ATOMIC NEURAL NETWORKS FOR PREDICTING MOLECULAR PROPERTIES

A thesis submitted in partial fulfilment of the requirements for the award of the degree of

INTEGRATED MASTER OF SCIENCE IN PHYSICS

Submitted by

HEMAPRASATH V

(Reg. No. I150117)

Under the guidance of

DR. M. PONMURUGAN

(Assistant Proffessor)

DEPARTMENT OF PHYSICS
SCHOOL OF BASIC AND APLLIED SCIENCES
CENTRAL UNIVERSITY OF TAMIL NADU
THIRUVARUR
MAY-2020

Declaration

I, HemaPrasath V, hereby declare that the project thesis, entitled *“Atomic Neural Networks for predicting Molecular Properties”* which is submitted by me in partial fulfilment of the requirements for the award of the degree of INTEGRATED MASTER OF SCIENCE in PHYSICS comprises only my original research work and due acknowledgement has been made in the text, is produced under the guidance of Dr. M. Ponmurugan, Assistant Professor, Department of Physics, School of Basic and Applied Sciences, Central University of Tamil Nadu, Thiruvarur.

Date:
Place: Thiruvarur

Signature of the Candidate
(HemaPrasath.V)

तमिलनाडु केन्द्रीय विश्वविद्यालय

(संसद द्वारा पारित अधिनियम 2009 के अंतर्गत स्थापित)

CENTRAL UNIVERSITY OF TAMIL NADU

(Established by an Act of Parliament, 2009)

CERTIFICATE

This is to certify that this project entitled **"ATOMIC NEURAL NETWORKS FOR PREDICTING MOLECULAR PROPERTIES"** submitted in partial fulfilment for the requirements for the award of the degree of **INTEGRATED MASTER OF SCIENCE** in **PHYSICS** to the Central University of Tamil Nadu, done by **Mr. HEMAPRASATH V**, Roll No. I150117 is an authentic work carried out by him at *Department of Physics, Central University of Tamil Nadu* under my guidance . The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Date:	Dr. M. Ponmurugan
Place: Thiruvarur	Project Supervisor

Head of the Department	Project Co-ordinator

Acknowledgement

I would like to express my gratitude to my supervisor Dr. M. Ponmurugan, Assistant Professor, Department of Physics, Central University of TamilNadu for the useful comments and all the support, motivation and encouragement he has given me throughout my project period. I extend my gratitude to the Head of the Department, Prof. L. Kavitha and all the faculty members of the department. I would like to thank the Ph.D students who worked in the same lab as me and my classmates for their advice and good intent along this work. I also thank my parents who have been in this through thick and thin with me.

Abstract

Atomic Neural Networks (ANNs) constitute a class of machine learning models establishing Quantitative structure-activity relationships (QSARs), structure-property relationships in molecules. PiNN, a Python library was developed for implementing interpretable ANN Architectures and existing architectures efficiently. It also offers built-in dataset loaders for ease of access for different formats of molecular data, custom ANN designs and model types. Molecular property is regressed against other molecular information using Machine Learning techniques. Here, Dipole moment is regressed against the structure of certain organic molecules, building a dipole model for predicton. The dipole model's accuracy is compared for different sample sizes of the number of organic molecules used for regression.

Contents

List of Figures

Chapter 1

Machine Learning Techniques

1.1 Introduction

One of the many Machine Learning techniques is to map a relation between information by inference without any explicit instructions. In many fields of science, mapping between informations can be used to establish relationship that help build an approximate function that in turn help build models to study further. As an Example, a function $f : \{x_i, Z_i\} \rightarrow P$ is used to map the structure of a molecule or a material to its property P, where $\{x_i, Z_i\}$ represent the structural information of the molecule/material. In Conventional methods, if P is the total energy, then approximate solutions to the Schrödinger equation[1] is found using traditional computational methods that are resource-intensive and available commercially even with today's better computational resources.

Machine Learning techniques recognise patterns from pre-existing information (from a database) to properly build functions that properly map relations to the property. This mapping function is then used to predict for new structures. The general task in machine learning is to determine what patterns to recognise from the database; to build the functions that map relations with ease and computational efficiency.

1.2 Neural Networks

Simple Neural Networks are directed 'graphs' of simulated nodes. A 'graph' consists of nodes and edges that connect them. The Neural Networks are inspired as a biological neuron. The nodes that connect to another node is one-directional i.e., the node that sends information to another node does not receive from the same. The nodes which connect to the next node belong to a single layer in the neural network. Each layer that sends information performs an operation on the information. The Neural Network Architecture refers to the number of layers, number of nodes in each of those layers, operation for each layer.

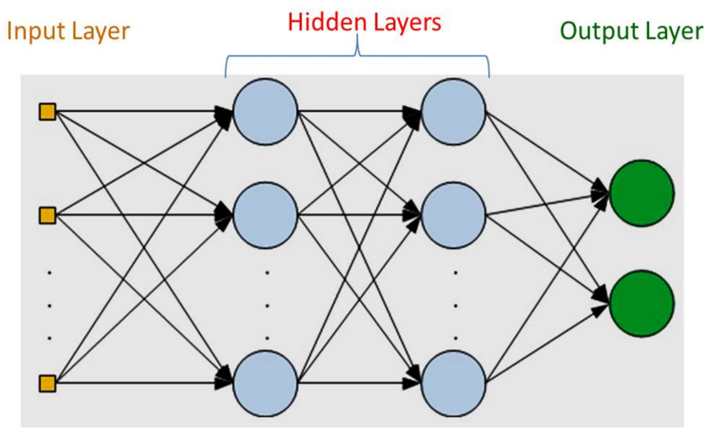


Figure 1.1: A Simplified Neural Network Architecture

Edges determine the weight of the connection between the nodes that connect them. The Hidden Layers operate on the information sent from the previous layer using a set of activation functions. Mathematially, if the inputs are $a_1, a_2, ..., a_n$ and the weights are $w_1, w_2, ..., w_n$, then the value output for a single node would be:

$$a^j = f(b + \sum_{i=1}^{n} a_i^{j-1} * w_i^{j-1}) \tag{1.1}$$

where f is the activation function of choice, b is the bias, a^j refers to the nodes in the j^{th} layer. Each a_i^{j-1} refers to the values of the previous layer and w_i, the weight between the nodes connecting the previous layer. Thus the output layer gives a combined value of all the hidden layers, depending on the weighted connections between nodes and the bias value b. Thus, the value of the output node will be:

$$y = f(b + \sum_{i=1}^{n} a_i^j * w_i^j) \tag{1.2}$$

The 'learning' aspect using pre-existing information takes place here. As the output value is determined by the bias b and weights between each nodes of every layers w_i^j, a cost function is introduced that gives us the error between the Neural network predicted value and the value from the database. Hence, it is necessary for the database to contain both the information that is used to predict and the predicted value to build a model. The cost function is of choice and depends on the model that is built. The cost function is then used to modify the weight values from Eq. 1.2.

The modification of the weight values are such that the cost function *reduces* over many iterations of the neural network through a huge database for generality. This is done by another operation using optimizers. A common optimizer operation is a stochastic gradient descent which finds the global minima for the cost function and the multiple weight values. Other optimizers can also be used depending on the complexity of the model and the efficiency required. A single best type of optimizer/activation functions cannot be used, as different optimizers fit in better for different models.

Certain hyperparameters are pre-defined before building a model that helps make the learning process efficient and accurate such as the activation functions, the learning rate and the architecture of the neural network itself. The learning rate specifies the rate at which the global minima of the cost function w.r.to the learnable parameters (weights and biases) is approached.

Machine Learning models contain various techniques for prediction, classification or recognizing patterns/trends[2]. Of which the above technique is a generic technique of using existing database of information from both the predictor and prediction, usually called the values and its labels i.e., each value is mapped to its label value. This mode of operation is called the supervised learning.

From this chapter, the concept of Machine Learning and its potential for predicting regressions and recognising patterns is introduced. The technique of supervised learning using pre-existing database of information for building prediction models of choice is understood. It is also understood that the machine learning techniques can be used to increase efficiency.

Chapter 2

Atomic Neural Networks

2.1 Introduction

Atomic Neural Networks (ANNs) are a class of machine learning models that help predict physico-chemical properties of molecules and materials. As mentioned before in 1.2, ANNs usually use supervised learning techniques. More advanced NN architectures are prepared for accuracy based off on this technique. The existing database for ANN models are collected from pre-existing information that are calculated by traditional methods.

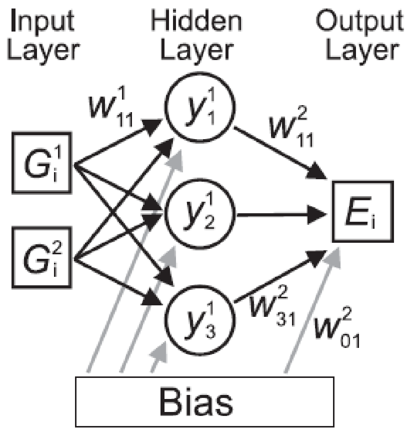


Figure 2.1: Example of a simple ANN model that predicts the energy E_i of a configuration i using the generalized coordinates G_i^1, G_i^2.

The node in the output layer yields the energy E_i of the configuration i depending on the generalized coordinates G_i^1 and G_i^2. The hidden layers connect the input and the output layer with a specified number of nodes. These nodes are connected by the weights between the nodes, which are real-valued and are chosen randomly, initially. For the example, the output of the energy configuration is given by the expression:

$$E_i = f_a^2[w_{01}^2 + \sum_{j=1}^{3} w_{j1}^2 f_a^1(w_{0j}^1 + \sum_{\mu=1}^{2} w_{\mu j}^1 G_i^\mu)] \tag{2.1}$$

The weights w_{ij}^k and the bias weight w_{0j}^k which are initially randomly chosen, does not correspond to the correct total energy output. Hence, to adjust these learnable parameters for proper prediction, the known total energies for a known set of configurations are used to optimize the set of these parameters, a cost function is constructed to minimize the error in an iterative way. The optimized set of 'learned' parameters can then be used to calculate the energy for a new set of coordinates.

Although this approach is novel[3], there are disadvantages associated with this setup. Namely, the coordinates of the configuration is fixed (number of degrees of freedom) and cannot be applied for higher set of coordinates as the optimized set of weights are vallid only for a specified size. Also, the weights of the NN are different for each configuration, meaning NN is not arbitrary towards similar configurations, allowing to predict falsely.

In general, the ANN models can be made to predict with greater accuracy and hence more efficiently. This means that the ANN model must preserve translational, rotational and even permutational invariance across different configurations of the system used to predict.

2.2 Generalized Atomic Neural Networks

Another ANN topology is introduced that addresses the challenges mentioned in 2.1. Generalized Atomic Neural Networks[4] were introduced that considers different system sizes and invariance in different configurations. The idea of empirical potentials of the total energy E of a configuration being represented as a sum of the atomic contrbutions E_i.

$$E = \sum_i E_i \tag{2.2}$$

From the input layer to the second hidden layer, the cartesian coordinates $\{R_i^\alpha\}$ or any form of coordinates that specify structural configuration of a system are transformed to a set of symmetry function values $\{G_i^\mu\}$. These symmetry functions represent the energetically relevant local environment of each atom i and are subsequently used for the input for the ANN. These function values depend on the positions of all the atoms in a system, shown by the dotted lines in Figure 2.2

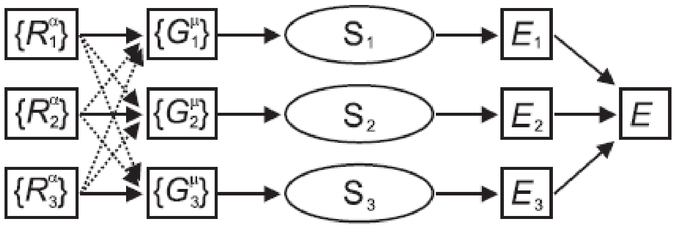


Figure 2.2: Structure of a Generalized Neural Network to a system containing three atoms

For each atom i in the system there is now a 'simple' ANN as mentioned in 2.1, which is called a subnet S_i and which after the optimization of the weights yields the energy contribution E_i of an atom in the system. Summing these energy contributions then finally yields the total energy of the system. To ensure the invariance of the total energy with respect to similar configurations of the system, the values of the weight parameters are constrained to be identical in each S_i of such system.

2.2.1 Symmetry and cut-off Functions

Radial Symmetry functions are constructed as a sum of Gaussians with the parameters η and R_s,

$$G_i^1 = \sum_{j \neq i}^{all} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij}) \tag{2.3}$$

Here, the symmetry function is set such that it is rotationally, translationally and permutationally invariant. Also, the number of symmetry functions is independent of the coordination number of an atom in the system, this is ensured by the summation over all the neighbours j for the atom i. The value of the Radial symmetry function depends on these parameters η, R_s and a cutoff function $f_c(R_{ij})$.

$$f_c(R_{ij}) = \begin{cases} 0.5 \times \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{for} R_{ij} \leq R_c \\ 0 & \text{for} R_{ij} > R_c \end{cases} \tag{2.4}$$

At interatomic distances R_{ij} larger than the cutoff radius R_c, this cutoff-function yields zero value and slope. The cutoff R_c has to be sufficiently large to include several nearest neighbours. This cutoff function helps determine the energetically relevant local environment for an atom i in the system.

Angular symmetry functions are constructed for all triplets of atoms by summing the cosine values of the angles $\theta_{ijk} = \frac{\mathbf{R_{ij} \cdot R_{ik}}}{R_{ij} R_{ik}}$ centered at

atom i, with $\mathbf{R_{ij}} = \mathbf{R_i} - \mathbf{R_j}$
.

$$G_i^2 = 2^{1-\zeta} \sum_{j,k\neq i}^{all} (1 + \lambda \cos \theta_{ijk})^{\zeta} \times e^{-\eta(R_{ij}^2 + R_{jk}^2 + R_{ik}^2)} f_c(R_{ij}) f_c(R_{jk}) f_c(R_{ik}), \tag{2.5}$$

with the parameters λ (=+1,-1), η and ζ. The multiplication by the three cutoff-functions and by the Gaussian gives a smooth decay to zero incase of large interatomic seperations. The use of symmetry functions and parameters is not unique, and many types of functions can be used to describe the suitable environment of an atom i.

In Conclusion, this generalized ANN architecture can be used to build models to predict for various system sizes and they provide with more accuracy, as the total energy is partitioned into effective atomic contributions[4] (Eq. 2.2).

Hence, this ANN describes the local chemical environment of each atom (also called the fingerprint of the atomic environment[5]) using the set of symmetry functions from Eqs. 2.3 and 2.5. This is generally called the Behler - Pahrinello Neural Network or the BPNN and has been succesfull for a wide-range of molecules and materials.

2.3 Graph Convolutional Atomic Neural Networks

In the field of Deep Learning is one of the most successful end-to-end techniques, Convolutional Neural Networks. The convolution in an NN, helps learn features (in terms of a simple ANN, the optimized set of weights) by creating a feature hierarchy. For Graph Convolutional Neural Networks (GCNN) in atomic systems, the molecules and materials can be viewed as fully connected graphs[6].

A graph neural network considers an atom as nodes and their pairwise-interactions as edges of a graph. These pairwise interactions are are weighted edges that define the interaction features between the atoms connecting them. The essence of a Convolution, gathers information for each atom from their neighbouring atoms within a cut-off radius R_c and creates a feature hierarchy. Node and edge feature vectors are updated through a function.

This way the features are learned automatically, instead of manually hand-crafting them. Hierarchical atomic features[7] are obtained by applying multi-stage concatenated convolution operations through a neural message passing function. By construction, the generation of the node features includes the element-specificity. Hence, GCNN has the same subnet for each element. This also proves effective performance for a variety of systems, also retaining invariance properties.

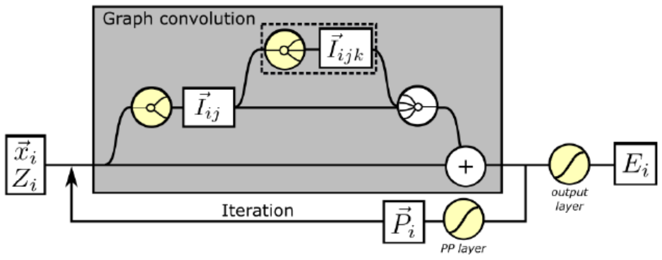


Figure 2.3: Representation of a GCNN framework in atomic systems

In Fig. 2.3, the input matrices $\vec{x}_i$ and Z_i provide the structural information of a system. The $\vec{I}_{ij}$ and $\vec{I}_{ijk}$ represent the pairwise and the triplewise interactions caused by the radial and the angular terms respectively. The Graph Convolution block (black box) is the convolution filter that creates a feature hierarchy iteratively updating the node and edge feature vectors.

This chapter introduces the concept of Atomic Neural Networks and its evolution with developments in the machine learning. It also shows the increased computational efficiency and accuracy achieved using these techniques, with accessible computational resources.

The ANN helps predict properties from the structural information as $f : \{x_i, Z_i\} \rightarrow P$ stated. The Generalized ANN partakes this concept to partition effective atomic contributions of these properties and introduces symmetry functions to build prediction models for various system sizes and ensure rotational, translational and permutational invariance. The Graph Convolutional NNs considers the molecular systems as undirected graphs and creates hierarchical features to learn pairwise interactions between atoms as atomic fingerprints[5]. These different variations help create sophisticated systems for building prediction models of various molecular system properties which in turn can be used to build physical models.

Chapter 3

PiNN Python Package for ANN

3.1 Introduction

Implementing Atomic Neural Networks through computational systems is the next step in the process of building the aforementioned NN models in 2. The computer code for the building of models can be done through multiple channels, and the choice is dependent on factors like the ease of computational coding in different platforms, complexity of the model that is required to build, or the freedom to implement wide-variety of systems. Each coding platform offers different features and the choosing of one platform does not mean that the one is a better.

The PiNN library[8] is implemented in Python to make the building of ANNs interpretable and implementing with comptuational ease efficiently. Hence, to promote the application of ANNs in the scientific communities, PiNN is a reliable, general-purpose, open-source-code.

Here, we use PiNN, a library in the Python coding platform for computational ease and also for building a variety of systems using the PiNN - Python Library. The code for the library is not implemented in the Python and is an external code distributed under a permisive BSD license and is freely accessible at https://github.com/Teoroo-CMC/PiNN/ with full documentation and tutorials.

3.2 Using the PiNN - Python Package

The PiNN library is built on top of the TensorFlow[10] by Google, which is generally used to implement machine learning and deep-learning models for general purposes. The TensorFlow library offers to use the Graphical Processing Units (GPUs) of computer systems for computational power. PiNN offers modularization of existing models (as discussed in

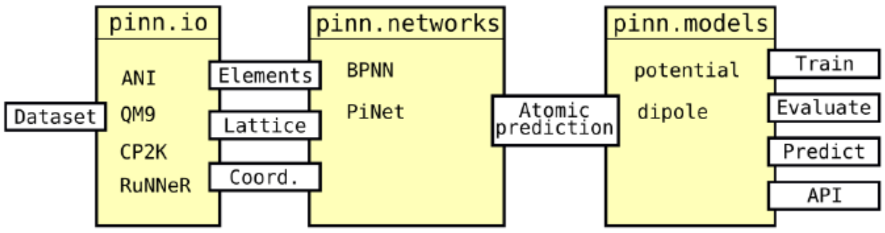


Figure 3.1: Illustration of a structure of an ANN using the PiNN library

2), which can be re-used for different uses. To reduce difficulty of the training tasks, customized dataset loaders are in-built within the library. Also, it has in-built physical models that can be built for prediction of certain properties.

Building of an ANN can be split into three stages and the PiNN modules consist of three modules in par to this:

- input/output (io) module for Preparing the datasets (as inputs for the ANN from various formats of data)
- Networks module for defining the ANN architecture (with the different layers and different nodes for each of these layers and their functions, also the final cost functions and optimization technique choices)
- Models for model defnition (to predict the required property that is regressed from the structural information)

The Modularization of networks can be done without touching the internal Python code base. This design enables to easily import an arbitrary dataset within the ANN.

One can also implement new ANN architectures from scratch in the "networks" module and use the other modules with code ease. Similarly, the "models" module could be extended to build for different property predictions with the same existing ANN architectures or to interface with external codes.

3.2.1 Pairwise Interaction and Interaction Pooling

In PiNN, both the BPNN and GCNN is defined with two important abstractions : The Pairwise Interaction (PI) and Interaction Pooling (IP) operations. Labelling each atom in the system with i and its atomic property $\vec{P_i}$, the PI is expressed as a function of the intial atomic properties of two atoms and their distance,

$$\vec{I^t_{ij}} = PI(\vec{P^t_i}, \vec{P^t_j}, r_{ij}) \tag{3.1}$$

where t is an iterator. For BPNN, $t = 0$ and for GCNN, $t + 1$ is the number of Graph Convolution (GC) blocks.

The IP is in contrast to the PI operation, where the pairwise interactions associated with the atom i generate the atomic property. This is done by passing all the pairwise interactions of the atom i and summing them to another function. The summation ensures permuational invariance of the generated atomic property.

$$\vec{P_i^{t+1}} = IP\left(\sum_j \vec{I_{ij}^t}\right) \tag{3.2}$$

The combination of the PI and IP operations used generates an updated atomic property through $t + 1$ iterations of GC blocks, with information collected from neighbouring atoms. This is referred to as the atomic fingerprint or a neural message passing function[9]. Through the iterations of GC blocks $PI + IP \rightarrow PI + IP, ..., \vec{I_{ij}}$ also gets updated. Although the actual forms of PI and IP operations differ from the mentioned, the abstraction remains, allowing to create different functions for novel ANN architectures.

3.3 PiNet - A GCNN variant of an ANN

The feature of PiNN that allows modularization of any existing ANN architecture is greatly exploited, as the library offers its own custom GCNN variant of an ANN architecture, allowing for interpretation of the ANN activations. The PiNet[8] uses GC blocks to iteratively update molecular features. The PiNet can be used for various other model definitions apart from the in-built models - Potential and Dipole models.

3.3.1 Architecture

In PiNet, the PI is defined as a function of distance between both the interacting atoms, affecting the atomic property in turn. The weight matrix $\mathbf{W_{ij}}$
depends on both atomic properties $\vec{P_i}$ and $\vec{P_j}$. This weight matrix generates the PI value $\vec{I_{ij}}$. The advantage of the weight matrix is to have different radial dependence for each component of $\vec{I_{ij}}$, which is unique to PiNet.

$$\mathbf{W_{ij}} = NN^{PI-Layer}([\vec{P_i}, \vec{P_j}]) \tag{3.3}$$

where NN is a feed-forward neural layer generating the weight matrix from the atomic properties of the interacting atoms. Expressing the interatomic distances through a radial symmetry function, e_{ij} (From Eq. 2.3):

$$\vec{e_{ij}} = f_c(r_{ij}) \cdot [e^{-\eta(r_{ij}-r_1)^2}, e^{-\eta(r_{ij}-r_2)^2}, \ldots] \tag{3.4}$$

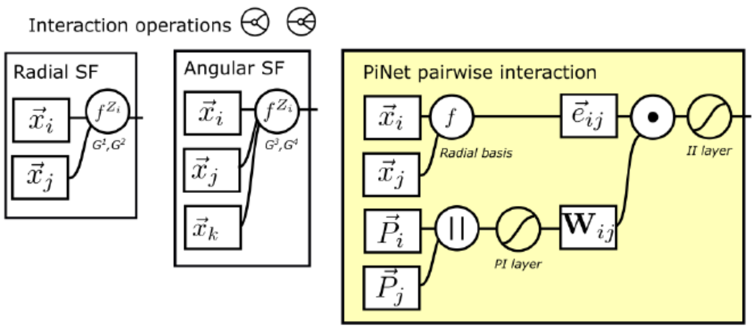


Figure 3.2: Illustration showing the Pairwise Interaction Operation in a PiNet

where the cutoff function is from the Eq.2.4 ensuring the interaction and its slope vanish at the cutoff radius R_c. The centers of the Gaussian functions $r_1, r_2, \ldots, r_{n_{basis}}$ are chosen to be evenly spaced between 0 and R_c, and the η determines the width of the Gaussians. Although, the freedom to integrate other cutoff functions are possible, the default cutoff function is set to this equation.

Then, the activation through the II layer (namely Interaction-to-Interaction), generates using another feed-forward NN using the information from the radial basis $\vec{e_{ij}}$ and the weight matrix $\mathbf{W_{ij}}$
.

$$\vec{I_{ij}} = NN^{II-Layer}(\mathbf{W_{ij}}\vec{e_{ij}}) \tag{3.5}$$

This process of generating the PI operation makes PiNet unique in contrast to other approaches where the interaction is directly generated from the distance and the atomic properties. It is to be noted that, despite not including a triplewise interaction part, the angular information is captured through the multiple iterations of the GC block.

After the PI Layer and the II Layer, the updated atomic property is calculated from all the pairwise interactions $\vec{I_{ij}}$ as part of the IP operation (As per Eq. 3.2) through another feed-forward NN layer. And for further refinement of the atomic property, another PP (Property-to-Property) feed-forward NN layer is passed. In this approach, each feed-forward NN layer has its own unique activation functions. After multiple iterations through the GC block, the output layer is then used for optimization of the final set of weights to minimize errors.

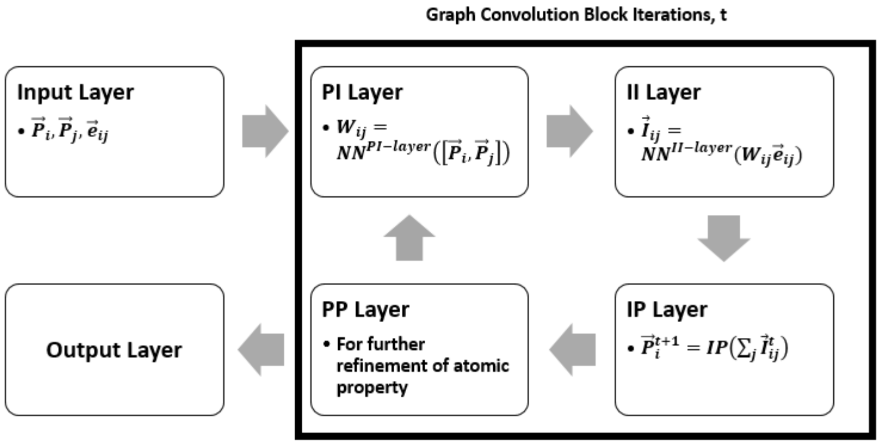

Graph Convolution Block Iterations, t
Input Layer
• $\vec{P}_i, \vec{P}_j, \vec{e}_{ij}$
PI Layer
• $W_{ij} = NN^{PI-layer}([\vec{P}_i, \vec{P}_j])$
II Layer
• $\vec{I}_{ij} = NN^{II-layer}(W_{ij}\vec{e}_{ij})$
IP Layer
• $\vec{P}_i^{t+1} = IP(\sum_j \vec{I}_{ij}^t)$
PP Layer
• For further refinement of atomic property
Output Layer

Figure 3.3: Schematic Representation of a basic PiNet Architecture

3.4 Building a Dipole model using PiNet

As an example of property predictions, PiNet ANN Architecture is used to regress the molecular dipole moment and partial charges, accordingly,

$$\mu = \left| \sum_{i=1}^{N} \tilde{q}_i \vec{r_i} \right| \tag{3.6}$$

where $\tilde{q}_i$ is the predicted partial charge on atom i and the molecular dipole moment is acquired from a database. PiNN library offers for computational efficiency and ease. The property prediction is done in accordance to improve computational efficiency with limited power and resources. The aim is to predict values with higher accuracy using only these limited resources.

Hence, multiple ANN model predictors are trained based on the corresponding computational resources the training model requires, ranging from highly accurate models (requiring higher computational power and time) to low accurate models (hence, only requiring lower computational time and power).

3.4.1 QM9 Dataset

No.	Property	Description	Unit
1	tag	'gdb9' string to facilitate extraction	-
2	i	Consecutive, 1-based integer identifier	-
3	A	Rotational Constant	GHz
4	B	Rotational Constant	GHz
5	C	Rotational Constant	GHz
6	μ	Dipole moment	D
7	α	Isotropic Polarizability	a_0^3
8	ϵ_{HUMO}	Energy of HUMO	Ha
9	ϵ_{LUMO}	Energy of LUMO	Ha
10	ϵ_{gap}	Gap(ϵ_{LUMO}-ϵ_{HUMO})	Ha
11	$\langle R^2 \rangle$	Electronic spatial extent	a_0^2
12	zpve	Zero point vibrational energy	Ha
13	U_0	Internal Energy at 0 K	Ha
14	U	Internal Energy at 298.15 K	Ha
15	H	Enthalpy at 298.15 K	Ha
16	G	Free Energy at 298.15 K	Ha
17	C_v	Heat capacity at 298.15 K	Cal/mol.K

Table 3.1: List of properties reported in a QM9 dataset for molecules

The QM9 dataset[11] is a database made of 134,000 small organic molecules containing computed geometric, electronic, energetic and thermodynamic properties at B3LYP/6-31G(2df,p) level of theory[12], is a highly recommended for benchmarking ANNs. The organic molecules are composed of all molecules of upto 9 heavy atoms.

Other informations about the properties (listed in the Table 3.1) are also reported in this dataset for their corresponding organic molecules. The Table 3.1 is in the format provided from the dataset.

3.4.2 Benchmarks

The Dipole model in-built with PiNN is used with constraint terms that ensure partial charges of each of the organic molecule to be zero. For building model variants based on computational power used, models with varying number of training data is used i.e., each model is built with random QM9 molecules with varying sample sizes. The N number of molecules used for training and testing are:

[10,50,100,200,500,1000,1500,2000,4000,5000,6500,8500,10000]. A total of 13 dipole models are built. Each model uses the same hyperparameters (listed in 3.2)

Hyperparameter	Value
PI Layer	[64]$\times$ 10
II Layer	[64,64,64,64]
PP Layer	[64,64,64,64]
Output Layer	[64,1]
R_c	4.5 Å
Basis	Gaussian
GC blocks	5
η	3.0 Å^{-2}
η_{basis}	10
Activation (each layer)	Hyperbolic Tangent
Learning rate	3.0 $\times 10^{-4}$
Training Steps	10000

Table 3.2: Hyperparameters used for Dipole model building with N molecules

The QM9 molecular samples are split into training set and validation set with a ratio of 80:20, and the training samples are split into mini-batches with the following condition:

$$mini - batch - size = \begin{cases} N & \text{for N} \leq 100 \\ 100 & \text{for N} > 100 \end{cases} \tag{3.7}$$

As the Number of molecules used for training for comparing accuracies, the cutoff and the radial basis functions are defaulted (3.2.1).

3.5 Results and Discussions

The Dipole models built provided with reasonable accuracy, with increase in number of molecules, the error in the molecular dipole moment μ is minimized. The accuracy is compared using the validation split of the QM9 dataset in each model. Since, PiNN is built on top of the TensorFlow[10] Framework, the Tensorboard utility is used to visualize the training of the dipole model built for each sample size.

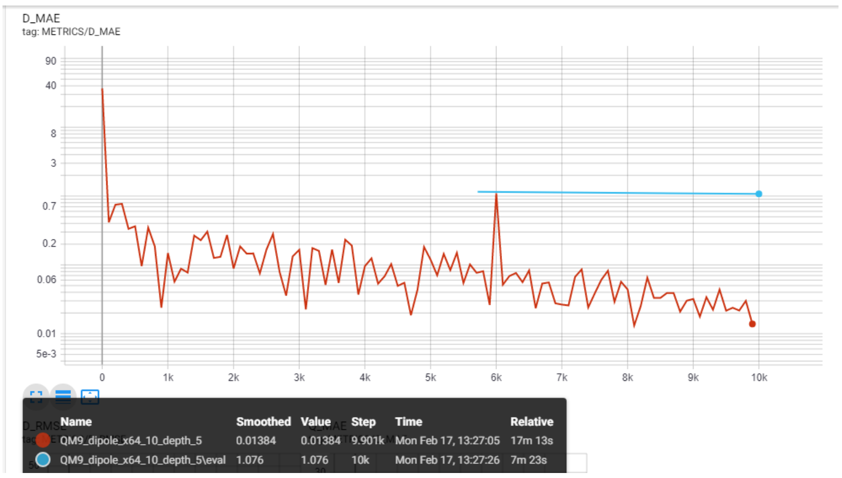
The decreasing trend seems to follow a logarithmic trendline, as the decrease in error for each model as the training size is increased, i.e., the accuracy of the model is more or less the same for even a significant increase in the training size (for larger training sizes).

The error function that the building of the model is trained using the Mean Absolute Error (MAE), whereby the molecular dipole moment error μ_{MAE} is calculated as:

$$\mu_{MAE} = \left|\mu_{predicted} - \mu_{database}\right| \tag{3.8}$$

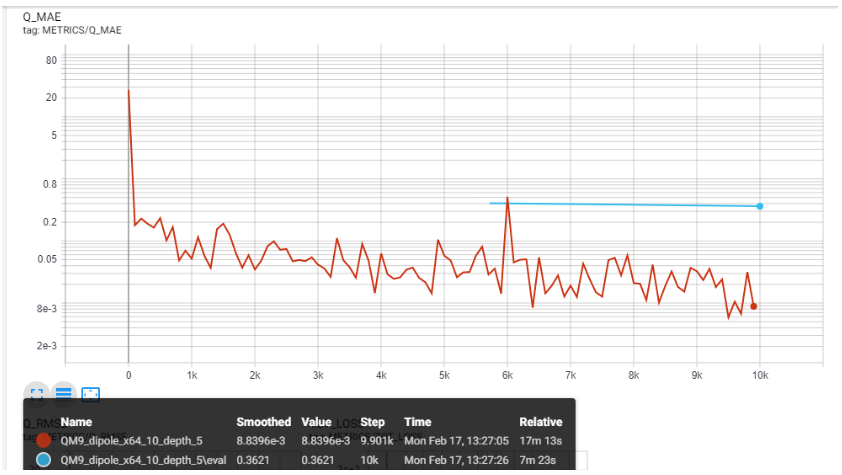
Similarly, using the dipole moment prediction, the value of the partial charges, $\tilde{q}_i$ are predicted using the Eq. 3.6. Then, the error in the partial charges $\tilde{q}_{i_{MAE}}$ is given by:

$$\tilde{q}_{i_{MAE}} = \left| \tilde{q}_{i_{predicted}} - \tilde{q}_{i_{database}} \right| \tag{3.9}$$



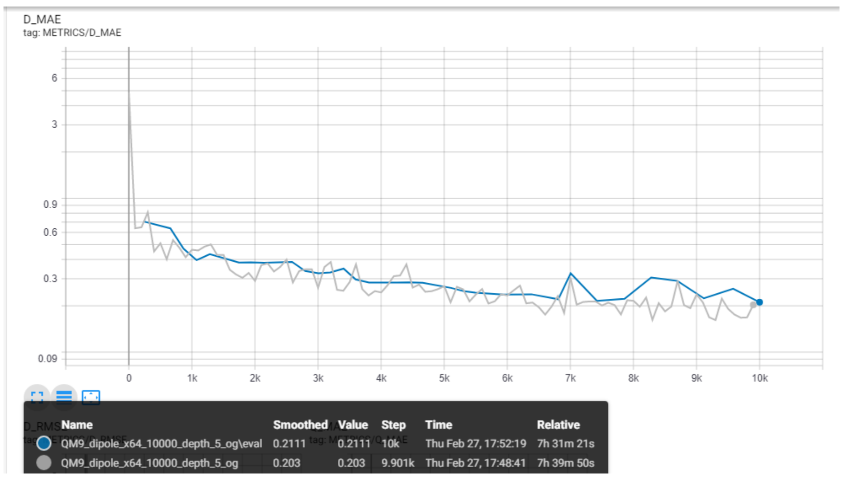
	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10_depth_5	0.01384	0.01384	9.901k	Mon Feb 17, 13:27:05	17m 13s
●	QM9_dipole_x64_10_depth_5\eval	1.076	1.076	10k	Mon Feb 17, 13:27:26	7m 23s

Figure 3.4: The MAE trend for molecular dipole moment upto 10k training steps built using (N=10) QM9 molecules with a training time of 17 minutes



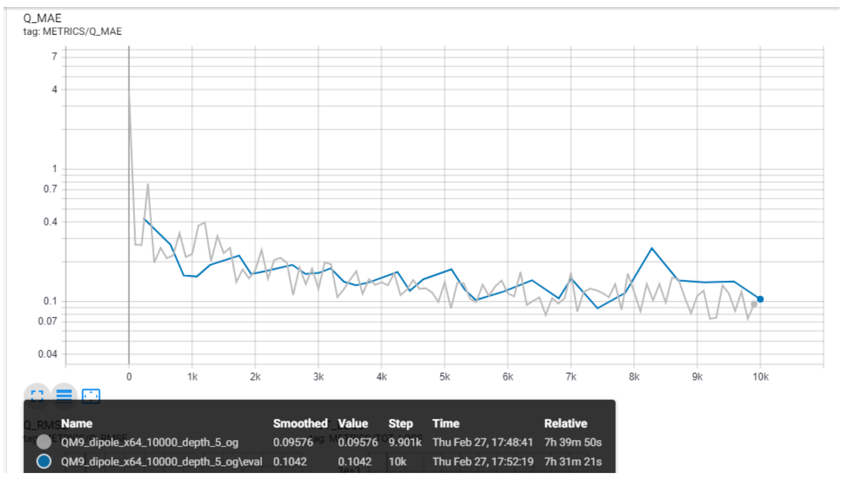
	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10_depth_5	8.8396e-3	8.8396e-3	9.901k	Mon Feb 17, 13:27:05	17m 13s
●	QM9_dipole_x64_10_depth_5\eval	0.3621	0.3621	10k	Mon Feb 17, 13:27:26	7m 23s

Figure 3.5: The MAE trend for partial charges regressed against molecular dipole moment upto 10k training steps built using (N=10) QM9 molecules with a training time of 17 minutes



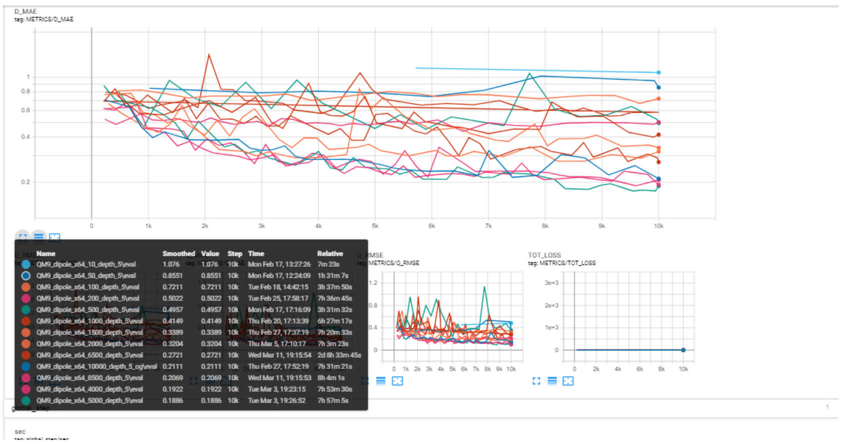
	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10000_depth_5_og\eval	0.2111	0.2111	10k	Thu Feb 27, 17:52:19	7h 31m 21s
●	QM9_dipole_x64_10000_depth_5_og	0.203	0.203	9.901k	Thu Feb 27, 17:48:41	7h 39m 50s

Figure 3.6: The MAE trend for molecular dipole moment upto 10k training steps built using (N=10000) QM9 molecules with a training time of 7 hours and 40 minutes



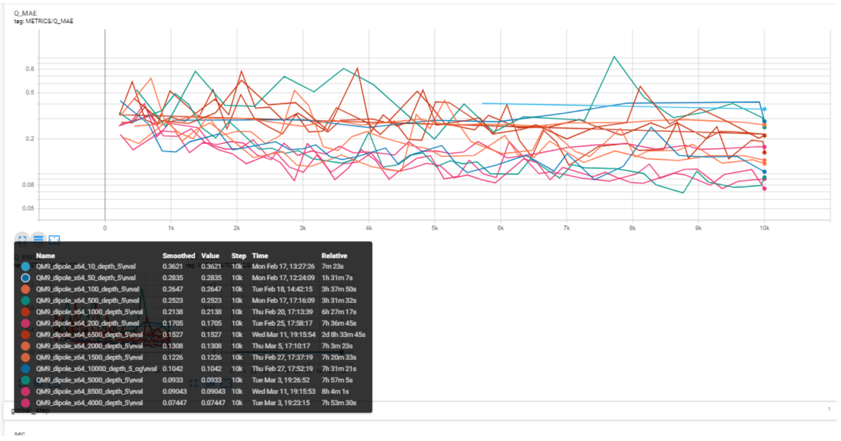
	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10000_depth_5_og	0.09576	0.09576	9.901k	Thu Feb 27, 17:48:41	7h 39m 50s
●	QM9_dipole_x64_10000_depth_5_og\eval	0.1042	0.1042	10k	Thu Feb 27, 17:52:19	7h 31m 21s

Figure 3.7: The MAE trend for partial charges regressed against the molecular dipole moment upto 10k training steps built using (N=10000) QM9 molecules with a training time of 7 hours and 40 minutes



	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10_depth_5\eval	1.076	1.076	10k	Mon Feb 17, 13:27:26	7m 23s
○	QM9_dipole_x64_50_depth_5\eval	0.8551	0.8551	10k	Mon Feb 17, 12:24:09	1h 31m 7s
●	QM9_dipole_x64_100_depth_5\eval	0.7211	0.7211	10k	Tue Feb 18, 14:42:15	3h 37m 50s
●	QM9_dipole_x64_200_depth_5\eval	0.5022	0.5022	10k	Tue Feb 25, 17:58:17	7h 36m 45s
●	QM9_dipole_x64_500_depth_5\eval	0.4957	0.4957	10k	Mon Feb 17, 17:16:09	3h 31m 32s
●	QM9_dipole_x64_1000_depth_5\eval	0.4149	0.4149	10k	Thu Feb 20, 17:13:39	6h 27m 17s
●	QM9_dipole_x64_1500_depth_5\eval	0.3389	0.3389	10k	Thu Feb 27, 17:37:19	7h 20m 33s
●	QM9_dipole_x64_2000_depth_5\eval	0.3204	0.3204	10k	Thu Mar 5, 17:10:17	7h 3m 23s
●	QM9_dipole_x64_6500_depth_5\eval	0.2721	0.2721	10k	Wed Mar 11, 19:15:54	2d 8h 33m 45s
●	QM9_dipole_x64_10000_depth_5_og\eval	0.2111	0.2111	10k	Thu Feb 27, 17:52:19	7h 31m 21s
●	QM9_dipole_x64_8500_depth_5\eval	0.2069	0.2069	10k	Wed Mar 11, 19:15:53	8h 4m 1s
●	QM9_dipole_x64_4000_depth_5\eval	0.1922	0.1922	10k	Tue Mar 3, 19:23:15	7h 53m 30s
●	QM9_dipole_x64_5000_depth_5\eval	0.1886	0.1886	10k	Tue Mar 3, 19:26:52	7h 57m 5s

Figure 3.8: The MAE trend for molecular dipole moment for all the 13 dipole models trained upto 10k training steps built using N=10,50,100,200,500,1000,1500,2000,4000,5000,6500,8500,10000 QM9 molecules



	Name	Smoothed	Value	Step	Time	Relative
●	QM9_dipole_x64_10_depth_5\eval	0.3621	0.3621	10k	Mon Feb 17, 13:27:26	7m 23s
○	QM9_dipole_x64_50_depth_5\eval	0.2835	0.2835	10k	Mon Feb 17, 12:24:09	1h 31m 7s
●	QM9_dipole_x64_100_depth_5\eval	0.2647	0.2647	10k	Tue Feb 18, 14:42:15	3h 37m 50s
●	QM9_dipole_x64_500_depth_5\eval	0.2523	0.2523	10k	Mon Feb 17, 17:16:09	3h 31m 32s
●	QM9_dipole_x64_1000_depth_5\eval	0.2138	0.2138	10k	Thu Feb 20, 17:13:39	6h 27m 17s
●	QM9_dipole_x64_200_depth_5\eval	0.1705	0.1705	10k	Tue Feb 25, 17:58:17	7h 36m 45s
●	QM9_dipole_x64_6500_depth_5\eval	0.1527	0.1527	10k	Wed Mar 11, 19:15:54	2d 8h 33m 45s
●	QM9_dipole_x64_2000_depth_5\eval	0.1308	0.1308	10k	Thu Mar 5, 17:10:17	7h 3m 23s
●	QM9_dipole_x64_1500_depth_5\eval	0.1226	0.1226	10k	Thu Feb 27, 17:37:19	7h 20m 33s
●	QM9_dipole_x64_10000_depth_5_og\eval	0.1042	0.1042	10k	Thu Feb 27, 17:52:19	7h 31m 21s
●	QM9_dipole_x64_5000_depth_5\eval	0.0933	0.0933	10k	Tue Mar 3, 19:26:52	7h 57m 5s
●	QM9_dipole_x64_8500_depth_5\eval	0.09043	0.09043	10k	Wed Mar 11, 19:15:53	8h 4m 1s
●	QM9_dipole_x64_4000_depth_5\eval	0.07447	0.07447	10k	Tue Mar 3, 19:23:15	7h 53m 30s

Figure 3.9: The MAE trend for partial charges for all the 13 dipole models trained upto 10k training steps built using N=10,50,100,200,500,1000,1500,2000,4000,5000,6500,8500,10000 QM9 molecules

The Figures 3.8 and 3.9 both show *the decreasing trend of the MAE values* for all the 13 dipole models using N QM9 molecules built with various sample training sizes. It is apparent that increasing the training sample size, decreases the MAE value causing the dipole model to be more accurate in this trend. For the purpose of testing the ANNs with different training sample sizes, we focus on the final MAE value at the last training step.

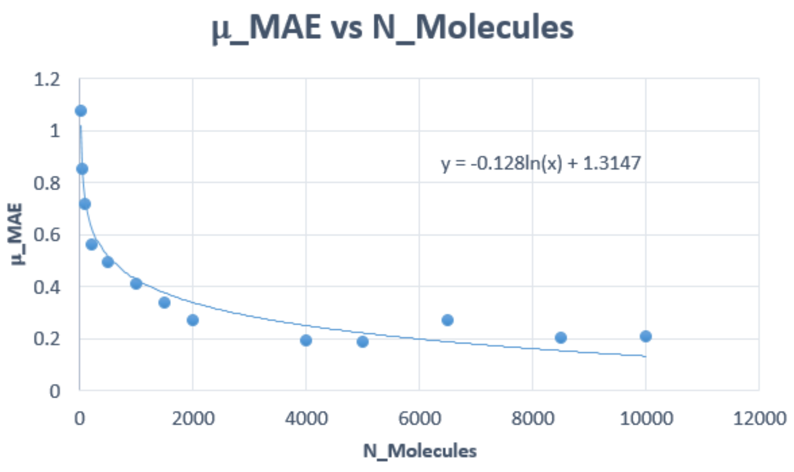


Figure 3.10: The Logarithmic trend for the MAE of molecular dipole moment μ at 10k training steps built using N=10,50,100,200,500,1000,1500,2000,4000,5000,6500,8500,10000 QM9 molecules

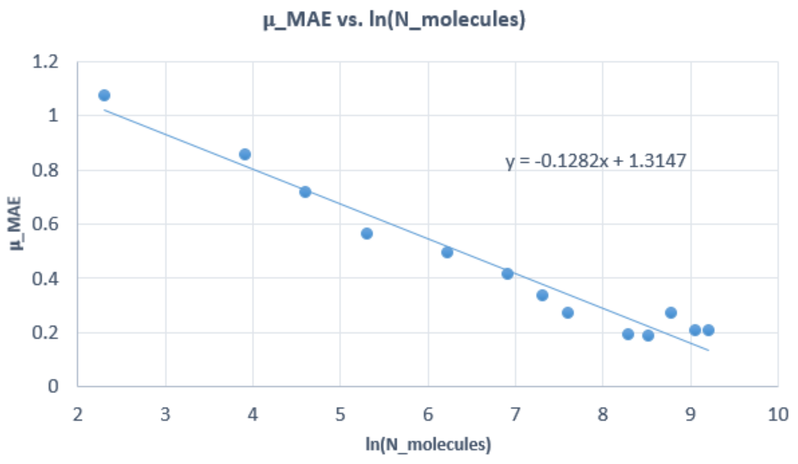


Figure 3.11: The Semi-Logarithmic trend for the MAE of molecular dipole moment μ at 10k training steps built using N=10,50,100,200,500,1000,1500,2000,4000,5000,6500,8500,10000 QM9 molecules gives a slope value of -0.128

The Logarithmic trend follows the equation,
$\mu_{MAE} = -0.128ln(N) + 1.3147$. This equation for the logarithmic trend in error values can be used as an indicator for the number of molecules N required for training set to predict the molecular dipole moment μ with an expected accuracy.

Chapter 4

Conclusions

Using the PiNN - Python Library for building Atomic Neural Networks for molecules and materials, a quantitative structure-property relation was established for the molecular dipole moment of a database of molecules (QM9 Dataset).

Molecular dipole moment is predicted using PiNet with different training sizes and their accuracies are compared with exact values. The result showed that the error decreases with the increasing number of trained molecules which follows the logarithmic trend.

The result showed the logarithmic trend in error values which can be used as an indicator for number molecules required for training set to predict the molecular dipole moment with an expected accuracy. Thus, building ANNs with improved accuracy and minimizing computational time.

Bibliography

[1] Dirac, P. A. M. *Quantum mechanics of many-electron systems.* Proc. R. Soc. London, Ser. A 1929, 123, 714–733.

[2] Michael A. Nielsen, *Neural Networks and Deep Learning*, Determination Press 2015.

[3] Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. *Machine learning for molecular and materials science.* Nature 2018, 559, 1–9.

[4] Behler, J.; Parrinello, M. *Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces.* Phys. Rev. Lett. 2007, 98, 146401.

[5] Duvenaud, D. Maclaurin, D. Aguilera-Iparraguirre, J. Gómez-Bombarelli, R.Hirzel, T. i. Aspuru-Guzik, A.Adams, R. P. *Convolutional networks on graphs for learning molecular fingerprints.* Adv. Neural Inf. Process. Syst. 2015; pp 2224–2232.

[6] Schütt, K. T. Arbabzadah, F.Chmiela, S. Müller, K. R.Tkatchenko, A. *Quantum-chemical insights from deep tensor neural networks.* Nat. Commun. 2017, 8, 13890.

[7] Lubbers, N.; Smith, J. S.; Barros, K. *Hierarchical modeling of molecular energies using a deep neural network.* J. Chem. Phys. 2018, 148, 241715.

[8] Yunqi Shao, Matti Hellström, Pavlin D. Mitev, Lisanne Knijff, and Chao Zhang, *PiNN: A Python Library for Building Atomic Neural Networks of Molecules and Materials.* arXiv:1910.03376v1 [physics.comp-ph] 8 Oct 2019

[9] Behler, J. *Constructing High-Dimensional Neural Network Potentials: A Tutorial Review.* Int. J. Quantum Chem. 2015, 115, 1032–1050.

[10] Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.;

Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mane, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viegas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.* 2016.

[11] Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. *Quantum chemistry structures and properties of 134 kilo molecules.* Sci. Data 2014, 1, 140022.

[12] Becke, A. D. *Density-functional thermochemistry. III. The role of exact exchange.* J. Chem. Phys. 1993, 98, 5648–5652.