the XYZ file and the Coulomb matrix. These representations have been discussed in Sections 2.1.1 and 2.1.2 respectively.

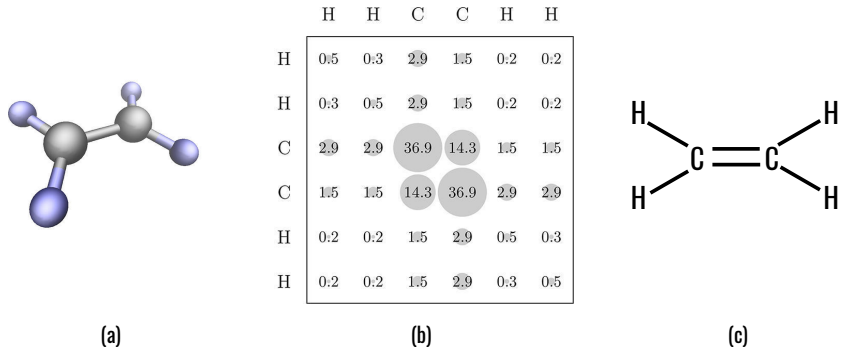


Figure 2.1: *Illustration of three different representations of ethane (C_2H_4) (a) The ball-and-stick model which shows the relative position of atoms, represented as spheres, in three-dimensions and the bonds between them. (b) The Coulomb matrix representing the pairwise interaction of atoms. It encodes the charges of atoms and their interatomic distances in its respective cells. (c) The structural formula presents a graphical representation of the molecule's structure, indicating different bonds between the atoms as one or more lines joining them.*

2.1.1 XYZ representation

The XYZ format is a plain text representation of molecules. Each molecule with d atoms corresponds to $d+2$ lines in the file. The first line contains a positive integer indicating the number of atoms in the molecule. The second line is a comment field and can contain any free form text. Following that, are d lines one for each atom in the molecule. Each line contains four fields; the first field contains the chemical symbol of the atom and fields two through four the Cartesian coordinates of the atom. The coordinates of the atoms are with respect to an arbitrary basis and are in Angstrom (Å) units, which corresponds to 10^{-10} meters. Figure 2.2 presents the XYZ representation of a methane molecule.

```
5
free=-1545.46
C 5.18160188 -0.90711087 -2.80992509
H 0.64223971  0.28323184  1.01144250
H 0.59174445 -1.01258058 -0.19836824
H 0.60046186  0.68235075 -0.71605884
H 0.90046186  0.48235075 -0.21605884
```

Figure 2.2: *Illustration of the XYZ representation of methane (CH_4). The first line indicates the number of atoms in the molecule. The second line is a comment field which can contain any free-form text. Following that, are entries containing chemical symbol of each atom in the molecule with their corresponding Cartesian coordinates.*

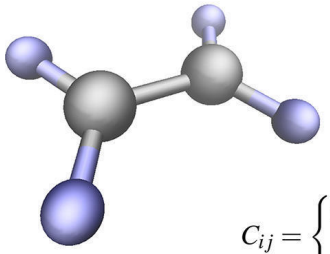
While using the XYZ representation, as an input to an ML algorithm, we should consider the following:

The order of atoms: The XYZ representation of a molecule is invariant to the ordering of its constituent atoms. Different permutations of the rows, corresponding to the coordinates of atoms, don't have any physical significance. Therefore the order of atoms is fixed, arbitrarily, when creating datasets of XYZ representations of molecules.

The Cartesian coordinate of atoms: The coordinate of the atoms in the XYZ representation is with respect to an arbitrary basis. The representation is also invariant to translation and rotation of the molecule. This poses a problem for ML models which, if presented with the XYZ representation without any preprocessing, would learn a function mapping from the coordinates of atoms to target properties. Since, physically, the properties are not dependent on the exact location of the atoms but their relative position from each other, naively using the XYZ representation for machine learning is a problem.

The ML models presented in Chapter 3 mitigate this problem by accepting representations which include interatomic distances rather than the coordinates of the atoms. DTNN accepts the interatomic distances where as MLP and CNN accept the Coulomb matrix (CM), which is computed from the inverse of these distances. Both of which are easily computed from the XYZ representation. In the following section, we describe the CM representation.

2.1.2 Coulomb matrix



$$C_{ij} = \begin{cases} 0.5\, Z_i^{2.4} & \forall i = j \\ \frac{Z_i Z_j}{\|\mathbf{R}_i - \mathbf{R}_j\|_2} & \forall i \neq j. \end{cases}$$

$\longrightarrow$ $\mathbf{C} =$

	H	H	C	C	H	H
H	0.5	0.3	2.9	1.5	0.2	0.2
H	0.3	0.5	2.9	1.5	0.2	0.2
C	2.9	2.9	36.9	14.3	1.5	1.5
C	1.5	1.5	14.3	36.9	2.9	2.9
H	0.2	0.2	1.5	2.9	0.5	0.3
H	0.2	0.2	1.5	2.9	0.3	0.5

Figure 2.3: *Coulomb matrix of the ethene molecule (C_2H_4). Figure edited from Hansen et al. (2013)*

The Coulomb matrix (CM) representation was proposed by Rupp et al. (2012). For a molecule with d atoms its corresponding CM, denoted as $\mathbf{C}$, is a $d \times d$ symmetric matrix. For two arbitrary atoms with atomic numbers Z_i, Z_j and coordinates R_i, R_j the entries of $\mathbf{C}$ given by,

$$C_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4}, & i = j \\ \frac{Z_i Z_j}{\|R_i - R_j\|_2}, & i \neq j. \end{cases} \tag{2.1}$$

The off-diagonal entries of the CM are, inversely proportional to the interatomic distances, this ensures that the representation is invariant to translation and rotation of the molecule in 3D space.

Random Coulomb matrices

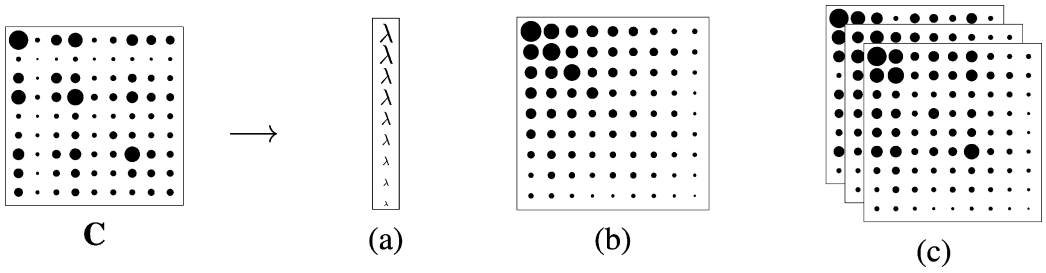


Figure 2.4: *Generating random instances of a Coulomb matrix* $\mathbf{C}$*. The size of the circles represent the magnitude of the corresponding entry in the CM. (a) Row norm of* $\mathbf{C}$*, represented by* λ*, is computed and then sorted in non-increasing order. (b) Rows and columns of* $\mathbf{C}$ *are ordered by the sorted row norm. (c) Gaussian noise is added to the sorted row norm. To obtain a random instance of* $\mathbf{C}$ *its rows and columns are permuted with the same permutation that sorts the sorted-row-norm with noise. Figure from Hansen et al. (2013)*

The Coulomb matrix (CM) of a molecule is permutation invariant to the order of its atoms. In this section we describe a CM randomization scheme proposed by Montavon et al. (2013). Training machine learning models using this randomization scheme ensures that the ML models learn function mappings which are invariant to the atom ordering in the CM. Figure 2.4 shows a visualization of the randomization scheme described in Algorithm 1.

Binarizing Coulomb matrices

Montavon et al. (2013) propose a scheme for binarizing the entries of a CM and state that the it allows real valued entries of the CM to be represented

Algorithm 1: Coulomb matrix randomization algorithm, as proposed by Montavon et al. (2013).

Data: Coulomb matrix ($\mathbf{C}$) of a molecule. Sorted by row norm.
Result: Randomized instance ($\mathbf{C}_r$) of the Coulomb matrix.

$r \leftarrow \|\mathbf{C}\|_2$; // row-wise or column-wise
$r \leftarrow r + \mathcal{N}(0,1)$
$i \leftarrow \text{argsort}(r)$
$\mathbf{C}_r \leftarrow \mathbf{C}[i][i]$; // reorder the rows and columns of C

over many dimensions of lower information content, this aided in optimizing the MLP used in their experiments. Yaeger et al. (1998) also show empirical evidence of easier optimization of neural networks when presented with smoothly varying and distributed inputs. This scheme is used to pre-process the CMs in experiments which predict sixteen HOMO energies using an MLP, in Chapter 4.

Equation 2.2 describes the CM binarization technique proposed by Montavon et al. (2013). In the equation $\phi(x)$ denotes the binarization of x, θ is the step size, and $\text{sigm}(x)$ represents the sigmoid[1] function. Montavon (2013) show that the binarization gets more fine-grained as θ approaches 0 and is more coarse as θ assumes larger positive, or negative values.

$$\phi(x) = \left[\ldots, \text{sigm}\left(\frac{x-\theta}{\theta}\right), \text{sigm}\left(\frac{x}{\theta}\right), \text{sigm}\left(\frac{x+\theta}{\theta}\right), \ldots\right] \tag{2.2}$$

Binarization of a $d \times d$ CM transforms it into a $d \times d \times \infty$ tensor. In the next two sections, we show the binarization of an arbitrary real number and discuss the process to transform an infinite dimensional, binarized CM, tensor to a finite one.

Binarizing a real number

When binarizing $x = 3$ using the previously proposed scheme, with $\theta = 1$, we get the following vector:

$$\phi(3) = \left[\ldots, \text{sigm}\left(\frac{3-2}{1}\right), \text{sigm}\left(\frac{3-1}{1}\right), \text{sigm}\left(\frac{3}{1}\right), \text{sigm}\left(\frac{3+1}{1}\right), \text{sigm}\left(\frac{3+2}{1}\right), \ldots\right] \tag{2.3}$$

which evaluates to the following infinite vector:

[1] $\text{sigm}(x) = \frac{\exp(x)}{1+\exp x}$

```
[..., 0.000, 0.000, 0.000, 0.001, 0.002, 0.007, 0.018, 0.047,
0.119, 0.269, 0.500, 0.731, 0.881, 0.953, 0.982, 0.993, 0.998,
0.999, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000,
...].
```

Dealing with the infinite dimensional binary vector

In the infinite vector above we observe that the sequence saturates, at zero or one, as the steps go farther away from $x = 3$. In practice, the stationary part of the vector can be pruned away resulting in a finite length vector. The number of steps after which to prune depends on the data set. For example, in a data set of Coulomb matrices (CMs), we can pick the largest entry among all CMs and then check the number of steps after which the binarized vector becomes stationary, the same number of steps (let us say s) is used for all the entries in the data set. Pruning in this way transforms the $d \times d \times \infty$ tensor to a finite $d \times d \times s$ tensor.

2.2 Target values

Machine learning models detailed in Chapter 3 take the molecular representations from Section 2.1 and predict target vectors. In this section, we describe the computation of the target vectors by building on intuition from physics.

Intution from an atom

In an atom, illustrated in Figure 2.5, the negatively charged electrons revolve around the positively charged nucleus in atomic orbitals. The energy corresponding to an orbital is the amount of energy that must be supplied to free an electron occupying that orbital. Hence, the orbital energies are negative. The energies are expressed in electron volts (eV) which is equal to 1.6×10^{-19} Joule, a unit of energy. An electron volt corresponds to the amount of energy gained, or lost, by an electron moving across an electric potential difference of one volt.

Molecular orbitals

When atoms bond together to form molecules, their atomic orbitals split in energy and cluster into bands of *molecular orbitals* (MO), which can be either occupied or unoccupied. The highest occupied molecular orbital is called

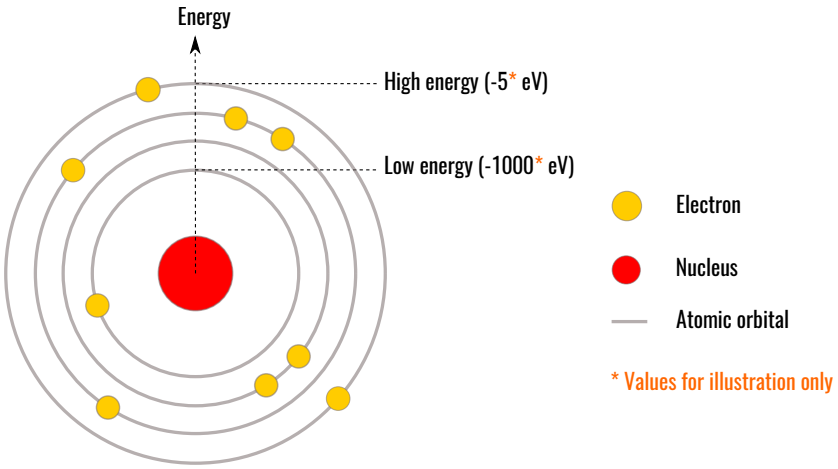


Figure 2.5: *Illustration of the Rutherford-Bohr model of an arbitrary atom. The positively charged nucleus, depicted as the red circle, is surrounded by electrons, represented as yellow circles. The electrons travel around the nucleus in circular orbits. The electron in the orbital closest to the nucleus has the lowest energy.*

HOMO, and the lowest unoccupied molecular orbital as LUMO. Figure 2.6a shows an illustration of molecular orbitals along with the HOMO and LUMO. Orbital with the second highest energy to HOMO is the HOMO-1, the one after that is HOMO-2, etc.

In the next two sections, we describe the output vectors of the machine learning models used in this thesis.

2.2.1 Sixteen HOMO energies

One of the output vectors is a vector of energies corresponding to HOMO, HOMO-1, HOMO-2, ..., HOMO-15 of the input molecules. The energies corresponding to the molecular orbitals are computed[2] from the XYZ representation of the molecules. Since the number of orbitals keeps varying between molecules, we select those with at least sixteen HOMO energies and pick their top sixteen HOMO energies for prediction. Figure 2.7 shows the distribution of the HOMO energies of a dataset consisting of 132 thousand molecules used in the experiments presented in Chapter 4.

[2]Using simulations based on *density functional theory* (DFT).

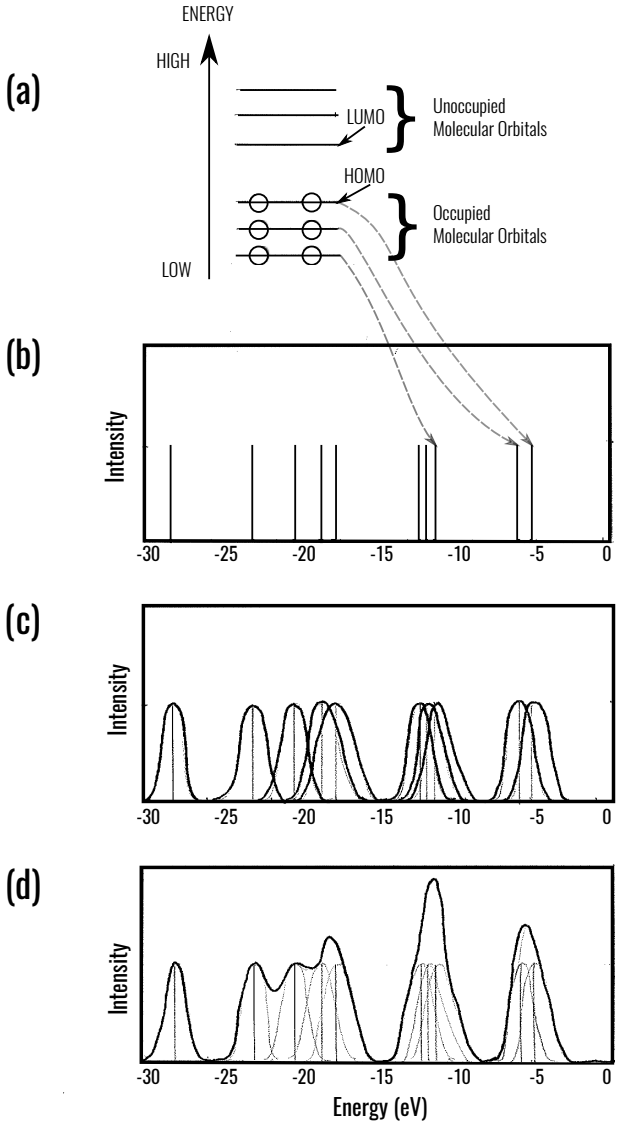
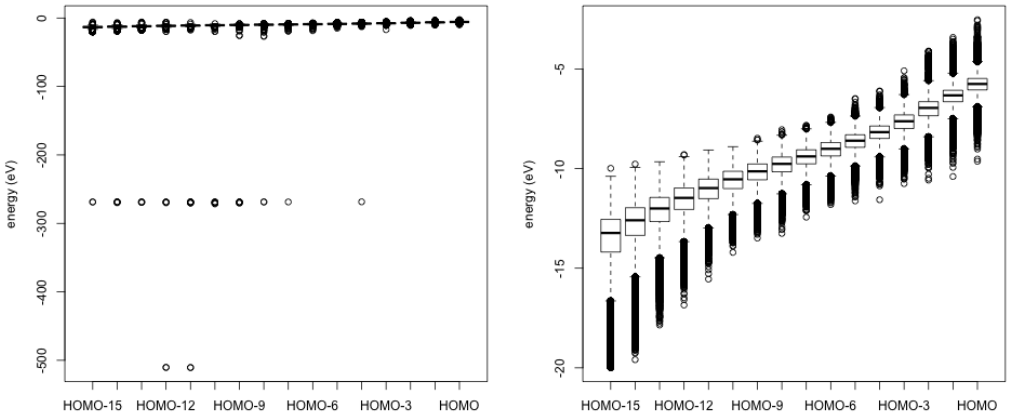


Figure 2.6: *Illustration of steps taken to compute the sixteen HOMO energies and the spectra used for the experiments in this thesis. (a) Canonical diagram of molecular orbitals. (b) Delta functions, computed from the XYZ representation of a molecule, correspond to the energies of HOMO, HOMO-1, HOMO-2, . . . , HOMO-15. (c) Imposing Gaussians to broaden the delta functions, accounts for the finite spread in the energies corresponding to molecular orbitals because of the vibration of molecules about their equilibrium position. (d) The magnitude of the Gaussians is summed to generate the spectrum.*

2.2.2 Discretized photoemission spectra

The machine learning models also predict photoemission spectra of the input molecules, which is computed from the sixteen HOMO energies by imposing Gaussians to broaden the delta functions correspond to the HOMO energies. The Gaussian broadening accounts for the finite spread in the energies corresponding to molecular orbitals because of the vibration of molecules about their equilibrium position. The 300-dimensional vector representing the spectrum is computed by evaluating the Gaussians at 300 equidistant points between minus thirty electron volts (eV) and zero eV and summing the magnitudes. Figure 2.6 illustrates the steps to compute the photoemission spectra from the HOMO energies. A representative example of the, thus computed, spectra is shown in Figure 2.8b.

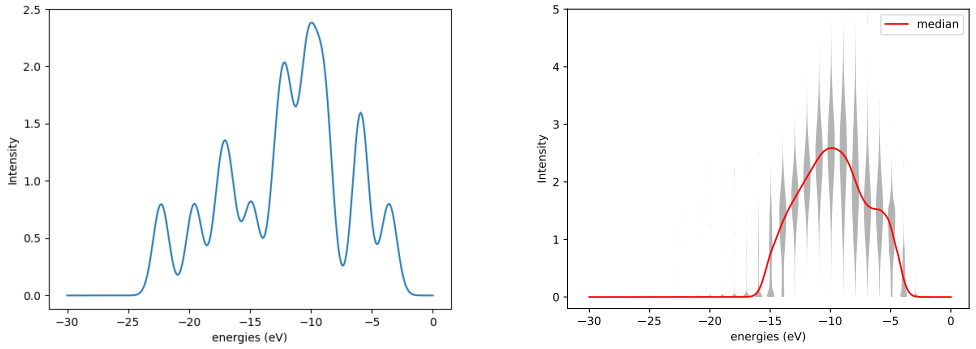


(a) ***(b)***

Figure 2.7: *Distribution of the sixteen HOMO energies of molecules from the 132K data set. (a) Box plot of the sixteen HOMO energies for all the molecules in the data set (b) Plot of the energies with outliers, molecule with one or more HOMO energies less than -250 eV, removed.*

2.3 Molecular generated databases

Standardized datasets, like the MNIST and CIFAR-10 image datasets by LeCun et al. (1998) and Krizhevsky and Hinton (2009) respectively, have ensured the results of the experiments conducted by researchers in the machine learning



(a) *(b)*

Figure 2.8: *The photoemission spectra. (a) The computed spectrum of a molecule from the 132K dataset. (b) The distribution of spectra of all molecules in the 132K dataset. The red line represents the median computed at each of the 300 points which constitute the spectra. Instead of the standard deviation, we draw violin plots. The plots, depicted in gray, describe the distribution of 30 equally spaced points among the 300 points between -30 and 0 eV. From the median, we infer that, in the 132K dataset, the peaks of the spectra occur mainly between -20 and 0 eV.*

community remain comparable. Likewise, standard datasets of molecules have been compiled by the material science community. In this section, we describe the two commonly used molecular datasets in material science research followed by their subsets used to conduct the experiments in Chapter 4.

GDB13 molecule dataset is a collection of 977 million molecules composed of carbon (C), hydrogen (H), oxygen (O), nitrogen (N), sulphur (S) and chlorine (Cl). Each molecule in the dataset contains up to thirteen non-hydrogen atoms.

GDB17 dataset contains 166 billion molecules made of C, H, O, N and F (fluorine). With each molecule having up to seventeen non-hydrogen atoms.

2.3.1 Molecular datasets for machine learning

In this section, we detail three subsets of GDB13 and GDB17; namely, the QM7, QM7b and the QM9 datasets which have been used quite frequently in the literature of machine learning for material science. The experiments presented in Chapter 4 take as input subsets of QM7 and QM9. The QM7b dataset is used to compare our results with those from Montavon et al. (2013) In Chapter 5.

QM7 by Blum and Reymond (2009); Rupp et al. (2012) is a subset of the GDB13 dataset, with 7 or fewer C, N, O and S atoms, resulting in a dataset of 7165 molecules. The dataset contains the coulomb matrix (CM) representation of molecules and a corresponding molecular property, the atomization energy. The atomization energy of a molecule is, intuitively, the amount of energy required to disintegrate the molecule into its constituent atoms such that they can no longer interact with each other.

QM7b, courtesy Blum and Reymond (2009); Montavon et al. (2013), extends the QM7 dataset by including molecules with chlorine atoms, resulting in a set of 7211 molecules. The dataset consists of the CM representation of molecules accompanied by their HOMO, LUMO, atomization energy and eleven other physical properties.

QM9 by Ramakrishnan et al. (2014); Ruddigkeit et al. (2012), is a subset of the GDB17 dataset containing 133885 molecules consisting of nine or fewer C, N, O and F atoms. It contains the XYZ representation of the molecules and corresponding 18 properties.

2.3.2 Molecular datasets used in this work

The two datasets used for the experiments presented in Chapter 4 are subsets of QM7 and QM9. The datasets were preprocessed by optimizing[3]the geometry of the molecules to obtain their lowest energy state. The resulting structure was then used to compute the new XYZ representation, Coulomb matrices (CM), sixteen HOMO energies and the spectra.

The first dataset was computed by optimizing the geometries of 7165 QM7 molecules. Out of the 7165 simulations, 247 did not converge and the corresponding molecules were discarded. Of the remaining, those with less than sixteen HOMO energies were dropped resulting in 5883 molecules which are hence forth called the *6K dataset*.

Similarly, the next dataset was computed by optimizing the geometries of 133885 QM9 molecules. The 132531 molecules in the *132K dataset* were obtained after leaving out the 71 molecules for which the geometry optimization did not converge and omitting the 1283 molecules with less than sixteen HOMO energies.

[3]The optimization was performed by our collaborator Annika Stuke

Chapter 3

Deep learning models

This chapter describes the neural network models that are evaluated in this thesis. We describe the feed-forward multi-layer perceptron (MLP) in Section 3.1 followed by the convolutional neural network (CNN) and deep tensor neural network (DTNN) in Sections 3.2 and 3.3, respectively.

3.1 Feed-Forward MLP

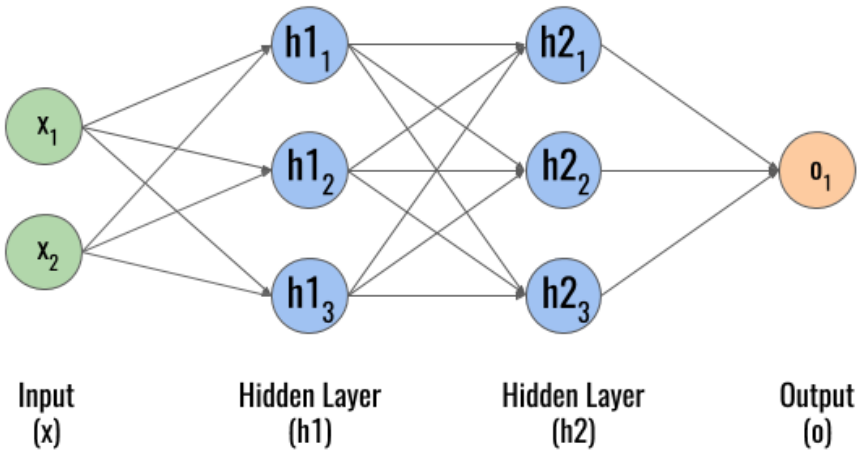


Figure 3.1: *An Illustration of a Feed-Forward MLP with a two-dimensional input* **x** *and a scalar output o. Each of the circles indicate a scalar quantity. This instance of the MLP has two hidden layers* **h1** *and* **h2** *each being a vector in* $\mathscr{R}^3$

The canonical representation of the Feed-Forward MLP is represented in

Figure 3.1. The MLP maps an input vector **x** to an output vector **o**. Typically the output is produced after one or more non-linear affine transformations of the Input. In Figure 3.1, the transformation of the input to the output is achieved as follows:

$$\begin{aligned} \mathbf{h1} &= \sigma\left(\mathbf{W_{x-h1}} \times \mathbf{x} + \mathbf{b1}\right) \\ \mathbf{h2} &= \sigma\left(\mathbf{W_{h1-h2}} \times \mathbf{h1} + \mathbf{b2}\right) \\ o &= \sigma\left(\mathbf{W_{h2-o}} \times \mathbf{h2} + \mathbf{b3}\right) \end{aligned} \tag{3.1}$$

In Equation 3.1 the σ represents some non-linear function like the sigmoid, hyperbolic tangent (tanh), or rectified linear unit (ReLU).[1]

The dimensions of the weight matrices and bias vectors in Equation 3.1 are as follows:

$$\begin{aligned} \mathbf{W_{x-h1}} &\in \mathscr{R}^{3\times 2} \\ \mathbf{W_{h1-h2}} &\in \mathscr{R}^{3\times 3} \\ \mathbf{W_{h2-o}} &\in \mathscr{R}^{3\times 1} \\ \mathbf{b1}, \mathbf{b2} &\in \mathscr{R}^{3\times 1} \\ \mathbf{b1} &\in \mathscr{R} \end{aligned} \tag{3.3}$$

The weight matrices and biases mentioned in Equation 3.3 are randomly initialized and are optimized using some gradient based method like stochastic gradient descent (SGD). For more background information on this model we refer the reader to Chapter 6 of Goodfellow et al. (2016).

3.2 Convolutional neural network

The Convolutional neural network (CNN) proposed by Lecun et al. (1998) is a model most appropriate for data which exhibits spacial locality, for example images.

The convolution operator is as follows:

$$\mathbf{O}(i,j,k) = \sum_l \sum_m \sum_n \mathbf{X}_{\text{part}}(i,j,k)\mathbf{W}(l-i, m-j, n-k) \tag{3.4}$$

[1] ReLU is a piece-wise approximation of the sigmoid using two linear functions. It is computed as follows:

$$\text{ReLU}(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0. \end{cases} \tag{3.2}$$

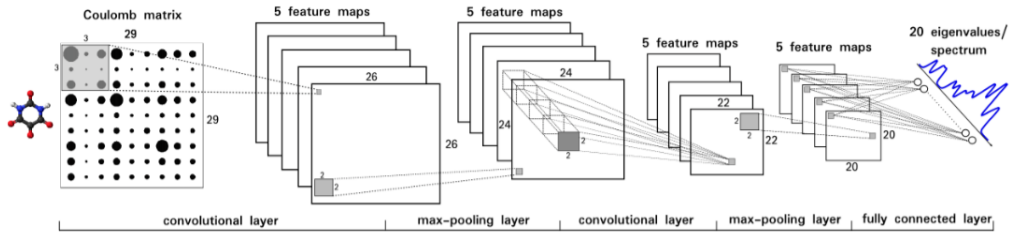


Figure 3.2: *An Illustration of the convolutional neural network (CNN) taking in a Coulomb matrix as input and predicting the molecular property (spectra or 20 eigen values). The CNN mainly consists of the convolution and pooling operations. The prediction is obtained by flattening the output of the last convolutional layer and projecting it to the desired output dimension using a fully connected layer. Figure courtesy of Annika Stuke.*

In Equation 3.4, $\mathbf{O}(i, j, k)$ denotes a scalar entry of the output tensor. $\mathbf{X}_{\text{part}}$ denotes the sub-tensor of $\mathbf{X}$ which has the same dimensions as the convolution filter $\mathbf{W}$. These tensors are illustrated in Figure 3.3.

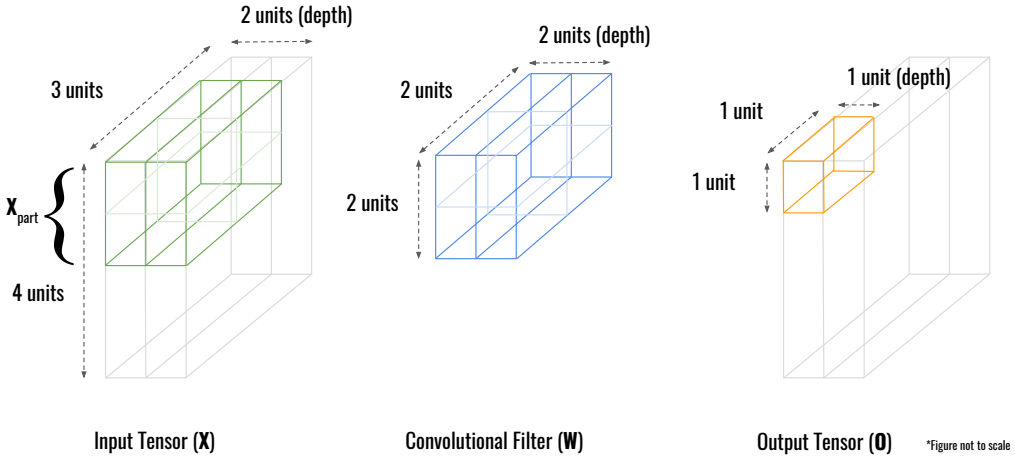


Figure 3.3: *An illustration of the dimensions of tensors when performing the convolution operation. Assuming an input tensor $\boldsymbol{X}$ ($\in \mathscr{R}^{4\times3\times2}$) and a convolution filter $\boldsymbol{W}$ (Tensor $\in \mathscr{R}^{2\times2\times2}$). While performing the convolution operation the filter $\boldsymbol{W}$ is convolved with a sub-tensor of $\boldsymbol{X}$, denoted as $\mathbf{X}_{part}$, with the same dimensions (as $\boldsymbol{W}$) resulting in a scalar output. Consequently, the convolution doesn't change the dimensions of the input. Note that the* depth *of the convolution filter is always same as the depth of the input on which it is applied.*

While implementing a CNN, it is common to group together multiple convolu-

tion filters into a *convolution layer* which may be followed by another convolution layer or a *pooling layer*. A pooling layer implements a *pooling function* which is a differentiable function which maps a tensor to a scalar.

Commonly used pooling functions are *max pooling* and *average pooling*. Assuming $\mathbf{X}_{\text{part}} = \{x_1, x_2, \ldots, x_n\}$ to be a part of the input matrix $\mathbf{X}$ and denoting the pooling operation applied on $\mathbf{X}_{\text{part}}$ as $f(\mathbf{X}_{\text{part}})$ the max pooling operation can be written as

$$f(\mathbf{X}_{\text{part}}) = \max(\{x_1, x_2, \ldots, x_n\}) \tag{3.5}$$

and average pooling operation as

$$f(\mathbf{X}_{\text{part}}) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.6}$$

In a CNN the convolution filters are updated using gradient based optimization techniques, just like the MLP in Section 3.1. A detailed discussion on CNNs can be found in Chapter 9 of Goodfellow et al. (2016).

3.3 Deep tensor neural network

The Deep tensor neural network (DTNN) model proposed by Schütt et al. (2017) takes inspiration from the quantum many-body Hamiltonian concept[2] and predicts the total energy of molecules as a sum of atomic energy contributions. Total energy of a molecule is the summation of its atomization energy and and the energies of each of its constituent atoms in isolation.

The DTNN, unlike the MLP and the CNN models (described in Sections 3.1 and 3.2 respectively) uses the XYZ input representation of the molecules. A molecule is input into the DTNN in two parts. One of the inputs is the vector $\mathbf{z}$, containing the atomic numbers of atoms in the molecule. The other part of the input is the $\mathbf{D}$ matrix consisting of inter-atomic, Euclidean, distances of atoms (where the atoms are arranged in the same order as in the $\mathbf{z}$ vector). Since a dataset might have molecules with different number of atoms. The $\mathbf{z}$ vector and the $\mathbf{D}$ matrix of molecules are zero-padded to ensure their dimensions match those of the largest molecule in the dataset.

[2]Intuitively, according to the quantum many-body Hamiltonian concept, molecular properties can be represented as a sum of atomic contributions.

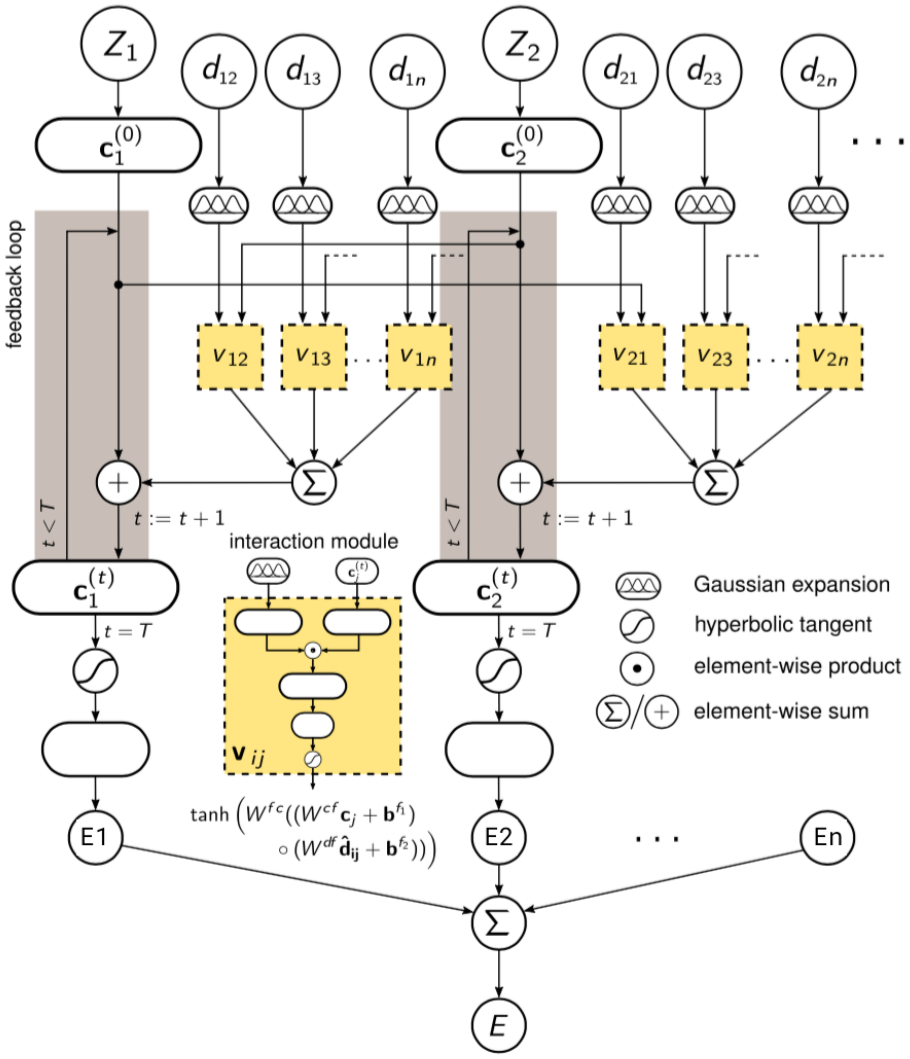


Figure 3.4: *An illustration of the deep tensor neural network (DTNN). Figure edited from Schütt et al. (2017).*

Important components of the DTNN

The atomic embeddings (c vector)

In DTNN each atom type i, for instance carbon or hydrogen, is represented by a vector $\mathbf{c}_i$ in $\mathscr{R}^B$. This is inspired[3] by the word-embeddings (word2vec) proposed by Mikolov et al. (2013) where each word in the corpus of documents

[3]As mentioned in the supplementary materials presented in Schütt et al. (2017).

is represented by a vector. In DTNN the embeddings $\mathbf{c}_i$ are randomly initialized by sampling from $\mathcal{N}(\mathbf{0}, \mathcal{I}/\sqrt{B})$. In the original DTNN model, Schütt et al. (2017) fixed B to 30, however for the experiments in our work we treat B as a hyper-parameter and optimize it over a range of possible values to get the lowest prediction error.

Gaussian expansion of the distances

The elements d_{ij} of the inter-atomic distance matrix $\mathbf{D}$ are expanded using equally spaced Gaussians. Consequently, the real vector d_{ij} is expanded to a vector $\hat{\mathbf{d}}_{ij}$ which we assume to be of length G i.e $\hat{\mathbf{d}}_{ij} \in \mathscr{R}^G$. The expansion is as follows:

$$\hat{\mathbf{d}}_{ij} = \left[\exp\left(-\frac{(d_{ij} - (\mu_{\min} + k\Delta\mu))^2}{2\sigma^2} \right) \right]_{0 \leq k \leq \frac{\mu_{\max}}{\Delta\mu}} \tag{3.7}$$

Schütt et al. (2017) used the following initialization scheme for the constants k, $\mu_{\max}$, $\mu_{\min}$, σ and $\Delta\mu$. We have used the same initialization scheme in the experiments presented in this work.

- $\mu_{\min}$ is initialized to -1.
- $\mu_{\max}$ is initialized depending on the range of d_{ij} values in a given dataset. In our experiments we have chosen $\mu_{\max} = 6.8$ (more about this in our discussion about $\Delta\mu$ and σ).
- $\mathbf{\Delta}\mu$ and σ were both initialized to 0.2 as was chosen by Schütt et al. (2017). This ensured that we had the range $[-1, 6.8]$ spanned by 40 (k in Equation 3.7) Gaussians. Both these values (and consequently $\mu_{\max}$) were limited by the memory (4GB) of the graphics processing unit (GPU) used for our experiments.

Gaussian expansion an example

In this section, we illustrate Gaussian feature expansion by applying it on a real number. Let us assume the following values for the variables in Equation 3.7; $d_{ij} = 2.1$, $\mu_{\min} = 1$, $\mu_{\max} = 3$, $\Delta\mu = \sigma = 1$. Using these values, we get $0 \leq k \leq \frac{\mu_{\max}}{\Delta\mu}$ which simplifies to, $0 \leq k \leq 3$ where k is an integer.

So, we have four Gaussians with unit standard deviation and means at 1, 2, 3 and 4 as illustrated in Figure 3.5 by broad black lines. Calculating the likelihood[4] of $d_{ij} = 2.1$ occurring in each of the Gaussians, we get 0.55, 0.99,

[4]The likelihood of the Gaussian needs to be scaled by $\sqrt{2\pi\sigma}$ so the we evaluate the expression as in Equation 3.7

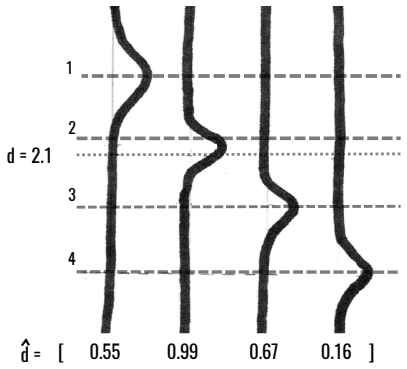


Figure 3.5: *An illustration of the Gaussian feature expansion of a real number.*

0.67 and 0.16 respectively. Thus, the scalar 2.1 has been expanded into the vector $[0.55, 0.99, 0.67, 0.16]^T$.

The interaction passes

As described in Section 3.3, the deep tensor neural network (DTNN) assumes that any molecular property can be expressed as a sum of atomic contributions (taking inspiration from the many-body Hamiltonian concept). The DTNN represents these atomic contributions as a non-linear projection of the *refined* vector $c_i^{(t=T)}$. The randomly initialized $c_i^{(t=0)}$ is *refined* over T iterations (the interaction passes) also depicted by the *feedback loop* in Figure 3.4. The *refinement* of the atomic embedding vector, in the iteration pass $t+1$ is given by Equation 3.8.

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \sum_{j \neq i} \mathbf{v}_{ij} \tag{3.8}$$

where $\mathbf{c}_i^t$ is the value of the embedding vector in the previous iteration and $\mathbf{v}_{ij}$ is defined in Equation 3.9.

$$\mathbf{v}_{ij} = \tanh\left(\mathbf{c}_j^{(t)} V \hat{\mathbf{d}}_{ij} + W^c \mathbf{c}_j^{(t)} + W^d \hat{\mathbf{d}}_{ij} + \mathbf{b}\right) \tag{3.9}$$

Understanding the interaction passes

Before moving forward, let us try to develop an intuition about what Equations 3.8 and 3.9 achieve. We notice from 3.8 that, given a molecule, the *refinement* of the $\mathbf{c}$ vector of a constituent atom i in an interaction pass is performed by adding contributions from $\mathbf{v}$ vectors of other[5] atoms.

[5]Notice the $j \neq i$ in the sum.

In Equation 3.9, we see that the $\mathbf{v}$ vector is an output from the hyperbolic tangent (tanh) function with a range of $(-1, 1)$. Consequently, the contributions from $\mathbf{v}_{ij}$ towards $\mathbf{c}_i$ (in Equation 3.8) can be additive or subtractive. Looking further at Equation 3.9, we notice that the contribution of an atom j towards the refinement of the embedding vector of an atom i is obt§ained from the combination of affine transformations of $\mathbf{c_j}$ (embedding of atom j), $\mathbf{d_{ij}}$ and an *interaction term*[6] between $\mathbf{c_j}$ and $\mathbf{d_{ij}}$. We observe the following in the first few interaction passes:

In Interaction pass $t = 0$, $\mathbf{c}_i^{(t=0)}$ are randomly initialized for all i.

In Interaction pass $t = 1$, all embedding vectors $\mathbf{c}_i^{(t=1)}$ are updated with contributions from other atoms $\mathbf{c}_j^{(t=0)}$ from the previous pass.

In Interaction pass $t = 2$, embedding vectors $\mathbf{c}_i^{(t=2)}$ are updated with contributions from other atoms $\mathbf{c}_j^{(t=1)}$. But $\mathbf{c}_j^{(t=1)}$ themselves contain contributions of other atoms. So, intuitively, $\mathbf{c}_i^{(t=2)}$ contains *pair-wise* contributions from all the other atoms which encodes angular information between atoms, as stated in Schütt et al. (2017). This allows DTNN to use the XYZ input representation, which scales linearly with the number of atoms in a molecule, yet express pair-wise and higher order interactions between atoms. In contrast, the MBTR input representation by Hu et al. (2017), encodes the pair-wise and higher order interactions explicitly, scales exponentially with the number of atoms.

Dimensions of the weights and biases

From Equation 3.8 we infer[7] that $\mathbf{v}_{ij} \in \mathscr{R}^B$. Therefore, the following are the dimensions of W^c, W^d, b and V.

$W^c \in \mathscr{R}^{B\times B}$ since it projects from $\mathbf{c} \in \mathscr{R}^B$ to $\mathbf{v} \in \mathscr{R}^B$.

$W^d \in \mathscr{R}^{B\times G}$ since $\mathbf{v} \in \mathscr{R}^B$ and $\hat{\mathbf{d}} \in \mathscr{R}^G$ (as per the assumption in Section 3.3).

$\mathbf{b} \in \mathscr{R}^B$ since it needs to have the same dimension as $\mathbf{v}$.

$V \in \mathscr{R}^{B\times B\times G}$ which is a *tensor* and is the reason why this model has been named deep tensor neural network.

[6] The interaction term is $\mathbf{c}_j^{(t)} V \hat{\mathbf{d}}_{ij}$ in Equation 3.9

[7] since $\mathbf{c}_i^t \in \mathscr{R}^B$

In the next few lines we check the dimensionality of V. Following Equation 3.9, we will check whether the dimensionality of $\mathbf{c}_j^t V \hat{\mathbf{d}}$ is the same as that of $\mathbf{v}$. We know that, $\mathbf{c}_j^t \in \mathscr{R}^B$ so $\mathbf{c}_j^t V$ i.e $\mathscr{R}^{1\times B} \times \mathscr{R}^{B\times B\times G} \in \mathscr{R}^{1\times B\times G}$. Furthermore, $\mathbf{c}_j^t V$ multiplied by $\hat{\mathbf{d}}_{ij}$ would be $\mathscr{R}^{1\times B\times G} \times \mathscr{R}^{G\times 1} \in \mathscr{R}^{1\times B\times 1}$ or $\mathscr{R}^B$, which is the dimensionality of $\mathbf{v}$.

Practical considerations while implementing the tensor

If we use V as a tensor, there would be $B \times B \times G$ parameters. To get a sense of the number of parameters, if we assume B = 30 and G = 40 (values used in some of our experiments) we have 36000 parameters from just this tensor. If we assume 2 interaction passes (as was used in our experiments) this results in 72000 parameters from the two V(s) alone. Given that our largest dataset of 132K molecules is in the same order of magnitude[8] as the number of parameters, this model is quite likely to overfit to the training data. To avoid the overfitting problem, Schütt et al. (2017) use a low-rank factorization of the tensor originally proposed by Taylor and Hinton (2009). Employing such a low-rank tensor factorization Equation 3.9 can be re-written as

$$\mathbf{v}_{ij} = \tanh\Big[\Big(W^{cf}(W^{fc}\mathbf{c}_j^{(t)} + \mathbf{b}_1^f) \circ (W^{fd}\hat{\mathbf{d}}_{ij} + \mathbf{b}_2^f)\Big)\Big] \tag{3.10}$$

where '$\circ$' denotes element-wise multiplication and the dimensionality of $\mathbf{c}_j^{(t)}$ and $\hat{\mathbf{d}}_{ij}$ is the same as that in Equation 3.9. b_1^f and b_2^f are column vectors in $\mathscr{R}^F$. W^{cf}, W^{fc} and W^{fd} are in $\mathscr{R}^{B\times F}, \mathscr{R}^{F\times B}$ and $\mathscr{R}^{F\times G}$ respectively. Here, F denotes the number of factors used in low-rank factorization of V, choosing an appropriate value for which, controls the number of parameters and also helps prevent overfitting.

DTNN training algorithm

To summarize, given a molecule (its corresponding atomic vector $\mathbf{z}$ and the inter-atomic distance matrix D) , the number of interaction passes T and the target[9] total energy e_{target}, Algorithm 2 can be used to learn the parameters which can then be used to predict total energy of similar[10] new molecules.

The DTNN algorithm is trained in a supervised manner. For each molecule, there is a corresponding target free energy e_{target}. The error, which is the difference between predicted free energy e and the target value e_{target}, is computed

[8] As a rule of thumb it is advised that the number of training examples must be an order of magnitude greater than the number of parameters in the model.

[9] Molecular total energy was the target value for the ML algorithm in Schütt et al. (2017)

[10] Similar to the molecules in the training set.

and a gradient based technique is used to update the parameters of the network.

3.3.1 Our modifications to the DTNN

In this section we describe the two main changes, to the original DTNN model, proposed in this thesis. The changes are as follows:

Addition of two fully-connected layers : Before the final fully-connected layer, predicting the total energy as in Schütt et al. (2017) or 16-energies and the spectrum in our work, we have added two[11] fully-connected layers. Therefore, in our implementation line seventeen of Algorithm 2 changes to

$$\begin{aligned} e_{i1} &\leftarrow \tanh(W\mathbf{c}_i + b) \\ e_{i2} &\leftarrow \tanh(W_1 e_{i1} + b_1) \\ e_i &\leftarrow \tanh(W_2 e_{i2} + b_2) \end{aligned} \tag{3.11}$$

This was done to facilitate the model to learn a more complex non-linear function, for mapping the atomic embedding ($\mathbf{c}_i^{(t=T)}$) to the output. The need to learn a more complex function mapping is reasonable because the target values being predicted in this work have a higher dimensionality (16 or 300 dimensional vector) as compared to that of Schütt et al. (2017).

Adding noise to the distance metric D : During training, we add Gaussian noise (zero mean, 0.1 standard deviation) to the distance matrix. So the distances d_{ij} on line 10 in algorithm 2 change to

$$d_{ij} \leftarrow d_{ij} + \mathcal{N}(0, 0.1) \tag{3.12}$$

Adding noise during the training of neural networks helps the model escape local minima and also leads to better generalization, as suggested by Du and Swamy (2014). In our experiments, we also observed that adding noise to the distance matrix led the model to achieve the same prediction accuracy in fewer epochs, as compared to the model which was trained without additive noise. A qualitative comparsion of the predictions with and without noise is presented in Figure 4.5.

In addition to the changes described above, we have also optimized the hyperparameters of the model for the datasets under investigation in this thesis, which is discussed in the following chapter.

[11] The number of fully-connected layers was arbitrarily chosen

Algorithm 2: The DTNN training algorithm. Schütt et al. (2017) used mini-batch gradient descent with Adam updates proposed by Kingma and Ba (2014). Here we use stochastic gradient descent instead of Adam.

Data: $\mathbf{z}$ the vector of atom types in a molecule.
Data: D the matrix of inter-atomic distances in the molecule, with entries d_{ij}.
Data: T the maximum number of interaction passes.
Data: e_{target} the target free energy of the molecule.
Result: e the predicted free energy of the molecule.

foreach *atom* i **do**
  if $\mathbf{z}_i$ *has corresponding* $\mathbf{c}_{z_i}$ **then**
    $\mathbf{c}_i^{(t=0)} \leftarrow \mathbf{c}_{z_i}$
  else
    $\mathbf{c}_{z_i} \sim \mathcal{N}(0, 1/\sqrt{B})$
    $\mathbf{c}_i^{(t=0)} \leftarrow \mathbf{c}_{z_i}$
  end
end
foreach *element* $d_{ij} \in D$ **do**
  $\hat{\mathbf{d}}_{ij} = \left[\exp\left(-\frac{(d_{ij}-(\mu_{\min}+k\Delta\mu))^2}{2\sigma^2}\right)\right]_{0\le k\le \frac{\mu_{\max}}{\Delta\mu}}$; `// Values as in 3.3`
end
`// T Interaction passes`
for $t \leftarrow 1$ **to** T **do**
  $\mathbf{v}_{ij} \leftarrow \tanh\left[\left(W^{cf}(W^{fc}\mathbf{c}_j^{(t)} + \mathbf{b}_1^f) \circ (W^{fd}\hat{\mathbf{d}}_{ij} + \mathbf{b}_2^f)\right)\right]$
  $\mathbf{c}_i^{(t+1)} \leftarrow \mathbf{c}_i^{(t)} + \sum_{j\neq i} \mathbf{v}_{ij}$
end
foreach *atom* i **do**
  $e_i \leftarrow \tanh(W\mathbf{c}_i + b)$
end
$e \leftarrow \sum_i e_i$
error $\leftarrow \left|e - e_{\text{target}}\right|$
foreach *parameter* $\in \left\{W^{cf}, W^{fc}, W^{fd}, W, b, \mathbf{b}_1^f, \mathbf{b}_2^f, \right\}$ **do**
  parameter $\leftarrow$ parameter $- \alpha \times \frac{d(parameter)}{d(error)}$
end

Chapter 4

Experiments and Results

The neural network models were implemented in Theano[1] to enable the use of graphics processing units (GPUs) to speed up computations. The models were trained on two almost identical machines with eight Intel(R) Xeon(R) E3-1230 v5 cores. One of the machines had an Nvidia GeForce GTX 1050Ti GPU and the other had an Nvidia Quadro K2200 both with 4GB of on-board memory. Additionally, we made use of GPyOpt[2] for hyperparameter optimization, which was run for ten iterations using the *expected improvement* (EI) acquisition function.

In the following sections, we first discuss the specifics of the hyperparameter optimization for each of the models in Section 4.1, and then assess the models' performance quantitatively and qualitatively in Sections 4.2 and 4.3, respectively.

4.1 Choice of hyperparameters

Feed-forward MLP

In this case, we use randomized and binarized Coulomb matrices, which are then flattened to yield a vector that is fed into the MLP. We use a two layer MLP[3] and optimized the *number of units* in each layer, to predict the single *HOMO energy* of the molecules in 6k and 132k dataset. During the optimization the number of units per layer were varied between 50 and 600 in steps of 50.

The hyperparameters of the MLP optimized to predict the HOMO energy, were also used to predict HOMO-1, HOMO-2, ..., HOMO-15. Separate hyper-

[1] Open-source package provided by the Theano Development Team (2016)

[2] Made available as an open-source package by The GPyOpt authors (2016)

[3] MLP implementation : `http://quantum-machine.org/code/nn-qm7.tar.gz` Accessed : 6.8.2017

parameter optimizations for the sixteen HOMO energies were not performed in the interest of time.

Convolutional neural network (CNN)

The CNN model accepts as input randomized Coulomb matrices. The size of the Coulomb matrices are dependent on the size of the largest molecule in the dataset. Consequently, the 6K dataset has 23×23 dimensional Coulomb matrices where as the 132K dataset has 29×29 dimensional Coulomb matrices. In a CNN it is common to have alternating convolutional and pooling layers as described in Section 3.2. The size of the pooling filters could also be a hyperparameter. However varying the size of the pooling layers also controls how many convolutional layers can follow[4]. To make the hyperparameter optimization easier, we utilize 2×2 max pooling layers, following the convolutional layers, for both datasets. We have three convolutional layers in the network and the number of filters in each layer is optimized.

To summarize, we optimized the *number of filters* in each convolutional layer. It was varied from 2 to 60 in steps of 5. The range was chosen to start from a very small 2 layer network to the largest we could accommodate in the GPU memory of our machine.

Deep tensor neural network (DTNN)

For the DTNN we optimized the *learning rate*, dimensionality of the *embedding vector* $\mathbf{c}$, the *mini-batch size* and the number of neurons in the final and penultimate hidden layers (see Section 3.3.1). Each of the hyperparameters was optimized over an equally spaced range of values. The upper limits for the range over which the hyperparameters were optimized, all except the learning rate, were dependent on the available GPU memory. We used the largest values of the hyperparameters for which the Theano model did not raise an out-of-memory exception during runtime. The corresponding lower limits were arbitrarily determined.

During the optimization, dimensionality of the atomic embedding vector $\mathbf{c}$ was varied between 20 and 41 in unit steps and the number of hidden units in the two fully connected layers were varied between 100 to 600 in steps of 50.

[4]Assuming a 23×23 Coulomb matrix and a 2×2 pooling layer which halves the height and width (the *depth* remains unchanged) of input every time its applied. Therefore we can have four sets of convolution and pooling layers i.e. $23 \times 23 \rightarrow 11 \times 11 \rightarrow 5 \times 5 \rightarrow 2 \times 2 \rightarrow 1 \times 1$ However if we have a 4×4 pooling layer we can only have two sets of convolution and pooling layers i.e. $23 \times 23 \rightarrow 5 \times 5 \rightarrow 1 \times 1$

In addition to the network specific hyperparameters, the mini-batch size used during training was optimized for all the network types. It was varied between 20 and 200, in steps of 5. The learning rate used for parameter updates was optimized in the CNN and DTNN models. The optimum learning rate was searched between 10^{-2} and 10^{-6} in multiples of 10.

4.2 Quantitative comparison of results

To achieve the lowest prediction error, hyperparameters of each model were optimized for each combination of dataset and prediction task. Thereafter the model training and evaluation cycle was repeated five times with the optimized hyperparameters. The corresponding results are summarized in Table 4.1. We observe that, DTNN achieves the lowest error while predicting the sixteen HOMO energies and the spectra (henceforth, referred to as *target properties*) regardless of the dataset size. CNN has marginally lower error while predicting the sixteen HOMO energies with the 6K dataset, when compared to the corresponding MLP.

Table 4.1: *Comparison of the error in predicting sixteen HOMO energies and spectra (target properties). Lower values are better. Each neural network model was trained, on the 6K and the 132K datasets, to predict the target properties. The root mean square error (**RMSE**) between the true data and ML model prediction is shown in the figure. The results were summarized from 5 runs, except for the spectra predictions of 132K dataset which were summarized from 3 runs. For the spectra the values presented are pointwise RMSE between the predicted and true functions, and don't have any units.*

Datasets →	6K		132K	
Model (Input) ↓	16 HOMO energies (eV)	Spectrum	16 HOMO energies (eV)	Spectrum (3 run summary)
MLP (Coulomb matrix)	0.317 ± 0.000	NA	NA	NA
CNN (Coulomb matrix)	0.304 ± 0.006	0.282 ± 0.002	0.231 ± 0.002	0.199 ± 0.000
DTNN (XYZ file)	$\mathbf{0.251 \pm 0.024}$	$\mathbf{0.210 \pm 0.000}$	$\mathbf{0.186 \pm 0.002}$	$\mathbf{0.145 \pm 0.000}$

In the next section, we perform a qualitative comparison of the results. Before we proceeed, we would like the reader to note that,

- In our experiments we found that the highest HOMO energy was the most difficult to predict. Therefore the hyperparameters of MLP were optimized to predict the HOMO energy of 132K molecules. Thereafter separate MLPs were trained to predict the sixteen HOMO energies with the

optimized hyperparameters. The training for eight of these models did not converge. Hence, the corresponding field in Table 4.1 is marked *not appliable* (NA). However, the MLP to predict the highest energy HOMO state trained successfully and the corresponding result is presented in Figure 4.2.

- Spectra of the 132K dataset were predicted by the CNN and DTNN using arbitrarily chosen hyperparameters. The corresponding results presented in Table 4.1 were summarized from three training runs, instead of five. Both, in the interest of time. DTNN was trained using a mini-batch size of 50, a learning rate of 10^{-5} and the last two fully connected layers with 100, 200 units, respectively. For the CNN, we used a learning rate of 10^{-4}, mini-batches of 90 training samples and convolutional layers with 22, 47 and 42 filters, respectively.

4.3 Qualitative comparison of results

Having compared the prediction of the models quantitatively, we now perform a visual comparison of the prediction in the following two sections. For the predictions of the sixteen HOMO energies, it will be interesting to observe that, some energies are easier to predict than the others. This was not apparent in the root mean sqaured error (RMSE) summary presented in the previous section, for all the sixteen HOMO energies combined.

In the next section we study the sixteen HOMO energies and the section following that, we consider the spectra.

4.3.1 Sixteen HOMO energies

Recall that, HOMO corresponds to the molecular orbital with the highest energy, followed by HOMO-1, HOMO-2, ..., and HOMO-15 corresponds to the lowest energy orbital among the sixteen we consider. In Figure 4.3 the predictions of MLP, CNN and DTNN are plotted when predicting the HOMO energies separately, for molecules from the 6K dataset. In Figure 4.4 we present the prediction accuracy of CNN and DTNN , for the same sixteen energies but for molecules from the 132K dataset. In both the figures, plots with higher correlation between predicted and true values indicates better performance. We observe a clear pattern, *predictions for lower energy orbitals is better than the higher energy orbitals*. This is also observed in Figure 4.1 where we see that most models make good predictions (high squared-correlation values) for

HOMO-15 and other lower energy states, but the performance is much more varied for the HOMO state.

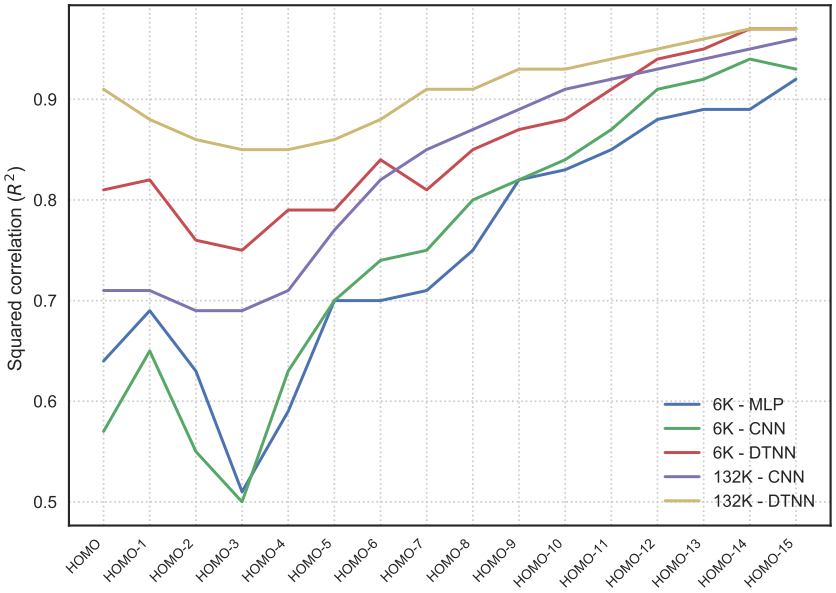


Figure 4.1: *Trends of Squared correlation (R^2) between predicted and true values of the sixteen HOMO energies with different combinations of datasets and machine learning models. We notice that most models achieve R^2 values close to 0.9 for lower energy HOMO energies (closer to HOMO-15). Higher energy states closer to HOMO are more difficult to predict, for these HOMO states we see that DTNN performs the best, with 6K and 132K datasets followed by the CNN with the larger 132K dataset. We notice that for the smaller 6K dataset, MLP performs better than the CNN.*

It is also interesting to note that (refer Table 4.1), although DTNN achieves almost the same RMSE values with 6K and 132K dataset, the prediction for the HOM energies of the 6K dataset observed separately in Figure 4.1, is worse compared to that of 132K with the same model. This highlights the need to look at predictions of individual HOMO states separately, instead of an averaged RMSE values over all the sixteen states.

The prediction accuracy of two DTNN models trained on the same dataset and with the same hyperparameters, with and without added noise appears to be comparable, as seen in Figure 4.5. However, training the model with added noise was much faster, around 36 hours compared to 42 hours for the one without noise. The results presented in Figure 4.5 are with the same optimized hyperparameters as used in models whose results are presented in Figure 4.4.

Finally, comparison of HOMO prediction accuracy of MLP, CNN and DTNN

across the 6K and 132K datasets is illustrated in Figure 4.2.

4.3.2 Discretized photoemission spectra

Spectra predictions have been considered only with the CNN and DTNN models in this thesis. The models were trained to minimize the mean squared error between the predicted and true specta. Illustrations comparing the predictions of test spectra by the two models, accompanied by histograms of prediction errors, are presented in Figures 4.6 and 4.7 for the 6K and 132K datasets, respectively. The figures display representative examples of predictions with lowest error, average error and the highest error. The spectra predictions are deemed good when they closely overlap the corresponding true spectra. We observe that, for the 132K dataset the best prediction with both models is very good. For the example corresponding to the average error, DTNN and CNN predict the true spectrum reasonably well, predicting peaks mostly where the true peaks appear, but smooth out some of the features. The worst predictions also predict the peaks of the true spectra with reasonable accuracy but underestimate their magnitude. Predictions of the spectra for the 6K molecules follow a similar trend.

This concludes the discussion of the experiments and results. In the next chapter, we draw conclusions from the observations presented in this chapter and place the results in a broader perspective.

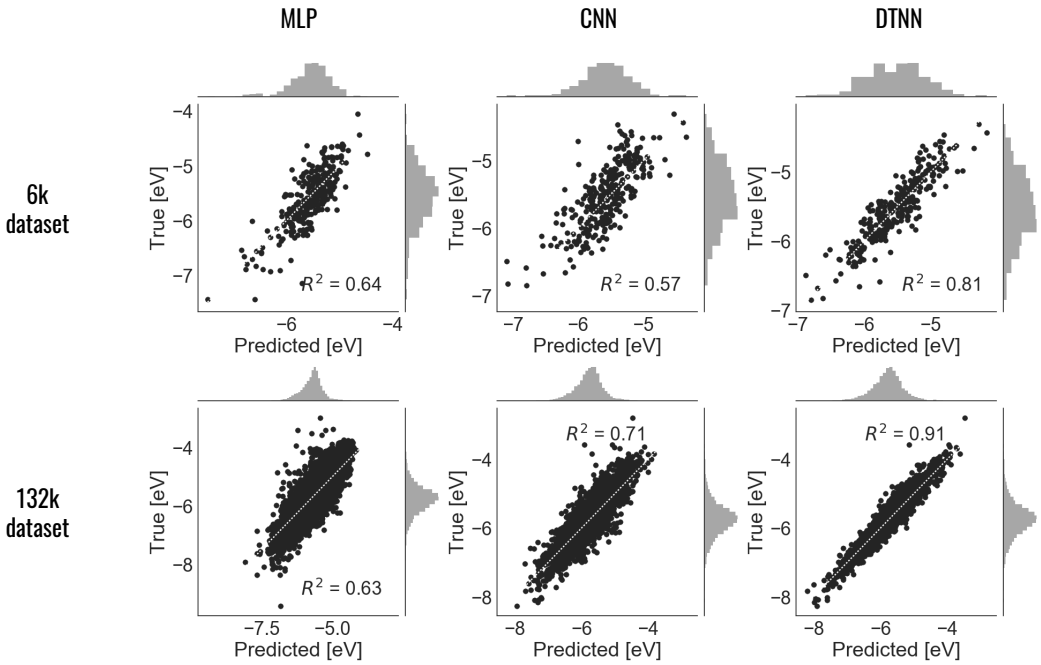


Figure 4.2: *Scatter plots showing correlation between predicted and true HOMO values. Higher correlation is better. The plots presented here are same as the corresponding ones in Figure 4.4 and 4.3, except for the MLP predictions on the 132K dataset, which is presented only here. The hyperparameters of each model has been optimized for both datasets. We observe that DTNN gives the best HOMO predictions regardless of the size of the training set. When comparing MLP and CNN, we find that CNN performs better for the larget 132K dataset. However, for the smaller 6K dataset MLP performs better than CNN.*

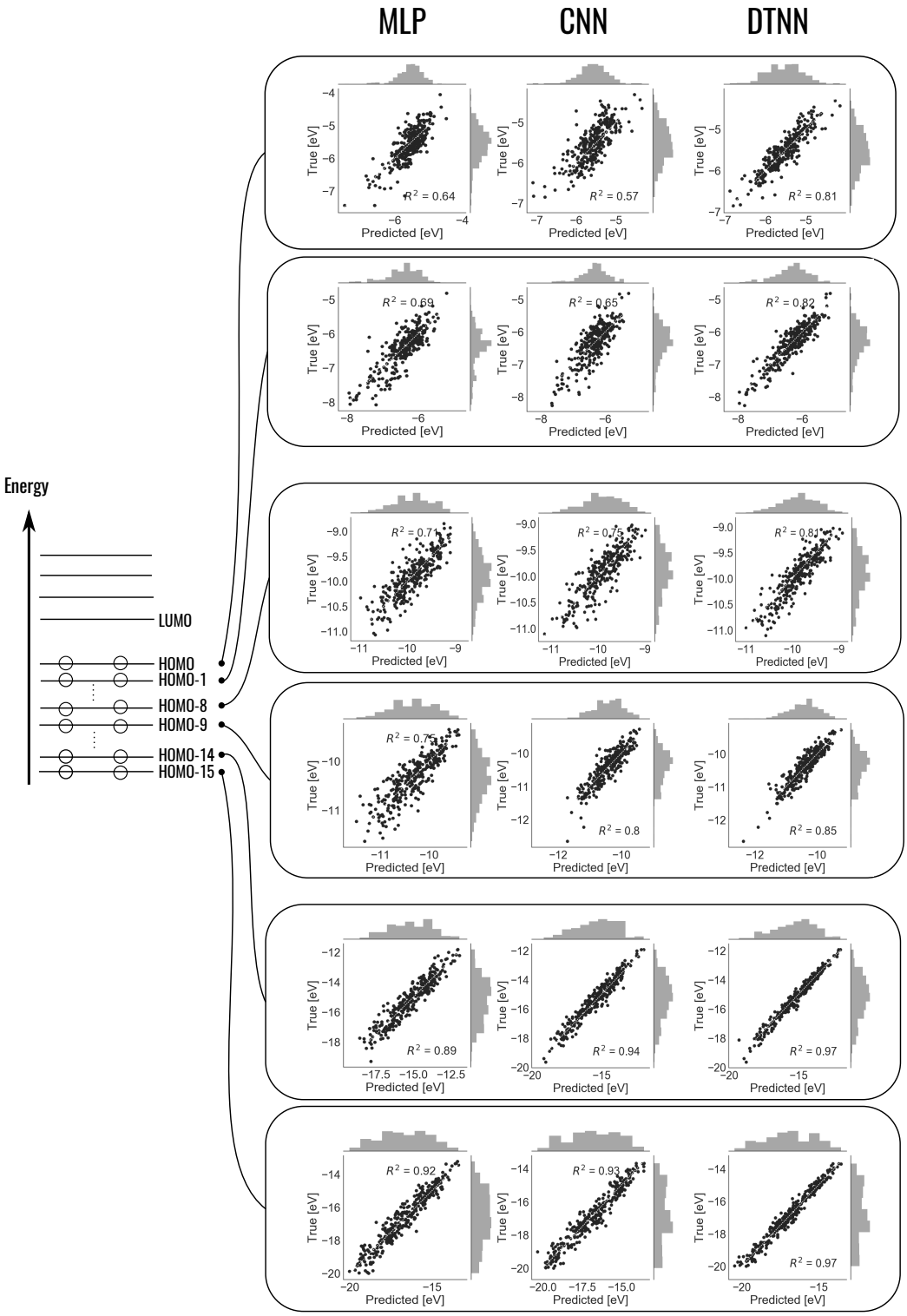


Figure 4.3: *Scatter plots showing correlation between predicted and true values of six, out of the sixteen, HOMO energies. Higher correlation is better. In columns, from left to right, we see predictions by the MLP, CNN and DTNN, respectively. Dots represent datapoints from the test set containing approximately 290 molecules from the 6K dataset. The predicted energy is plotted on the x-axis and the true energy on the y-axis.*

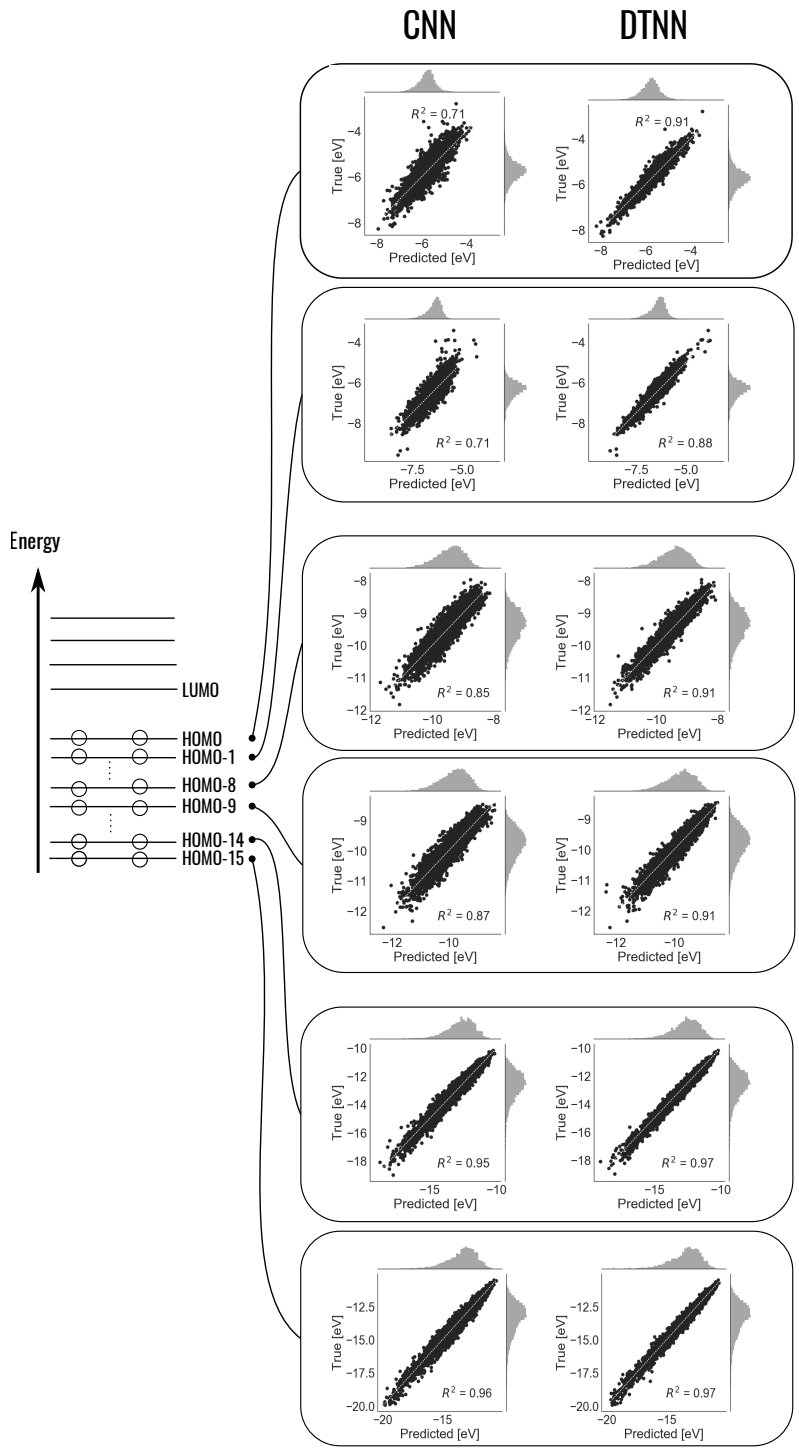


Figure 4.4: *Scatter plots showing correlation between predicted and true values of six, out of the sixteen, HOMO energies. Higher correlation is better. The left column of figures are predictions by CNN and the ones on the right are by DTNN. Dots represent the datapoints from the test set of approximately 13000 molecules from the 132K dataset. The predicted energy is plotted on the x-axis and the true energy on the y-axis.*

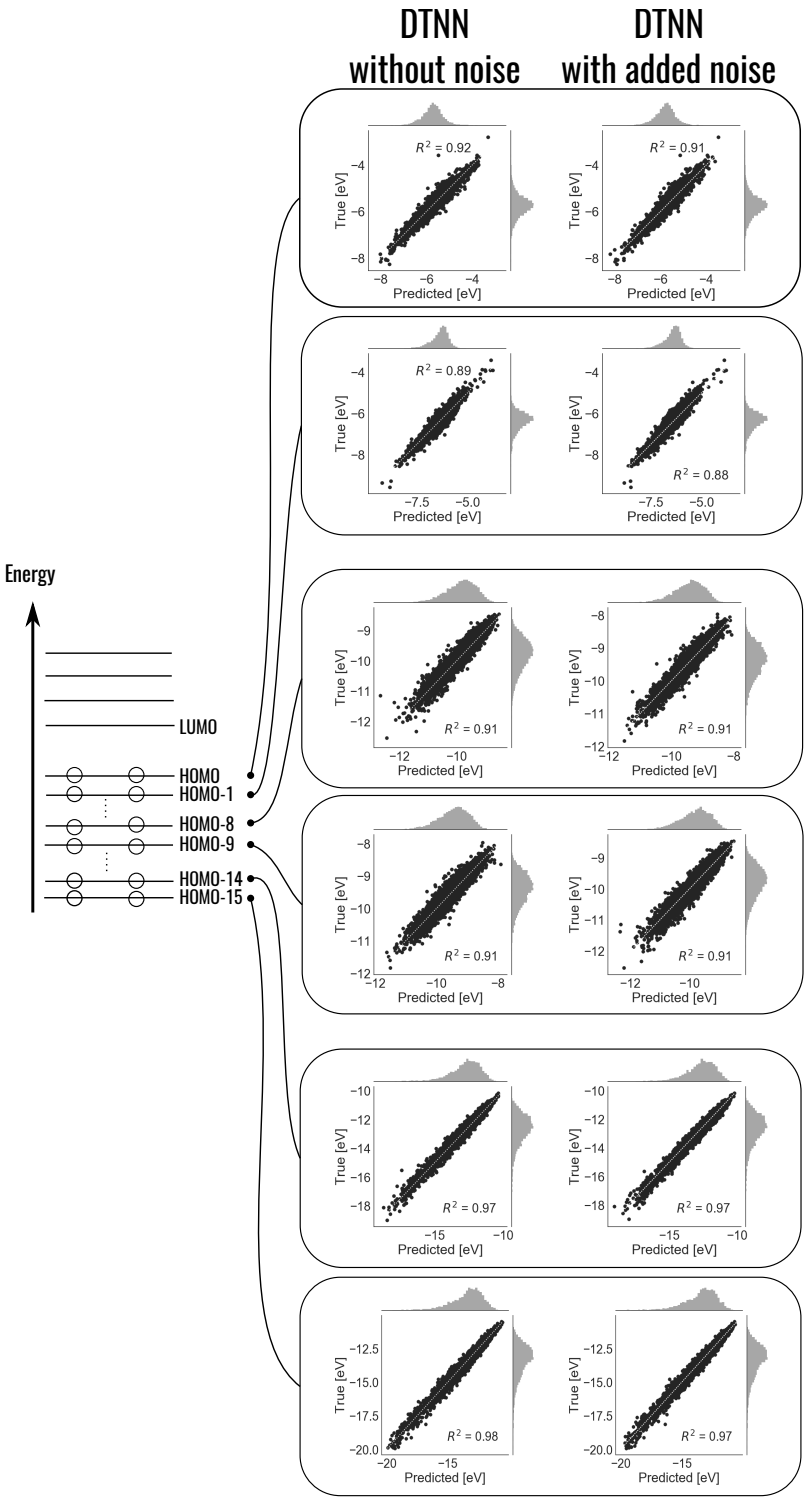


Figure 4.5: *Scatter plots showing correlation between predicted and true values of six, out of the sixteen, HOMO energies. Higher correlation is better. Predictions without added noise are in the left column and, the ones with added noise are to the right, hyperparameters of the two models kept the same. We observe that adding noise to the input does not significant improve the results. However, the model with noise took 36 hours to train whereas the one without noise took 42 hours. So, it is much faster to train the model with noise.*

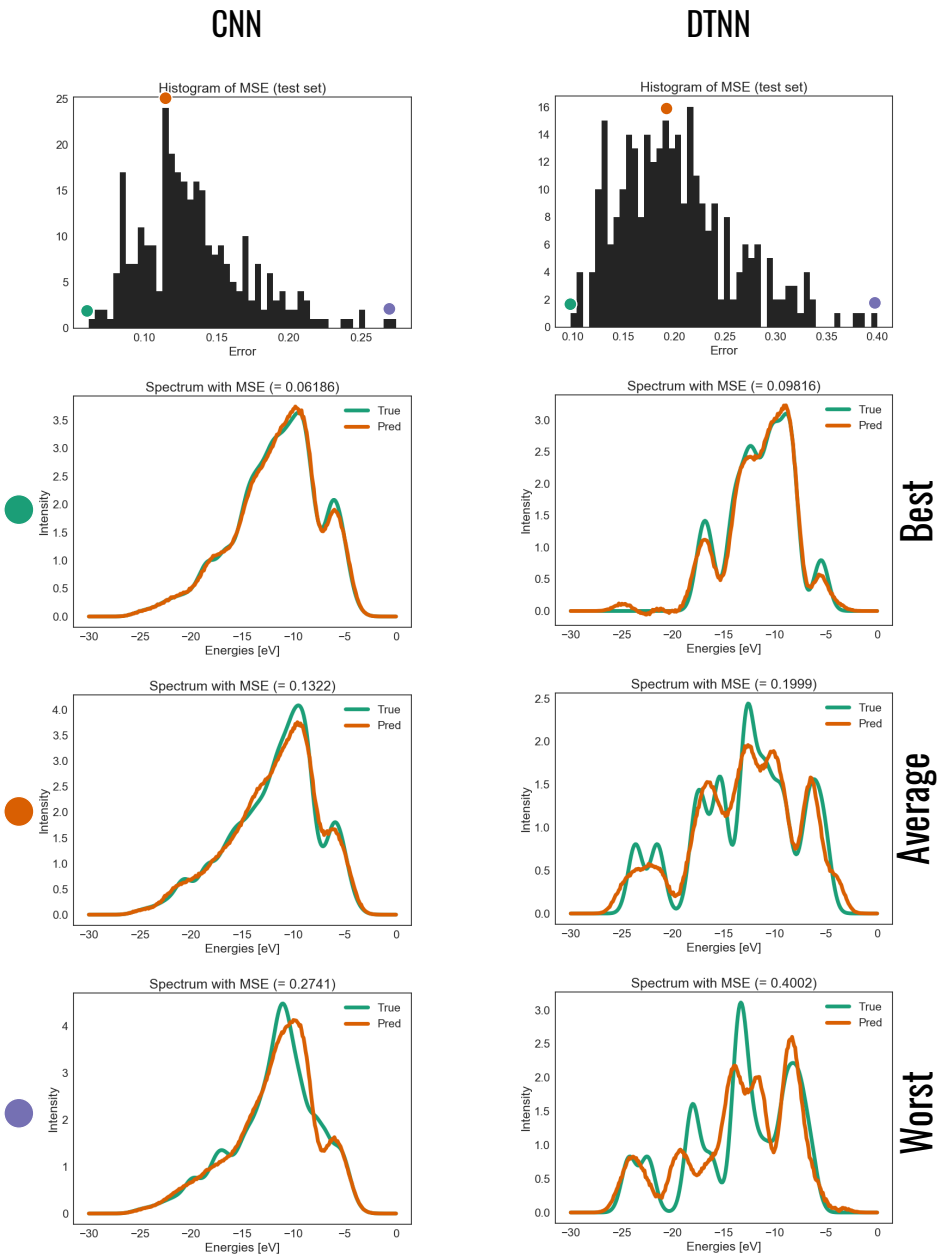


Figure 4.6: *Figure comparing the spectra predictions by CNN and DTNN. The histograms, on the first row, present the distribution of mean squared error (MSE) between predicted and true spectra of approximately 290 test molecules from the 6K dataset. The plots following the histograms depict the predicted and true spectra of a test molecule and their MSE. The bins, of the histogram, from which the molecules were sampled are depicted by coloured circles.*

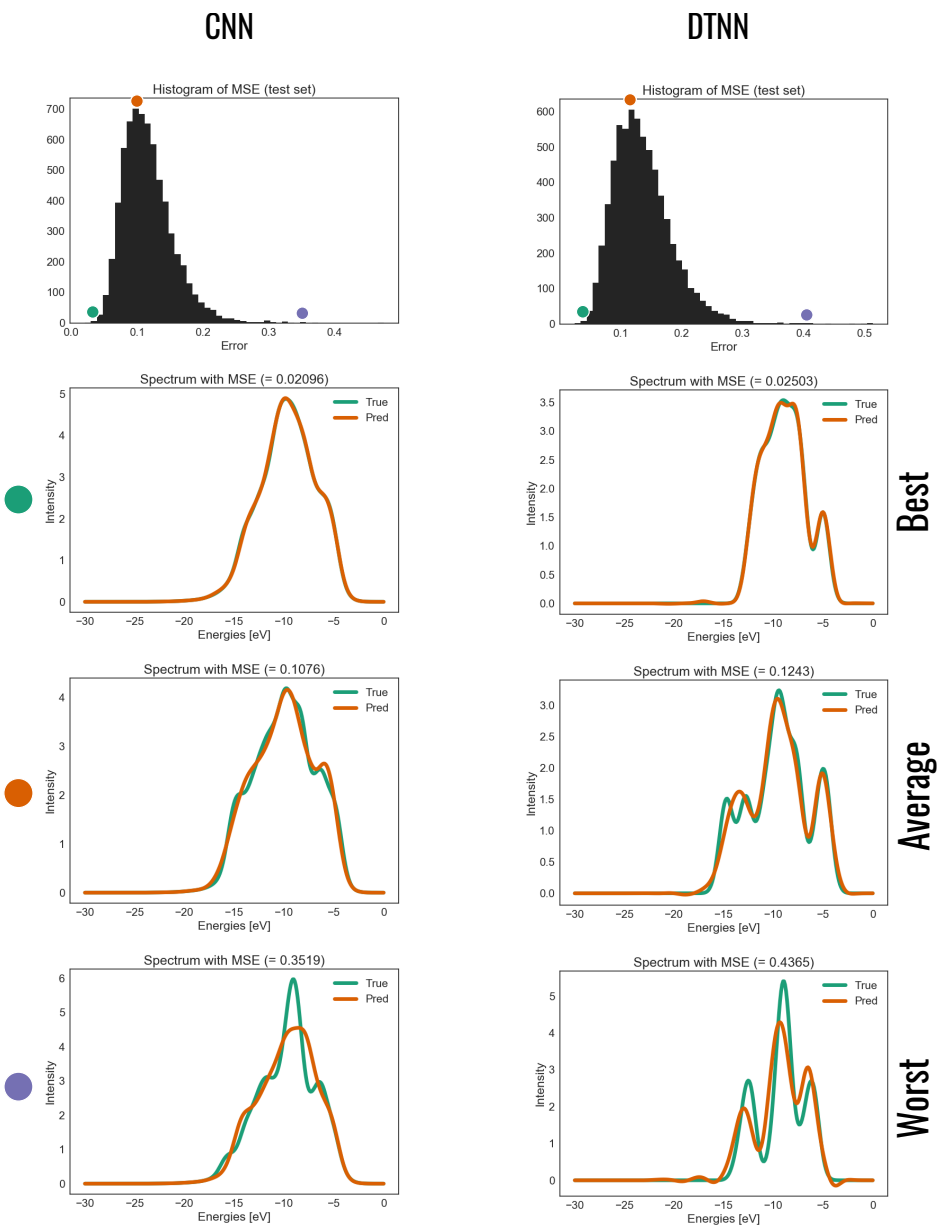


Figure 4.7: *Figure comparing the spectra predictions by CNN and DTNN. The histograms, on the first row, present the distribution of mean squared error (MSE) between predicted and true spectra of approximately 13000 test molecules from the 132K dataset. The plots following the histograms depict the predicted and true spectra of a test molecule and their MSE. The bins, of the histogram, from which the molecules were sampled are depicted by coloured circles.*

Chapter 5

Discussions and Conclusion

This thesis has presented deep learning models which have predicted energies corresponding to molecular orbitals and photoemission spectra, from the charges and inter-atomic distance of atoms constituting the molecules of interest. The thesis has also compared, in detail, the accuracy of three distinct neural network architectures for the prediction tasks.

From the experiments presented in Chapter 4 it is evident that, although it helps to have a large training dataset, a purpose designed machine learning (ML) model like the deep tensor neural network (DTNN) can out-perform generic models like the multi-layer perceptron (MLP) or models like the convolutional neural network (CNN). In addition, knowledge of *tricks of the trade*, like binarization of the input and adding noise to the model, can also have a significant impact on the performance and speed of training the neural networks. Given that, for the 132K dataset, HOMO predictions from CNN were quite close to those from DTNN. It would be interesting to investigate whether the predictive performance of CNN improves by adding noise to Coulomb matrices, which are input to the model. Since the off-diagonal entries of a Coulomb matrix are proportional to inverse inter-atomic distances, addition of noise can be achieved by scaling them by inverse of samples drawn from a standard Gaussian. This would be similar to adding noise to the inter-atomic distances. However, this is left for future work.

With the modifications to the DTNN proposed in this thesis, we were able to achieve a root mean square error (RMSE) of 0.186 for HOMO, HOMO-1, ..., HOMO-15 combined, surpassing the previously known state of the art RMSE of 0.21 by Montavon (2013) for HOMO prediction.

However, it is important to address the question of *how good, is good enough ?*. When addressing predictions made by ML models, there is no definitive answer, but the degree of accuracy depends on the application. Although not discussed in this thesis, it is important to study the applications which would

benefit from fast predictions[1] of HOMO energies and spectra. If a prediction error of approximately 0.18 eV for the top sixteen HOMO levels is adequate, then the methods proposed in this thesis are sufficient. If not, then there is a need for more accurate models. In the context of spectra prediction, neither model presented in this thesis predict the spectra well. If we recall how the spectra was computed (refer Figure 2.6) we notice that the peaks in the spectra represent the energies corresponding to the molecular orbitals. It is important to get the location and magnitude of the peaks in the spectra correctly to be able to identify the molecular orbital energies from the spectra. In that respect, it is interesting to note that the worst prediction by the DTNN in Figure 4.7 is *better* than the average prediction, since the location of the peaks were better predicted in the "worst" prediction compared to the average prediction where two neighboring peaks (as seen in the corresponding true spectra) were smoothed. This indicates that mean squared error (MSE) might not be the best cost function to optimize while learning spectra, paving the way for future work.

Furthermore, If we observe the illustration of DTNN (see Figure 3.4), we notice that the model is *not* invariant to the size of molecules. For molecular datasets with a large variation in size of molecules the input to DTNN would recieve a lot of zero-padded inputs. Thus, scope for future work lies in designing ML models invariant to the size of molecules.

In this thesis we were able to achieve our original objective and presented ML models capable of predicting molecular properties like HOMO energies and photoemission spectra. However, much work needs to be done before machine learning models can be deployed to realize the eventual goal of discovering novel materials.

[1] Predictions from the DTNN take only a few milliseconds.

References

Blum, L. C. and Reymond, J.-L. (2009). 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25):8732–8733.

Chu, S. and Majumdar, A. (2012). Opportunities and challenges for a sustainable energy future. *Nature*, 488(7411):294–303.

Du, K.-L. and Swamy, M. N. S. (2014). Multilayer Perceptrons: Other Learning Techniques. In *Neural Networks and Statistical Learning*, pages 127–157. Springer, London. DOI: 10.1007/978-1-4471-5571-3_5.

Faber, F. A., Hutchison, L., Huang, B., Gilmer, J., Schoenholz, S. S., Dahl, G. E., Vinyals, O., Kearnes, S., Riley, P. F., and von Lilienfeld, O. A. (2017). Fast machine learning models of electronic and energetic properties consistently reach approximation errors better than DFT accuracy. *arXiv:1702.05532 [physics]*. arXiv: 1702.05532.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Hansen, K., Montavon, G., Biegler, F., Fazli, S., Rupp, M., Scheffler, M., von Lilienfeld, O. A., Tkatchenko, A., and Müller, K.-R. (2013). Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419.

Hu, Z., Yang, Z., Salakhutdinov, R., and Xing, E. P. (2017). On Unifying Deep Generative Models. *arXiv:1706.00550 [cs, stat]*. arXiv: 1706.00550.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report, University of Toronto.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient BackProp. In Orr, G. B. and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, number 1524 in Lecture Notes in Computer Science, pages 9–50. Springer Berlin Heidelberg. DOI: 10.1007/3-540-49430-8_2.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Montavon, G. (2013). On layer-wise representations in deep neural networks. Doctoral Thesis, Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik.

Montavon, G., Rupp, M., Gobre, V., Vazquez-Mayagoitia, A., Hansen, K., Alexandre Tkatchenko, Müller, K.-R., and von Lilienfeld, O. A. (2013). Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003.

NREL (2017). National Center for Photovoltaics, Research Cell Record Efficiency Chart. https://www.nrel.gov/pv/ Accessed 4.8.2017.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:sdata201422.

Ruddigkeit, L., van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875.

Rupp, M. (2015). Machine learning for quantum mechanics in a nutshell. *International Journal of Quantum Chemistry*, 115(16):1058–1073.

Rupp, M., Tkatchenko, A., Müller, K.-R., and von Lilienfeld, O. A. (2012). Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Physical Review Letters*, 108(5):058301.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K.-R., and Tkatchenko, A. (2017). Quantum-Chemical Insights from Deep Tensor Neural Networks. *Nature Communications*, 8:13890. arXiv: 1609.08259.

Taylor, G. W. and Hinton, G. E. (2009). Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1025–1032, New York, NY, USA. ACM.

The GPyOpt authors (2016). GPyOpt: A Bayesian Optimization framework in python. `http://github.com/SheffieldML/GPyOpt` Accessed 4.8.2017.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

Yaeger, L. S., Webb, B. J., and Lyon, R. F. (1998). Combining Neural Networks and Context-Driven Search for Online, Printed Handwriting Recognition in the Newton. In Orr, G. B. and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, number 1524 in Lecture Notes in Computer Science, pages 275–298. Springer Berlin Heidelberg. DOI: 10.1007/3-540-49430-8_14.