PREDICTION OF ELASTIC AND OPTICAL PROPERTIES OF BINARY GLASS SYSTEM USING ARTIFICIAL INTELLIGENCE APPROACH

PRASAD A/L SOUNDRARAJAN

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering (Hons) Electronic Engineering

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

May 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : PRASAD A/L SOUNDRARAJAN

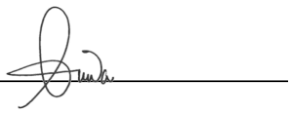
ID No. : 18AGB05217

Date : 6/5/24

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"PREDICTION OF ELASTIC AND OPTICAL PROPERTIES OF BINARY GLASS SYSTEM USING ARTIFICIAL INTELLIGENCE APPROACH"** was prepared by **PRASAD A/L SOUNDRARAJAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Nuraidayani Binti Effendy

Date : 14/5/24

Specially dedicated to
my beloved Grandmother, Mother and Father

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Nuraidayani Binti Effendy for her invaluable advice, guidance, and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents whose prayers have given me strength throughout this course and my friends who had helped and given me encouragement throughout this course. Finally, I would like to thank God for giving me the strength to complete this course.

PREDICTION OF ELASTIC AND OPTICAL PROPERTIES OF BINARY GLASS SYSTEM USING ARTIFICAL INTELLIGENCE APPROACH

ABSTRACT

Binary glass systems are a rising prospect in the industrial field and have piqued interest in research and the electronic field. The unique properties and distinctive structure of the binary glass systems provide a wide range of applications in the electronic field such as optical fibers, optical switching devices, laser hosts, and more. Additionally, during the manufacturing process of the binary glass, certain simulations is necessary to predict the characteristics of the glass before the pure materials of oxide are melted. Previous research has suggested and implemented the usage of artificial neural network models as instruments to simulate and predict the optical and elastic properties of binary glass series ZnO-TeO_2 glasses. Based on previous results, MATLAB software was used to predict the properties of the glasses, and sufficient results were produced for different types of ZnO-TeO_2 glass compositions. However, there was a drawback using MATLAB where the perfect fit correlation value, R^2 which represents the proportion of variance in the dependent variable that is predictable from the independent variable in a regression model is satisfactory as the correlation value R was all between 0.90361 and 0.99985. Nevertheless, the research established that the use of the ANN model is a good approach to be used in future research. In this project, python software with deep learning libraries such as PyTorch and scikit-learn was utilized to predict the elastic and optical properties of several glass series with different compositions. Thus, the results produced had a better and consistent R^2 value within a range of 0.97 to 0.99 which indicates a high degree of predictability in the relationship between the independent and dependent variables in the model. Furthermore, the training total loss on binary glass characteristics data set, graphs of predicted values, and real values were visualized, discussed, and studied in this report.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ZnO	Zinc Oxide
B_2O_3	Bismuth (III) oxide
SiO_2	silica
B_2O_2	boron oxide
Na_2O	sodium oxide
CaO	calcium oxide
Al_2O_3	aluminum oxide
TeO_3	tellurium trioxide
m	mass
η	pressure ratio
ρ	density, kg/m^3
v	volume
V_m	Molar Volume
M	molar mass
V_p	Longitudinal Velocity
K	Bulk Modulus
u	Shear Modulus

E	Young Modulus
σ	stress applied to material
ε	strain of the material

ANN	Artificial Neural Network
CNN	Convolution Neural Network
RNN	Recurrent Neural Network
MSE	Mean squared root error
R^2	R-squared value

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Glass materials are a growing prospect in the field of electronics due to their eccentric properties such as transparency, chemical inertness, thermal stability, and mechanical strength. Glass materials have a wide range of applications in the industrial field such as in chemical processing, laboratory equipment, insulation materials, manufacturing equipment, and optical components. Furthermore, binary glass systems refer to a type of glass composition that consists of a combination of two chemical compositions which mainly comprise oxides to form a glassy material. The most contemporary chemical component used is silica SiO_2 which forms the backbone of the glass structure. Thus, the second chemical component can vary depending on the desired properties of the glass and the intended applications. Besides that, the properties of a binary glass system are determined by the composition and structure of the glassy matrix. For instance, the addition of different oxides can change the optical, mechanical, thermal, and electrical properties of the glass. Hence, by varying the composition of the binary glass system, industrialists can tailor the glass to meet specific performance requirements for various applications. Henceforth, binary glass systems find widespread use across various industries due to their versatility and tunable properties.

1.2 Problem Statements

Glass fabrication is the process of manufacturing and assembling glass and glass parts using a variety of materials and technical processes. The usual method for fabricating glass includes engineering, machining, forming encapsulation, bevelling, extrusion, and moulding. However, the approach for fabricating binary glasses differs as it uses the trial-and-error method to fabricate the specific glasses. The trial-and-error method is used on binary glasses fabrication to discover its parameters however the trial-and-error method comes with several drawbacks. The trial-and-error approach in fabrication can ultimately lead engineers to a repetitive path that creates the preliminary design, putting it into production and often causing flaws to be detected lately. Finding the defects late when another process is present in production becomes a costly situation to recover in terms of money and time. Thus, discovering flaws in the fabrication approach could be a prominent key to cost and time saving. Hence, using simulation technologies like deep learning such as the ANN, design errors can be eliminated earlier and in the case of binary glass fabrication, the parameters of the glass system could be predicted earlier rather than using the trial-and-error approach which consumes time and money.

Traditional scientific theoretical models and experimental approaches in materials science often struggle to accurately predict the properties and behaviour of complex materials composed of more than five components. These limitations hinder the development and optimization of advanced materials for various applications. To address this challenge, we seek to use the flexibility and adaptability of Artificial Neural Networks (ANNs) to model and predict the properties of complex materials with compositions involving numerous elements and compounds. Furthermore, ANNs can overcome the constraints of conventional methods and provide a more accurate and versatile approach to predict the parameters of the binary glass system.

1.3 Aims and Objectives

The objectives of the thesis are shown as follows:

i) To develop an artificial neural network using Python software to predict elastic and optical properties of some binary borate compounds and tellurite glass systems.

ii) To simulate some elastic and optical properties of binary borate and tellurite glasses containing Zinc Oxide, (ZnO) and Bismuth (III) Oxide (B_2O_3) using an artificial neural network.

iii) To compare and quantitively analyse the experimental data from physical, elastic, and optical properties with the prediction of an artificial neural network.

1.4 Outline Of Report

Chapter 1 will be the introduction of the report. This chapter will consist of the background of the study of the project which illustrates a brief introduction of the research on the project and a general explanation of the project. Subsequently, this chapter will have the problem statements of the project where the problems, limitations, and solutions for the projects are discussed and finally, this chapter will be closed with the objectives and aims of the project where the goals to be achieved by to project are stated.

Chapter 2 will consist of the literature review of the project. In this section, an overview of previous research and experiments published related to the project being conducted will be studied and discussed. Furthermore, this chapter will demonstrate the knowledge and understanding of the project with several references from academic literature, research, and some online websites.

Chapter 3 will be the methodology of the project. In this subsequent chapter, the procedure and planning to conduct this project with detailed methods and techniques to assist in carrying out this project will be elaborately explained. In addition, several diagrams, flowcharts, models, and Gantt charts will be included in this chapter.

Chapter 4 will consist of the results and discussions of the project. This section will extensively analyse and study the outcome of the project. Furthermore, every data obtained from the experiment will be tabulated in this section and will be explained in full detail. In general, this section will show a presentation of the results in line with the objectives of the project.

Chapter 5 which is the final chapter will compromise of the conclusion and limitations of the project. Furthermore, this section will discuss the objectives achieved by this project and elaborate on the drawbacks of the project and provide some suggestions to improve the project for future usage.

CHAPTER 2

LITERATURE REVIEW

2.1 Binary Glass Systems

Binary glass systems are glass compositions made up of two main chemical elements. Glass is the most rudimentary form is an amorphous solid material shaped by the rapid cooling of a molten substance. Furthermore, in binary glass systems, the glass - forming elements are usually oxides such as silica, SiO_2, boron oxide and B_2O_2 combined with another oxide or modifier such as metal oxides like sodium oxide Na_2O, calcium oxide, CaO, aluminium oxide, Al_2O_3, and others. The various properties of binary glass systems depend on the chemical composition, the amount of glass forming oxides to modifiers, and the specific interactions between the elements. Some of the examples of the properties are the physical, elastic, and optical properties. Hence, by adjusting the composition of binary glass systems, manufacturers can alter the properties of the glass to suit specific applications. Additionally, these glass systems provide a huge range of properties depending on the specific composition making them versatile materials for various applications in the industry.

There are several types of binary glass systems used in the industry where each of them has its own distinctive properties. One of the most common binary glass materials used is zinc oxide, ZnO which is a combination between zinc and oxide. ZnO is an appealing material for applications in electronics, photonics, acoustics, and sensing. In the electronic field, ZnO has potential in transparent thin film transistors (TFTs) due to their high optical transmissivity and high conductivity. Furthermore, ZnO is also widely used in acoustic wave devices because of their larger electromechanical coupling in ZnO (Ü. Özgür et al, 2010). On the other hand, bismuth (III) oxide, Bi_2O_3 is another binary glass material which is composed of bismuth and oxygen atoms. Bi_2O_3 is an attractive Binary glass system among engineers and researchers because of its semiconducting behaviour, high energy bandgap, and high refractive index. Hence, Bi_2O_3 is widely used in humidity sensing and optoelectronics devices (Condurache-Bota S ,2018). Besides that, tellurium trioxide, TeO_3 is a chemical compound composed of tellurium and oxygen atoms. The TeO_3 has shown some properties where it can behave as a semiconductor and as an insulator when altered with specific compositions (Rada S et al, 2009). However, the TeO_3 does not have practical applications but understanding the behaviour of tellurium compounds can benefit the development of material science and new technologies. In addition, Boron trioxide, B2O3 is another binary glass system that consists of boron and oxygen atoms. The B2O3 is widely used in the manufacture of electronic components such as semiconductors and capacitors as well as in the production of boron-doped silicon for use in solar cells and other electronic devices.

2.2 Properties of Binary Glass Systems

2.2.1 Physical Properties

The physical property of a material is the characteristic of a substance that is observed or measured without altering the identity of the substance. Besides that, physical properties are features that scientists and engineers can calculate and measure without altering the composition of the substance or material that is under study (Chemistry LibreTexts, 2016). There are two important physical properties of binary glass systems which are the density and molar volume. The density of a substance is defined as its mass per unit volume. Hence, its essentially a measurement of how tightly matter is packed together (BYJU's, 2024). Furthermore, density plays a crucial role in determining the mechanical, thermal, optical, and chemical properties of binary glass compositions which impacts their suitability for various applications. Density is calculated by dividing the mass by the volume of the substance. The equation 2.1 below shows the formula to compute density, where ρ is density, *m is* mass and *v* is volume. The density is usually expressed in grams per cubic centimetre, g/cm^3.

$$\rho = \frac{m}{v} \tag{2.1}$$

where,

v = Volume

ρ = Density

m = Mass

On the other hand, molar volume is another prominent physical property that measures the volume occupied by one mole of a chemical element or a chemical compound. Similar to density, molar volume plays a significant role in the characterization, processing, and optimization of binary glass systems thus influencing their structural, mechanical, thermal, and optical properties. Molar volume is calculated by dividing the molar mass by the density of the substance. The equation 2.2 below shows the formula to compute the molar volume. Where, V_m is the molar volume, M is the molar mass, and ρ is the density. Molar volume is usually expressed in cubic centimetre per mol, cm^3/mol.

$$V_m = \frac{M}{\rho} \tag{2.2}$$

where,

V_m = Molar Volume.

M = Bulk Modulus

ρ = Density

2.2.2 Elastic Properties

Elastic properties often denoted as elastic modulus is another prominent property that determines the deformation of a substance or material under pressure or flexure (Kimbell and Azad, 2021). Elastic modulus consists of several other properties. The first property is the longitudinal velocity. Longitudinal velocity also known as primary wave velocity is the speed at which a longitudinal wave travels through a material.

The Longitudinal velocity is calculated by finding the root of the sum of the bulk modulus and shear velocity and dividing it by the density of the material. The equation 2.3 below shows the formula to compute the ultrasonic velocity. However, for binary glass systems, the computation of ultrasonic velocity is a little complex due to the forming of two glasses. The ultrasonic velocity for binary glass systems is computed by taking the average of both glasses. The longitudinal velocity is usually expressed as $VL(\frac{m}{s})$.

$$V_p = \sqrt{\frac{K + \frac{4}{3}u}{\rho}} \tag{2.3}$$

where,

V_p = Longitudinal Velocity.

K = Bulk Modulus

u = Shear Modulus

ρ = Density

The second property of elastic modulus is the shear velocity. Shear velocity is also called friction velocity is the speed at which a shear wave propagates through a material. In binary glass systems, shear velocity is more complex to calculate due to the presence of two different glass-forming materials. Thus, the weighted average of shear velocities of individual glass components is computed. Besides that, another important elastic property is the young modulus. Young modulus is a parameter that characterizes the behaviour of an elastic material depending on the direction in which a force is applied (Anon, 2020). The young modulus is usually determined by various testing methods and experiments to get the stress applied to the material, σ and the strain of the material, ε.

Then, the young modulus, *E* is calculated by dividing the stress applied to the material by the strain of the material. The equation 2.4 below shows the formula to compute the young modulus. The young modulus is generally expressed as pascals.

$$E = \frac{\sigma}{\varepsilon} \tag{2.4}$$

where,

$E =$ Young Modulus.

$\sigma =$ stress applied to the material.

$\varepsilon =$ strain of the material

Furthermore, the following elastic property is the shear modulus. Shear modulus also known as the modulus of rigidity is the measure of the rigidity of the body by computing the ratio of shear stress to shear strain. Shear modulus is often denoted as G and expressed as Pascals, (Pa) (BYJUS, n.d.). On the other hand, bulk modulus is another key elastic property. The bulk modulus is defined as the fraction of volumetric stress related to the volumetric strain of a material during the deformation of the material. Bulk modulus is commonly denoted with the symbol K and expressed in the units per square inch (N/m^2). In addition, Microhardness is the measurement of a material's hardness or resistance to deformation. Microhardness is usually measured by multiple testing methods depending on the material subjected to study. Microhardness is commonly denoted with the symbol H and is usually expressed as pascals, (Pa). In addition, fractal bond connectivity is another elastic property that is used to study how the arrangements of bonds influence a material's behaviour and conditions under various conditions. Thus, the fractal bond connectivity helps engineers and researchers to understand the mechanical, thermal, and optical properties of glass structures. Finally, Poisson's ratio is another elastic property which is the inverse of the ratio of the transverse strain to lateral or axial strain. Thus, understanding the Poisson's ratio is fundamental in predicting the glass's mechanical properties and its stiffness and flexibility to deformation. Poisson's ratio is denoted with the symbol σ.

2.2.3 Optical Properties

Optical Properties refer to a material's behaviour when electromagnetic radiation is in contact with the material's surface. Thus, in layman's terms, optical properties are how a material interacts with light (Griscom,1991.). Optical properties play an important role in binary glass systems for their utilization in a wide range of applications spanning from optics, photonics, and many more. One of the prominent optical properties of binary glass systems is optical bandgap. The optical bandgap is defined as the threshold for a photon to be absorbed by a material. The optical bandgap plays a significant role in determining the material's optical properties mainly about its absorption characteristics. Thus, in binary glass systems, the optical bandgap can be specifically engineered to suit the required application of the binary glass system.

2.3 Deep Learning

Deep learning is a part of machine learning that utilizes multi-layered neural networks to replicate and simulate the sophisticated decision-making ability of the human brain. Furthermore, deep learning models are trained on large quantities of data to identify and classify phenomena, recognize patterns and relationships, evaluate possibilities, and carry out predictions and decisions. Deep learning models try to imitate the mechanism of the human brain through a combination of data inputs, weights, and biases.

Furthermore, deep learning models compromise multiple layers of interconnected nodes where each node constructs them upon the previous layer to rectify and optimize the prediction or classification which is a process called forward propagation. On the other hand, backward propagation is another process in deep learning where algorithms such as gradient descent is used to compute errors in predictions and then alter the weights and biases of the function by reversing through the layers to train the model.

Hence, the forward and backward propagation works together to ensure that the deep learning model makes predictions and make corrections for any errors (IBM, 2023). There are various neural network models used in deep learning such as artificial neural networks (ANN), convolution neural networks (CNN) , and recurrent neural networks (RNN).

2.3.1 Artificial Neural Networks.

Artificial Neural Networks are heavily inspired to imitate the function of the human brain. An ANN consists of large numbers of simple processors linked together by weighted connections. Thus, by analogy, the processing nodes can be assumed as neurons. Hence, each node output depends only on the information that is locally available at the node either stored internally or being received by the weighted connections. Furthermore, each unit receives inputs from many other nodes and transmits its output to other nodes. On its own, a single processing element is not very powerful as it generates a scalar output with a single numerical value which is a simple non-linear function of its inputs. An error is generated from the difference between the desired response and the system output. This error information is fed back to the system, and it adjusts the system parameters in a systematic fashion. This process is repeated until the performance is acceptable. Therefore, it is clear that from the description, the performances heavily depend on the data provided. Hence, if one does not have the data to cover a significant portion of the operating conditions then the neural network technology is not the right solution.

On the contrary, if there is a lot of data and the problem is poorly understood to decode an approximated model then the neural network technology is the right choice for the specific problem. Besides that, the ANN doesn't solve the problem with the means of heavy mathematics, it displays data processing characteristics that give an approximate solution to a given problem. Furthermore, the components included in the ANN are similar to the previous neural networks where it has weights, an adder that sums up all the inputs altered by their respective weights, and finally an activation function. The ANN makes use of the backpropagation algorithm. This backpropagation algorithm is used in layered feed-forward ANNs which means that the ANNs are organized layers that send their signals forward while the errors are propagated backward. The ANN network is fed inputs in the input layer and the output of the layer is fed by the neurons on an output layer. The backpropagation algorithm implements supervised learning where an algorithm is provided with examples of the input and outputs that network functions to compute and the error which is the difference between the actual and expected results is computed. Hence, the main idea of the backpropagation algorithms is to minimize this error until the ANN learns the data (Dongare et al, 2012). The figure 2.1 below shows the model of an ANN.

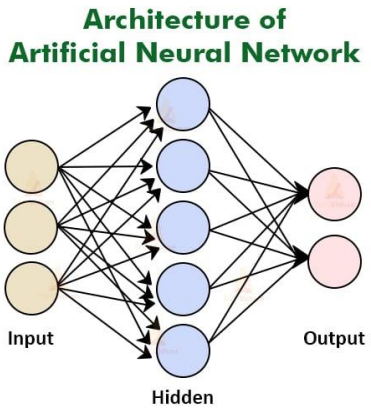


Figure 2.1 : Architecture of ANN model

(Team, 2020)

2.3.2 Convolution Neural Network

The CNN approach has been one of the most used and represented neural networks in the field of deep learning. The CNN is a type of feedforward neural network that can extract features from data with convolution structures. Thus, compared to the traditional feature extraction methods CNN does not extract features manually but it utilizes visual perception. The CNN also consists of the components of the previous neural networks discussed such as the weights, adders, outputs, and so on. The CNN also consists of loss functions and optimizers which are developed for the CNN system to learn what to expect. The CNN has many advantages such as the local connections where each neuron is not connected to all neurons of the previous layers but only to a minimum number of neurons which is efficient in minimizing parameters and speed convergence. The second advantage is weight sharing, where a group of connections can share weights which also minimizes parameters further. The third feature is the down-sampling dimension reduction which reduces the number of data while keeping beneficial data. Thus, these three characteristics make the CNN one of the most represented algorithms in the deep learning field (Li et al, 2021).

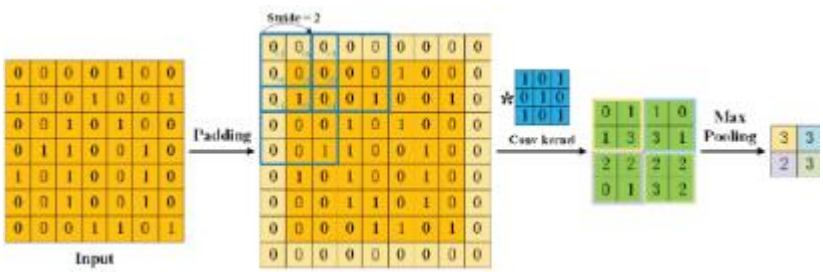


Figure 2.2 : Procedure of CNN

(Thube, 2020)

2.3.3 Recurrent Neural Networks

The Recurrent Neural Network is designed to save the output of a layer, thus the recurrent neural network will be fed back to the input to assist in predicting the outcome of the layer. The first layer is a feed-forward neural network followed by a recurrent neural network layer where some data in the previous time-step is kept by a memory function. The forward propagation is integrated in this scenario which stores data required for its future use. Hence, if the prediction is false, the learning rate is employed to conduct minimal changes. Thus, finally maximizing it towards making predictions during backward propagation. The figure 2.3 below shows the model of the recurrent neural network.

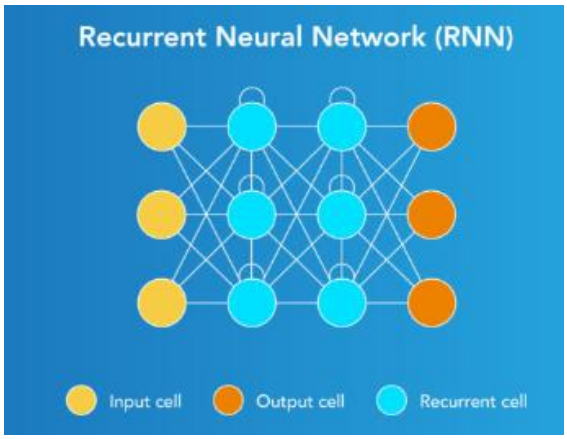


Figure 2.3 : Reccurent Neural Networks

(Great Learning Team, 2020)

2.3.4 Comparisons of advantages and disadvantages of different neural networks

Table 2.1: Table of comparisons between Neural Networks

Types of Neural Networks	Advantages	Disadvantages
Artificial Neural Network (ANN)	- Flexible - Parallel Processing - Adaptability	- Complexity - Sensitive to training data
Convolution Neural Network (CNN)	- Few parameters compared to fully connected layer	- Complexity - Slow to design
Recurrent Neural Networks (RNN)	- Uses convolution layers to extend the pixel effectiveness.	- Gradient vanishing and exploding problems. - Difficult to train. - Difficult to process
Modular Neural Network (MNN)	- Efficient - Independent training - Robustness	- Moving target problems

2.3.5 Selection of neural network for project

The Artificial Neural Network, ANN was selected over the other neural networks to predict the elastic and optical properties of the binary glass system because of several reasons. When compared to Convolution Neural Network, (CNN), the features of the CNN somehow seem superior to the ANN. However, CNN is more suited for applications that deal with image data such as image classification and computer vision. In addition, the input data which is usually fed to the CNN is usually in pixels. Thus, the CNN is not the right neural network which is required for this project as this project's input is mainly dependent on datasheets and compositions. Similarly, the Recurrent Neural Network is not suitable for this project as this neural network is more suited for text processing, image processing, and translation techniques. Besides that, when compared to a Modular Network, too many submodules or subtasks is not necessary for this project as the main goal of the project is more focused to the prediction of parameters of compositions rather than high-level arithmetic calculations.

The ANN is more suited for this project also because of its prediction quality. Furthermore, the network is capable of predicting parameters by the experimental system. On the other hand, the network has a parallel structure and very fast learning capacity. Besides that, any experimental data can be used as training and testing data for an artificial neural network. Hence, the ANN has much more superior performance to assist the targeted results of the system and is responsible to analyse any given parameters in practical applications.

2.3.6 Activation Functions in Neural Networks

Activation functions are widely used in neural networks to transform an input signal to an output signal which then is propagated as input to the next layer of the neural network. In an artificial neural network, the sum of the products of inputs and their weights is then input to an activation function to get an output of the particular layer and it is then fed to the subsequent layer. One of the most used activation functions are non-linear activation function. The concept of a neural network model is similar to a linear regression model where the predicted output is the same as the input supplied given that the activation function is not completely defined. Thus, a linear activation function boundary is linear and they are applied to the network which is only able to adjust to only the linear manipulations of the input. However, in real-life scenarios, the errors present have non-linear attributes that in turn with the neural networks' capability to learn about the inaccurate data. Hence, due to these the non-linear activation functions are favoured over linear activation functions (Siddarth Sharma et al, 2020).

There are multiple types of activation functions depending on the complexity of the deep learning model. One of the types of activation functions is called the binary step function. The binary step function is the simplest activation function and can be applied with a simple if-else statement in python. Binary activation functions are commonly used in binary classifiers however it cannot be implemented in cases such as multiclass classification. Furthermore, the gradient of the binary step function is equal to zero which results in obstacles in the backpropagation step. The mathematical binary step can be defined as shown in the equation 2.5 and 2.6 below. The figure 2.4 below shows the graph of a binary step function.

$$f(x) = 1, x >= 0 \tag{2.5}$$

$$f(x) = 0, x < 0 \tag{2.6}$$

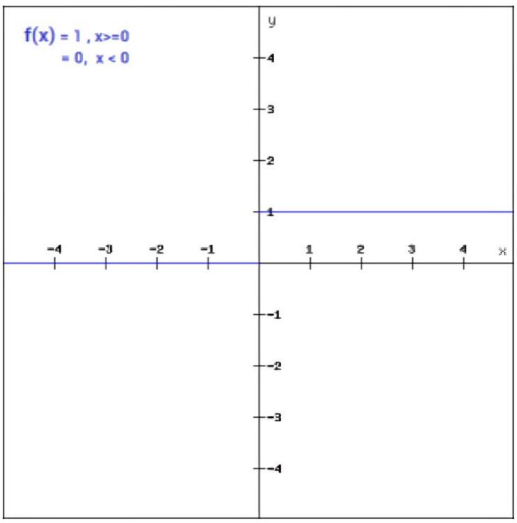


Figure 2.4 : Binary Step Function

The other type of activation used is called the sigmoid activation function. This sigmoid is one of the most well-known activation functions due to its non-linearity. The sigmoid activation function transforms the values in the range from 0 to 1. The sigmoid function can be expressed mathematically as shown in equation 2.7 below. The figure 2.5 below shows the plot of a sigmoid function.

$$f(x) = 1/e^{-x} \tag{2.7}$$

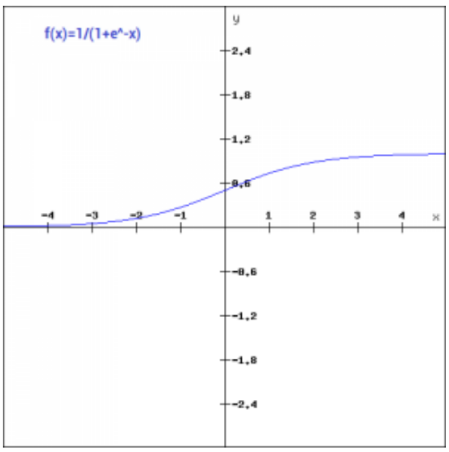


Figure 2.5 : Sigmoid Function

On the other hand, the Tanh function is another prominent activation function. The tanh function is identical to the sigmoid function, but it is aligned about the origin. Hence, the outputs from the previous layers will have different signs which will be fed as input to the next layer. The equation 2.8 shows the mathematical expression for the Tanh function.

$$f(x) = 2sigmoid(2x) - 1 \tag{2.8}$$

The Tanh function is continuous and differentiable, hence the values fall in the range of -1 to 1. The gradient of the Tanh is steeper compared to the sigmoid function. Thus, the Tanh is more preferred over the sigmoid function because its gradients are not confined to change within a particular direction and is zero cantered. The figure 2.6 below shows the plot of the Tanh function.

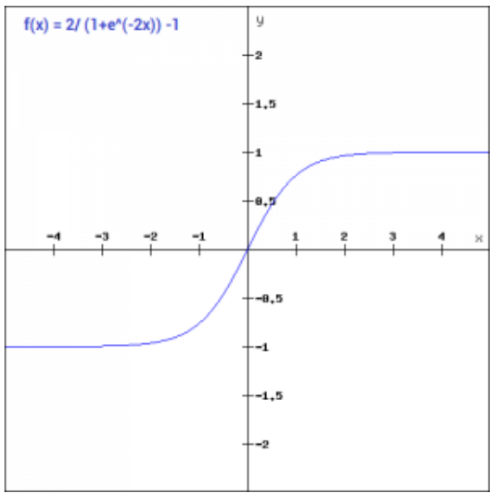


Figure 2.6: Tanh Function

The other well-known activation is the ReLU activation function. ReLU is a short form for rectified linear unit and it's a non-linear activation function. One of the benefits of The ReLU function is that all the neurons are not activated at the same time. This indicates that a neuron will be deactivated if the output of linear transformation is zero. The ReLU activation function is expressed mathematically as shown in the equation 2.9 below. The figure 2.8 below shows the plot for the ReLU activation function.

$$f(x) = \max(0, x) \tag{2.9}$$

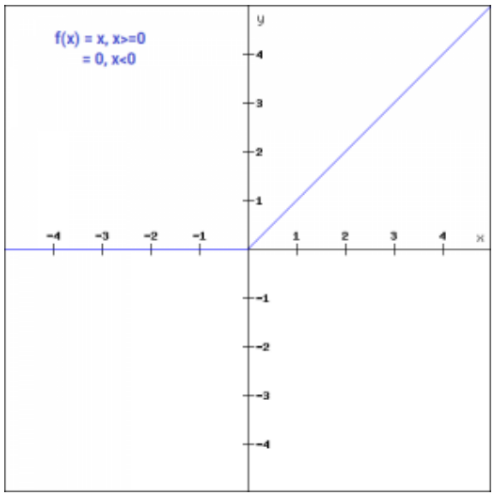


Figure 2.7 : ReLU function

2.4 Programming languages for deep learning

In this section several programming languages were studied and compared to know their advantages and drawbacks.

2.4.1 MATLAB

MATLAB is not only a programming language but a five-part system that consists of a language, development environment, graphics visualizer, math library, and an interface for coding programs in other languages. However, MATLAB is specialized in performing matrix computation. MATLAB is robust for performing mathematical operations and has built-in features for using machine learning models. Hence, MATLAB has an advantage over other programming languages such as python which require add-on toolkits and frameworks for mathematical functions and for simulating deep learning models. Furthermore, MATLAB simulates at a higher speed which has a slight edge over other programming languages. On the other hand, there are several disadvantages using MATLAB. One of the drawbacks of using MATLAB is its cost where a fee is needed to be paid to access the system and for additional functions. Besides that, MATLAB syntax is more challenging to learn compared to other programming languages (Houcque, 2005).

Figure 2.8: Logo of MATLAB

2.4.2 C++

C++ is one of the most well-known programming languages for general-purpose applications. It is the spine of operating systems like Windows, IOS and Linux, apps like Spotify, and Photoshop, and sites like Youtube and many more. C++ is a compile language, and it doesn't need an interpreter program which contributes to processing overhead. Furthermore, programs written in C++ are fast and efficient.

One of the disadvantages of C++ is the complexity of writing the code. Furthermore, programs in C++ take time to debug and take a longer time to alter hyperparameters compared to other programming languages (Kaijanaho, 2000).

Figure 2.9 : Logo of C++

2.4.3 Python

Python is the most popular general-purpose programming language that is easy to learn compared to other programming languages. It is also considered the best all-rounded programming language for AI. When compared to other programming languages, python has an easier syntax, words symbols, and expressions (Banerjee et al, 2022). Hence, there will be more time to focus on other aspects such as data to tune the models. Besides that, there is an extensive number of frameworks and libraries that are available in python that make it adaptable to various machine learning or data science tasks. Some of the libraries are NumPy, Scikit-learn, pandas Tensor flow, Keras, Pytorch which provide powerful abilities for data analysis, machine learning, deep learning and allow developers to concentrate on solving sophisticated tasks.

Figure 2.10: Logo of Python

2.4.4 Selection of programming language for project.

After reviewing several programming languages, Python was chosen as the programming language to be used for this project. One of the reasons is because of its easy-to-read syntax. Python has a very straightforward syntax making it easier to read and write the codes. Furthermore, the object-oriented programming structure of python provides the users with a logical approach to organizing the structure of the code to be clean and neat for complex projects. Furthermore, python's debugging speed is faster making it simpler to identify and rectify errors made in the code. Hence, python is extremely user-friendly which enables the user to adapt faster to the python's programming environment. Besides that, the factor that distinguishes python for deep learning compared to other programming languages is its extensive libraries and frameworks. Python provides a wide range of libraries specifically designed for machine learning simplifying user's job to develop deep-learning neural networks languages.

2.5 Techniques of using ANN for prediction

In this section, several journals regarding on how to utilize the ANN model for prediction of data will be reviewed and studied.

2.5.1 Price Prediction of Share Marketing

Based on the paper by Khan et al. (2011), researchers have used artificial neural networks to predict the share market price based on share price data. This approach is very similar to the prediction of elastic and optical properties of binary glass systems where a model is trained based on data which are numbers to predict the desired output. Similar to the elastic and optical properties of binary glasses, there is no fixed formula or rule to estimate the price of the share in the share market. The share market holds a place of high interest among investors as it allows them to receive

financial incentives by investing their resources in shares and secondaries of multiple companies. However, the share market is considered a chaos system due to its high level of uncertainty and unpredictability. Thus, the researchers have decided to use artificial neural networks, ANN to predict the outcome of the share market as ANN has the potential to distinguish unknown and hidden patterns in data which can be extremely efficient for share market prediction.

The researchers have conducted a comprehensive fundamental analysis of a company to analyse its product sales, manpower, quality, infrastructure, and other factors to get a better understanding of the company's standing in the market. Based on the analysis, it is believed that the market is determined 90% by logical and 10% by physiological factors. However, the researchers have concluded that this fundamental analysis is not suitable to be fed as data for the ANN as the data by the analysis is used to find out the intrinsic value of an asset that does not change daily hence which is not appropriate for short-term basis.

On the other hand, price charts are used to identify trends. These trends are believed to possess a cyclical or noticeable pattern. These parameters are coined as indicators and oscillators and it's a well-known method to predict the market. However, the researchers have decided not to use this approach because of its subjective nature. In this paper, the researchers have designed the ANN such that there is a training phase where there are weights, and a backpropagation algorithm is implemented for this training phase where it will calculate the error between the outputs and the actual targets. Hence, the researchers have used a feedforward network approach for this system. There are a lot of inputs in the share market which influence the share price but not all inputs are utilized in the system because their influence is not crucial in the share market price. Thus, 5 inputs were finalized and used for the ANN system which is General Index (GI), P/E ratio, Net Asset Value (NAV), Earnings per share (EPS), and volume. Following this, the data was standardized according to the network, and the inputs were fed to the network. In addition, the ANN model has been designed with an input layer with 5 neurons, one hidden layer which has 5 neurons, and an output layer with a single neuron. Hence, the backpropagation has been used for training the network. There are two phases in this ANN model which is the training phase and followed by the prediction phase. In the training phase, the input data has

been standardized to be fed to the network into the input nodes. The figure 2.11 below shows the architecture of the ANN model for the training phase.

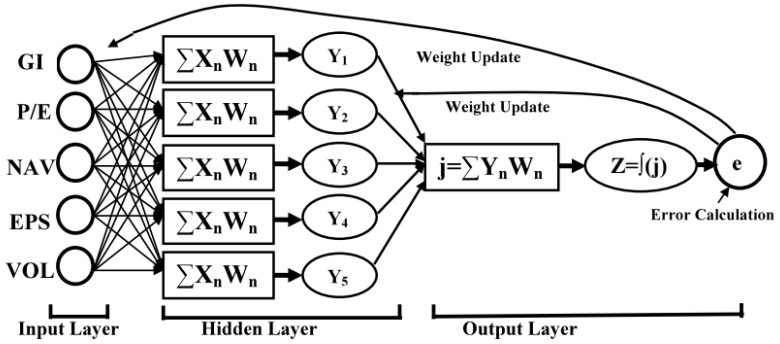


Figure 2.11 : Architecture of ANN model for training phase

From the figure above, it can be observed that the standardized input data is sent into the input layer, and the product of the weights and the input data are then fed to the neurons in the hidden layer. Following that, each neuron goes through the output to the subsequent neuron of the output layer. The error which is produced by the propagation phase is used to update the weight to produce a better output. The two phases of the ANN model are repeated until the sum of the square error is near zero. The figure 2.12 and 2.13 shows the results produced by the ANN model for the prediction of market share prediction of market share price throughout November 2010.

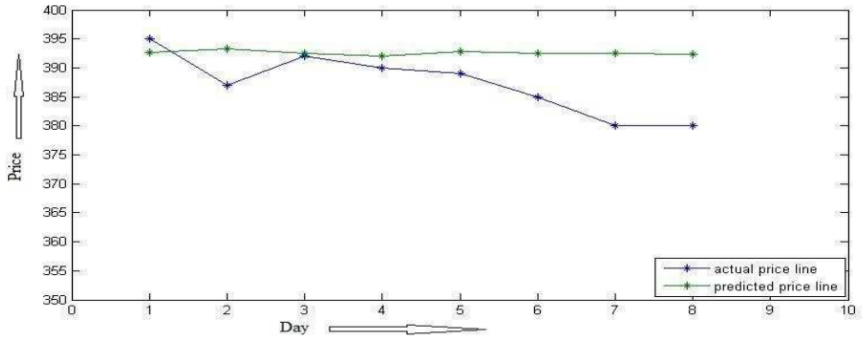


Figure 2.12 : Graph actual price line against predicted price line.

DATE	Predicted Price (TK)	Actual Price (TK)	Error (%)
01-NOV-2010	392.7	395	0.58
02-NOV-2010	393.3	387	1.62
03-NOV-2010	392.5	392	0.12
04-NOV-2010	392.0	390	0.51
08-NOV-2010	392.8	389	0.97
09-NOV-2010	392.5	385	1.94
10-NOV-2010	392.5	380	3.28
11-NOV-2010	392.3	380	3.23

Figure 2.13: ANN model for prediction of market share price throughout November 2010

Based on the simulation results shown above, the average error that was computed was 1.53%. Furthermore, the researchers have inferred that with more input data the training's phase efficiency is better and a more accurate result will be produced. Hence, with more data available for predicting financial markets, the higher the potential for an accurate prediction.

2.5.2 Ultrasonic Behaviour in Tellurite Glasses

Based on the paper by Effendy et al. (2020), researchers have used an ANN model to simulate the elastic properties of the binary series ZnO-TeO2 glasses using MATLAB software. The researchers have conducted various experiments to get the real values of the ZnO-TeO_2 glasses parameters with different compositions. Some of the parameters that were obtained by the experiment were molecular weight, density, longitudinal modulus, shear modulus, bulk modulus, and young modulus. Hence, these are the parameters that will be predicted by the ANN model.

In this paper, the researchers have decided to use a multilayer perceptron (MLP) feed-forward neural network to predict the properties of the ZnO-TeO_2 binary glass series. The MLP model was designed with one input layer, two hidden layers, and one output layer. Furthermore, the mean square error (MSE) was implemented to predict the accuracy of the output data. The training process starts with random input data and

several iterations of the training is conducted. After sufficient data is acquired the predicted final output values were determined. Additionally, the ANN performance was measured by computing the percentage error (PE). The figure 2.14 and 2.15 shows the values predicted by the ANN model.

Glass Sample	Elastic Moduli (GPa)														
	L_{EXP}	L_{BC}	L_{ANN}	G_{EXP}	G_{BC}	G_{MM}	G_{ANN}	K_{EXP}	K_{BC}	K_{MM}	K_{ANN}	E_{EXP}	E_{BC}	E_{MM}	E_{ANN}
0 ZT	60.612	135.892	58.948	20.029	45.519	20.132	19.786	33.906	70.859	25.332	20.002	50.203	112.474	47.748	49.511
5 ZT	57.781	144.479	57.781	20.494	48.774	20.174	20.494	30.455	74.942	25.644	22.393	50.218	120.238	47.950	50.061
10 ZT	56.282	154.026	55.372	21.121	52.153	20.200	21.122	28.120	79.162	25.901	23.988	50.677	128.287	48.098	50.358
15 ZT	59.698	164.878	59.698	20.915	56.148	20.342	22.627	31.810	84.257	26.564	25.541	51.467	137.828	48.616	51.117
20 ZT	61.181	177.257	61.986	21.125	60.465	20.500	21.162	33.014	89.766	27.297	26.868	52.234	148.136	49.187	52.158
25 ZT	63.476	190.472	63.379	21.537	65.617	20.790	21.611	34.759	96.432	28.516	37.158	53.552	160.458	50.177	53.230
30 ZT	71.240	202.662	71.240	20.166	70.529	20.941	20.406	44.351	102.662	29.250	45.036	52.537	172.163	50.721	51.595

Figure 2.14: Parameters by the ANN model

Glass Sample	H_{EXP} (GPa)	H_{ANN} (GPa)	σ_{EXP}	σ_{BC}	σ_{MM}	σ_{ANN}
0 ZT	5.175	5.169	0.253	0.235	0.185	0.242
5 ZT	5.636	5.628	0.225	0.232	0.188	0.219
10 ZT	6.086	6.079	0.199	0.229	0.190	0.223
15 ZT	5.691	5.684	0.230	0.227	0.195	0.235
20 ZT	5.676	5.668	0.236	0.224	0.199	0.213
25 ZT	5.698	5.694	0.243	0.222	0.206	0.236
30 ZT	5.194	4.498	0.285	0.220	0.211	0.224

Figure 2.15: Parameters by the ANN model

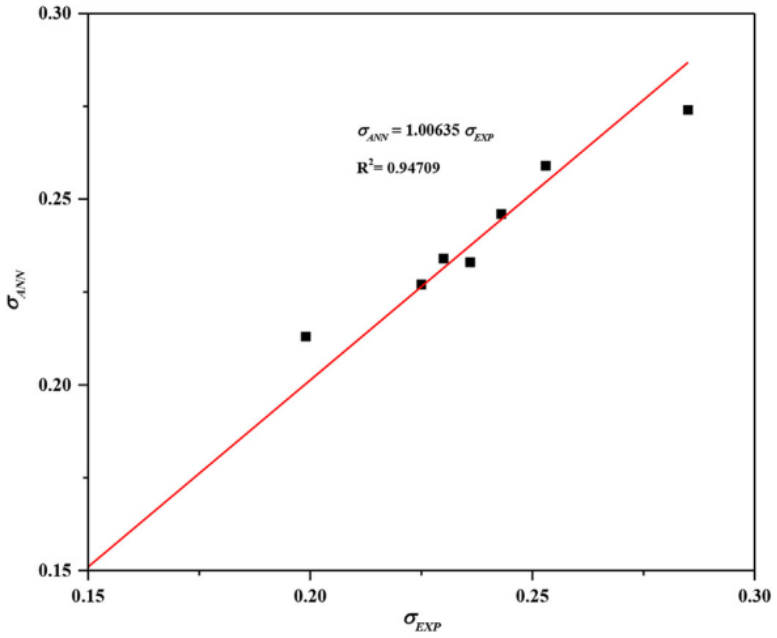


Figure 2.16 : R-squared value of Bulk Modulus parameter

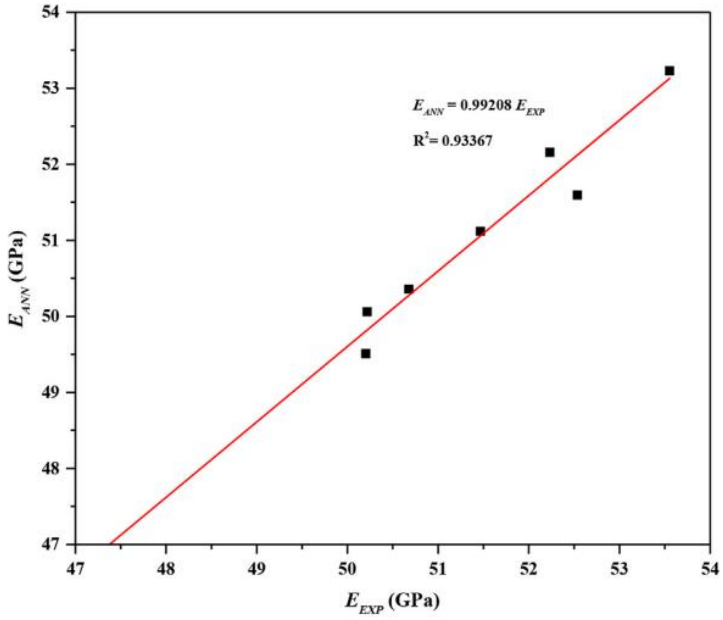


Figure 2.17: R-squared value of Poisson Ratio parameter

The plots above show the unity and goodness of fit *R2* value. The *R2* value indicates the proportion of the variation in the dependent variable that is predictable from the independent variable. Hence, if the *R2* value is nearer to one, it means that the model fits the data efficiently and can accurately predict the outcome of the predictor variables. Based on the plots above, the straight-line slope on all the plots indicates a great agreement between the results obtained from experimental and predicted values by the ANN model. Besides that, the goodness of fit, *R2* in all the plots is between 0.90361 and 09985 which is considered to be adequate. In addition, the difference in percentage error computed is less than 1% between the experimental and predicted values. Furthermore, the researchers have inferred that the upper hand of using an ANN model for predicting the elastic properties of binary glass systems is raw materials don't need to be melted hence saving time and cost of experiments. Thus, the results from this research indicate that the ANN model is viable to be used for future research.

CHAPTER 3

METHODOLOGY

3.1 System Overview

In this chapter, the method and justification for the hardware and software carried out are explained. Furthermore, the procedure to design and train the ANN model will be discussed and elaborated. Besides that, the project management for this project will also be discussed. The ANN model will be tested in various ways such as altering the activation functions and changing the number of epochs used for training. Hence, the best model that produces the most accurate results will be selected.

3.2 Hardware Overview

This project was performed on a laptop with Intel core i5 processor and a 16GB NVIDIA GeForce GTX1651 GPU. The hardware specifications are specified in the table 3.1 below.

Table 3.1 : Hardware specifications

CPU	Intel core i5 – 10300H CPU @ 2.50GHz
GPU	NVIDIA GeForce GTX 1651
OS Version	Microsoft Windows 11 Home Single Language
System Type	64-bit operating system, x64-based PC
RAM	16.0 GB
Storage	453 GB

3.3 Software Overview

The proposed prediction of elastic and optical properties of binary glass system deep learning model was developed using Pycharm with Python 3.9.13 programming language. The following section will discuss the software and libraries installed for the project.

3.3.1 Pycharm

Pycharm is an integrated development environment (IDE) that is tailored mainly to Python programming. Pycharm is developed by JetBrains and is commonly used by Python developers for its powerful features and ease of use. Furthermore, Pycharm provides features such as code highlights, code completion and debugging capabilities and supports multiple deep learning frameworks.

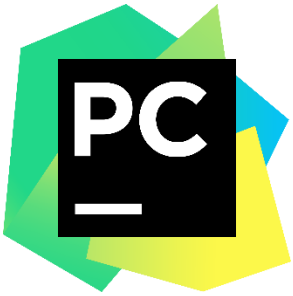


Figure 3.1: Logo of Python

3.3.2 Pytorch

Pytorch is an open-source machine learning framework that was developed by Facebook's Ai research lab. It is one of the popular deep learning frameworks used by developers for building and training deep learning models. Furthermore, the Pytorch framework has a unique computation graph which eases debugging and experimentation. The Pytorch was downloaded by Pycharm as the packages were already available hence no pip installation was required.

Figure 3.2: Logo of Pytorch

3.3.3 Numpy

Numpy is a powerful numerical computing library for Python It provides support for huge, multi-dimensional arrays and matrices with a vast collection of mathematical functions to operate on these arrays efficiently. The numpy library is required in this project for handling and manipulating the data regarding the binary glasses and to ease the plot of the results.

Figure 3.3 : Logo of NumPy

3.3.4 Pandas

Pandas is a powerful library for data manipulation and analysis. The library provides data structure and functions for efficiently handling structured data, mostly in the form of tabular data such as CSV files, Excel spreadsheets, and SQL tables. Furthermore, the pandas library has a data frame which is a primary data structure that represents tabular data with rows and columns. Additionally, pandas provide a wide range of functions and methods for manipulating and transforming data which includes filtering, sorting, merging, joining, grouping, and reshaping data (McKinney, 2011). Pandas is required in this project as the data regarding the binary glass systems has been prepared in Excel format, hence the pandas library will be used to convert the data in the Excel file to a pandas data frame so that the data can be interpreted by the Pycharm IDE.

Figure 3.4: Logo of Pandas

3.3.5 Matlpotlib

Matplotlib is a comprehensive Python library for developing static, animated, and interactive visualizations. Matplotlib provides a flexible and powerful interface for producing a wide range of plots and charts including plots, scatter plots, bar charts, histograms, pie charts and more (Barrett, 2005). Matplotlib is required for this project to visualize the data produced by the ANN model and to conduct an analysis on the data.

Figure 3.5 : Logo of Matplotlib

3.3.6 Scikit-Learn

Scikit-learn commonly abbreviated as sklearn is a popular machine learning library in Python. This library provides a simple, and efficient toolset for data mining and data analysis tasks including classification, regression, clustering, dimensionality reduction, model selection and preprocessing. Sklearn provides tools for the evaluation of machine learning models using metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and more (Pedregosa, 2011). Furthermore, sklearn provides functions for cross-validation, hyperparameter tuning, and model selection. Sklearn library is required for this project to calculate the mean squared error of the ANN model to evaluate its accuracy and efficiency in predicting the data.

Figure 3.6 : Logo of Scikit-Learn

3.3.7 Tkinter

Tkinter is a library in Python used for creating graphical user interfaces (GUIs). Tkinter provides a simple and intuitive way to build desktop applications with graphical elements such as windows, buttons, labels, textboxes, menus, and more. Tkinter has multiple built-in widgets such as buttons, labels, entry fields, textboxes, checkboxes, list boxes, and many more to create complex user interfaces (Beniz and Espindola, 2016). Thus, Tkinter is needed for this project to visualize the data better and to be integrated with the Matplotlib library to visualize the graphs plotted easily.

Figure 3.7 : Logo Of Tkinter.

3.4 Data Preparation

The dataset that is going to be used for this project has been prepared by my supervisor Dr Nuraidayani Binti Affendy based on her PhD research which she has conducted before. The parameters of the binary glass were achieved after multiple experiments on the binary glasses. The table 3.2 below shows the dataset that was provided. Thus, this data will be used to train and tune the ANN model to predict the elastic and optical parameters of the binary glass system.

Table 3.2: Data set to train the ANN model

ZnO	Bi2O3	TeO2	B2O3	Density, ρ (	Molar Volu	Longitudina	Shear veloc	Longitudina	Shear mod	Bulk modul	Young mod	Microhardi	fractal bon	Poisson ra	Optical bar
0	0	100	0	4.939	32.314	3503	2013	60.612	20.029	33.906	50.203	3.295	2.362	0.253	2.634
5	0	95	0	4.967	31.345	3410	2031	57.781	20.494	30.455	50.218	3.754	2.691	0.225	2.616
15	0	85	0	5.049	29.286	3438	2035	59.698	21.121	31.81	51.467	3.76	2.63	0.23	2.588
20	0	80	0	5.114	28.149	3458	2032	61.181	21.125	33.014	52.234	3.713	2.559	0.236	2.582
25	0	75	0	5.222	26.818	3486	2030	63.476	21.137	34.759	53.252	3.686	2.478	0.243	2.574
30	0	70	0	5.283	25.768	3672	1953	71.24	20.866	44.351	52.737	2.654	1.818	0.302	2.557
0	5	95	0	5.372	32.561	3217	1715	55.629	15.813	34.544	41.16	2.093	1.831	0.301	2.343
0	7	93	0	5.491	32.971	3159	1699	54.8	15.858	33.655	41.117	2.152	1.884	0.296	2.308
0	10	90	0	5.659	33.617	3172	1618	56.942	14.826	37.173	39.26	1.739	1.595	0.323	2.284
0	15	85	0	6.052	33.965	3185	1547	61.394	14.489	42.076	38.991	1.491	1.377	0.345	2.21
0	0	0	100	1.84	37.817	3496	1948	22.511	6.986	13.196	17.815	1.048	2.118	0.275	2.791
45	0	0	55	3.241	23.114	5114	2619	84.777	22.24	55.124	58.81	2.636	1.614	0.322	2.725
50	0	0	50	3.267	23.11	5320	2688	92.471	23.615	60.984	62.747	2.7	1.549	0.329	2.721
60	0	0	40	3.555	21.569	4850	2366	83.654	19.904	57.116	53.497	2.071	1.394	0.344	2.608
0	40	0	60	5.462	41.772	4225	2257	97.525	27.833	60.415	72.382	3.705	1.842	0.3	2.728
0	45	0	55	5.736	43.231	3988	2162	91.245	26.827	55.475	69.309	3.724	1.934	0.291	2.721
0	55	0	45	6.259	45.951	3528	2011	77.909	25.336	44.128	63.798	4.069	2.296	0.259	2.602
0	60	0	40	6.55	46.935	3260	1920	69.625	24.154	37.419	59.632	4.276	2.582	0.234	2.546

3.5 Architecture of the ANN model

The figure 3.8 below shows the architecture of the ANN model. The ANN was modelled by referring to the compositions of the binary glasses and the parameters to be predicted. From the dataset there are four compositions of binary glasses which are Zinc Oxide (ZnO), Bismuth (III) oxide (Bi_2O_3), Tellurium dioxide (TeO_2) and Boric Oxide (B_2O_3), hence these four compositions will be fed as input to the neural network. Furthermore, the subsequent layer is the hidden layer. The number of hidden layers is subject to change depending on the accuracy of the prediction. Thus, if the predicted results are satisfactory then the number of hidden layers will be further increased. The output layer predicts the parameters which are the elastic, optical, and physical properties of the binary glass system. Based on the data provided, there are 12 parameters to be predicted by the model which are the molar volume, longitudinal velocity, shear velocity, longitudinal modulus, shear modulus, bulk modulus, young modulus, microhardness, fractal bond connectivity, Poisson ratio, and optical bandgap.

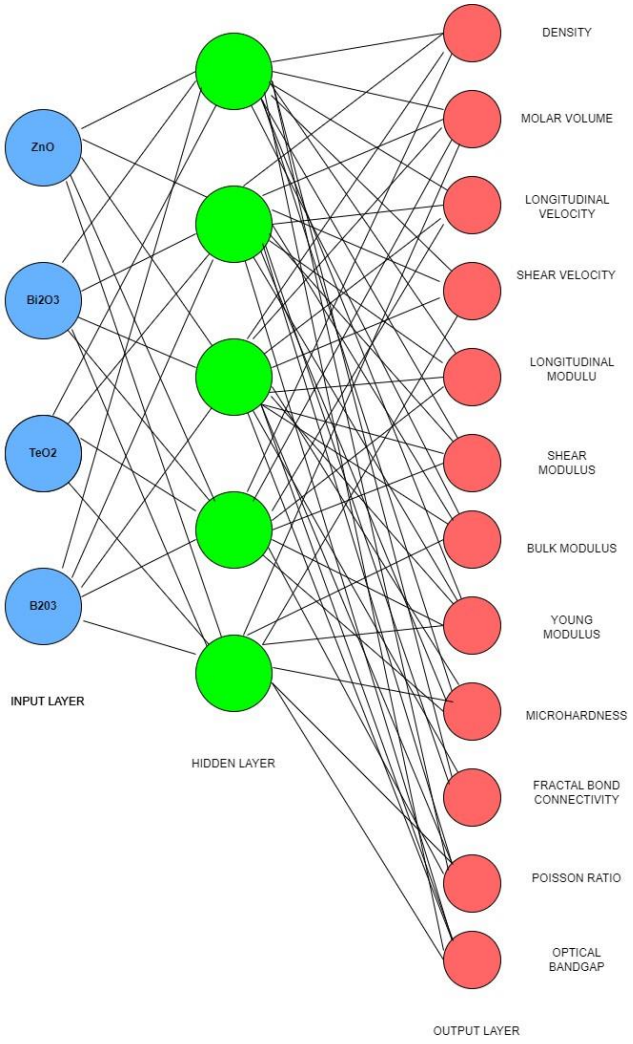


Figure 3.8 : Architecture of the ANN model.

3.6 ANN model training procedure

In this section, the training procedures for this proposed project will be discussed and explained.

3.6.1 Data Conversion

The data provided in the Excel file will have to be exported to the Pycharm IDE. The format of the data in the excel file is in float form. Since the Pytorch is the deep learning framework that is being implemented the data in float format needs to be converted to tensor format. The Pytorch tensor is a multidimensional array format used to represent scalars, vectors, matrices, and higher dimensional arrays. Tensors are extremely prominent as they are used to represent the data and parameters in the neural networks.

3.6.2 Configuration of input layer, hidden layers and output layers

As discussed, in section architecture of the ANN model, there is one input layer with 4 inputs and one layer with 12 outputs which is the predicted parameters of the binary glasses. Thus, the number of hidden layers and neurons in the layer will be increased or decreased according by comparing them to previous trials of the model. Besides that, the activation is also subjected to change if the prediction results are satisfactory. Thus, other viable activation functions such as the ReLU and the Leaky ReLU will be tested if the previous activation functions do not produce good results.

3.6.3 Configuration of learning rate, number of epochs and optimizers

Learning rate is an important hyperparameter in deep learning models as it determines the step size in which the model parameters are updated during optimization. Furthermore, setting an appropriate learning rate is important is essential for achieving optimal convergence and performance during training. Besides that, a suitable learning rate ensures that the optimization process converges to a minimum of the loss function effectively. Thus, if the learning rate is too small then the training process may be slow, and the model may be trapped in a local minimum. Conversely, if the learning rate is too large the optimization process may oscillate or diverge. Thus, the learning rate will be increased or decreased depending on the accuracy of the prediction.

Epoch is another important parameter in neural networks which depicts the complete pass through the whole training dataset. During each epoch, the models go through the training set once and update its weights and biases based on the training data to reduce the loss function. Furthermore, the number of epochs is used to control the number of times the entire training dataset is presented to the model during training. Additionally, increasing the number of epochs can benefit the model in learning more complex patterns in the data but could be risky as it might cause overfitting. Conversely, a low number of epochs may result in underfitting. Hence, choosing the appropriate number of epochs is prominent for developing efficient deep-learning models.

On the other hand, optimizers are another important parameter in deep learning models. In deep learning neural networks, an optimizer is an algorithm used to update the weights and biases during training to reduce the loss function. The optimizer adjusts the model's parameters based on the gradients of the loss function concerning those parameters. There are several types of optimizers commonly used in deep learning models such as stochastic gradient descent (SGD), Adam optimizer, Adagrad optimizer, and many more. The suitable optimizer will be chosen based on the accuracy of the prediction of the deep learning model.

3.7 Evaluation of ANN performance

The deep learning model will be evaluated based on several factors. One of them is the Mean Squared Error (MSE) loss. The MSE is a commonly used loss function in neural networks for regression tasks. This function computes the average squared difference between the predicted values and the real target values. The formula to calculate the MSE loss is shown in the equation 3.1 below. In the deep learning model, the MSE loss is also known as training loss.

$$MSE = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{3.1}$$

where :

n = the number of samples of the dataset.

y_i = the target value.

$\hat{y_i}$ = the predicted value

The MSE loss function is usually used in regression tasks where the objective is to predict continuous numerical values which is suitable for this project. Hence, during the training process, the ANN model's goal is to reduce the MSE loss by altering its weights and bias through the optimizer function. Thus, the gradients of the MSE loss concerning to the model parameters are calculated during the backpropagation and the parameters are updated in the direction that minimizes the loss (Terven et al, 2023). Additionally, one of the advantages of the Pytorch framework is that the MSE function is already built-in with the command torch.nn.MSELoss.

On the other hand, the percentage error will be computed between the real and predicted value to know the efficiency of the prediction. A smaller percent error means that the system is nearing an accepted value and if we get a larger percentage error it indicates that the system is a long way from the real value. The percentage error is calculated by computing the difference between the predicted value and the actual value and is multiplied by 100 and is usually expressed in percentage (BYJU'S ,n.d). The formula in the figure below shows the formula to compute the percentage error. Since there is 18 sets of training data sets used, the percentage error of each of the 18 predicted values will be computed . After each of the predicted parameters are calculated the mean absolute percentage error will be calculated of the 18 sets for a parameter.

The mean absolute percentage error (MAPE) is one of the most prominent metrics of model prediction accuracy which calculates the average magnitude of error produced by a model on how distant off predictions are on an average. Furthermore, the MAPE of each model will be calculated by summing the MAPE of each parameter predicted and dividing it by the total number of parameters which is 12. The MAPE of each model will be compared and the model with the lowest percentage error will be selected (Kim et al, 2016). The figure 3.9 below shows the interpretation of typical MAPE values. Hence, the MAPE will be analysed and judged based on the interpretation as shown in the figure below. The equation 3.2 and 3.3 show the formula to compute the percentage error and the MAPE.

$$Error = \frac{|predicted\ value - actual\ value|}{actual\ value} \times 100\ \% \tag{3.2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|predicted\ value - actual\ value|}{actual\ value} \times 100 \tag{3.3}$$

Where n = number of sets

Table 1
Interpretation of typical MAPE values

MAPE	Interpretation
<10	Highly accurate forecasting
10-20	Good forecasting
20-50	Reasonable forecasting
>50	Inaccurate forecasting

Source: Lewis (1982, p. 40)

Figure 3.9: Interpretation of typical MAPE values

Lastly, the R-squared, R^2 value also known as the goodness of fit will be used to evaluate the performance of the ANN model. The R^2 value is a statical measure that represents the proportion of the variance in the target variable that is described by the independent variables in a regression model. In addition, the R-squared value is a benchmark of correlation and indicates how strongly two variables can be related to each other. A correlation nearer to +1 indicates a strong relationship between the two variables while and correlation nearer to -1 indicates a tighter relationship in the opposite direction. On the other hand, a value closer to 0 indicates that there is not much relationship between the variables (Hussain, 2019).

3.7.1 Flowchart of ANN model training procedure

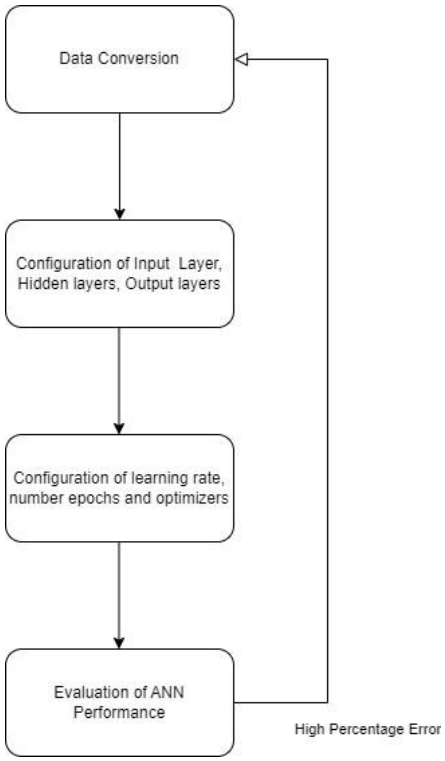


Figure 3.10: Flowchart of ANN model training procedure

3.8 Implementation of GUI

A simple graphical user interface (GUI) was designed to ease the visualization of the data. The GUI was designed using the tkinter library in python. Thus, the GUI provides a user-friendly interface that eases the interaction with the deep learning framework without the need to re-run the code or manually change the code. Furthermore, the GUI allows to inspect and analyse various factors of the model such as the training progress, loss curves and to understand the behaviour of the models easily. The figure 3.11 and 3.12 shows the design for the GUI for the ANN model. The table 3.3 below shows the function of each button.

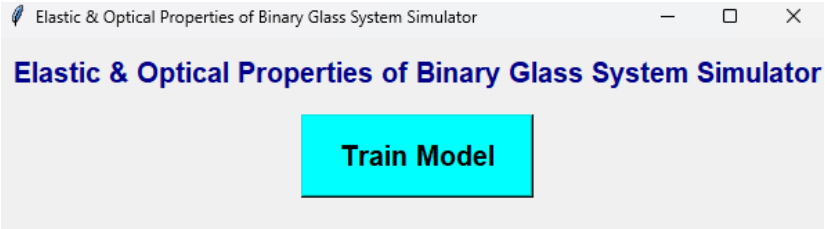


Figure 3.11 : GUI for ANN model

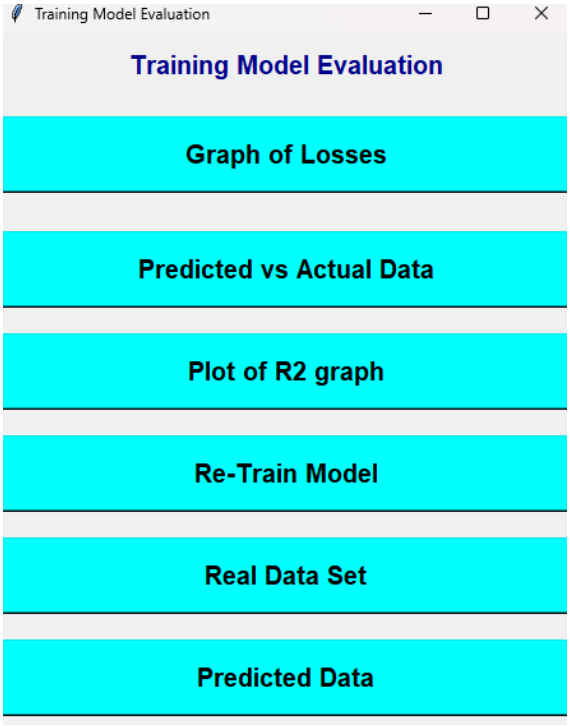


Figure 3.12 : GUI for ANN model

Table 3.3: Function of buttons in the designed GUI

Buttons	**Functions**
Train Model	- Train the ANN model.
Graph of Losses	- Plots the MSE loss graph over the number of epochs.
Predicted vs Actual Data	- Shows the graph of the predicted and real values of the parameters of the binary glass compositions.
Plot of R2 Graph	- Plots the R2 values which is the goodness of fit graph.
Re-Train Model	- Re-train the model to get a better output and desired results.
Real Data Set	- Shows the tabulated values of the real values of the parameters of the binary glass compositions.
Predicted Data Set	- Shows the tabulated values of the predicted values of the parameters of the binary glass compositions.

3.9 Gantt chart for project timeline

This project required adequate planning to finish the project within the due. Hence, a Gantt Chart was used to efficiently plan the phases of the project so that the project could be carried out systematically. The table 3.4 below shows the Gantt chart developed for final year project 1 and the table 3.5 shows the Gantt chart developed for Final Year Project 2.

Table 3.4 : Gantt chart for final year project 1

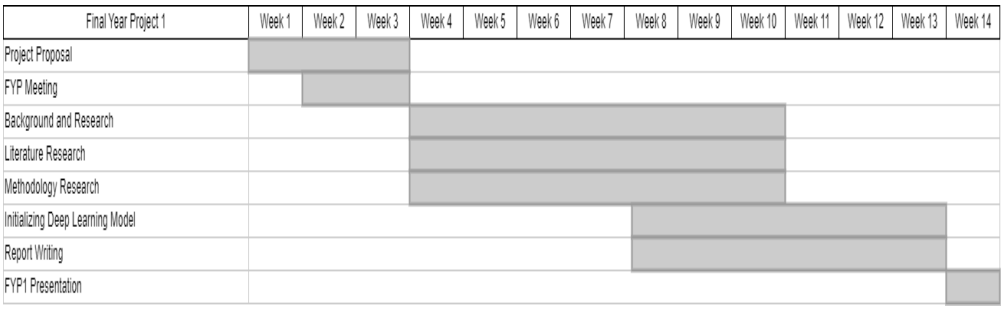
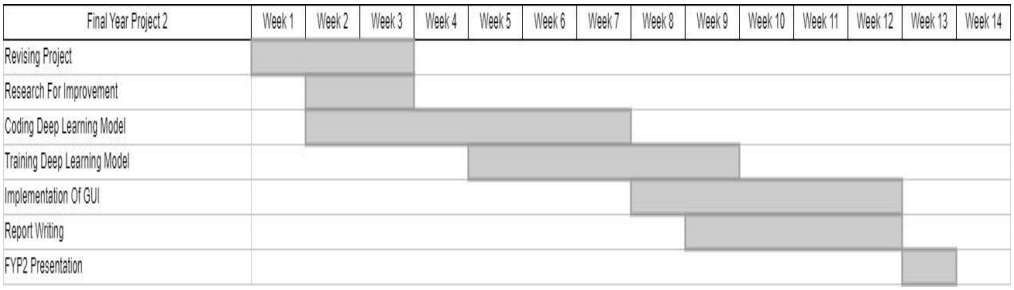


Table 3.5 :Gantt chart for final year project 2



CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Criteria and recommendation to develop ANN Model

Based on chapter 2 and chapter 3, there were several recommendations and criteria to develop a deep learning model for this specific type of project. The table 4.1 below summarizes the recommendations and criteria for building the deep learning model for this project. Thus, based on the recommendations two deep learning models, model A and model B was developed. These two model's performances were evaluated and compared with each other. The model which performed more efficiently was then selected as the final model. The evaluation of the performance is dependent on the goodness of the fit value, R^2, percentage error, and the MAPE error value of the models. The table 4.1 below shows the criteria and recommendations for developing an ANN model.

Table 4.1 : Criteria and recommendations to develop ANN model

Parameter	Recommendations
Deep Learning Model	Artificial Neural Network (ANN)
Programming Language	Python
Activation Function	1) ReLU 2) TanH
Number of hidden layers	One or two depending on the prediction accuracy

4.2 Results and analysis of Model A

The ANN model A was designed with 4 inputs and two hidden layers with 500 neurons in the first layer and 400 neurons in the second layer. The output layer has 12 outputs which are the parameters to be predicted. The ReLU activation was implemented in this deep learning model. The number of neurons in the hidden layer was experimented with high and low values before settling with the current number of neurons as it gave better output results compared to the other values. The table 4.2 below summarizes the settings of the parameters for the model A.

Table 4.2 : Parameters and settings of Model A

MODEL A	
Activation function	ReLU
Number of hidden layers	2
Number of neurons in the hidden layers	1st layer - 500 neurons 2nd layer - 400 neurons
Learning rate	0.001
Number of Epochs	5000

4.2.1 Graph of training loss over number of epochs

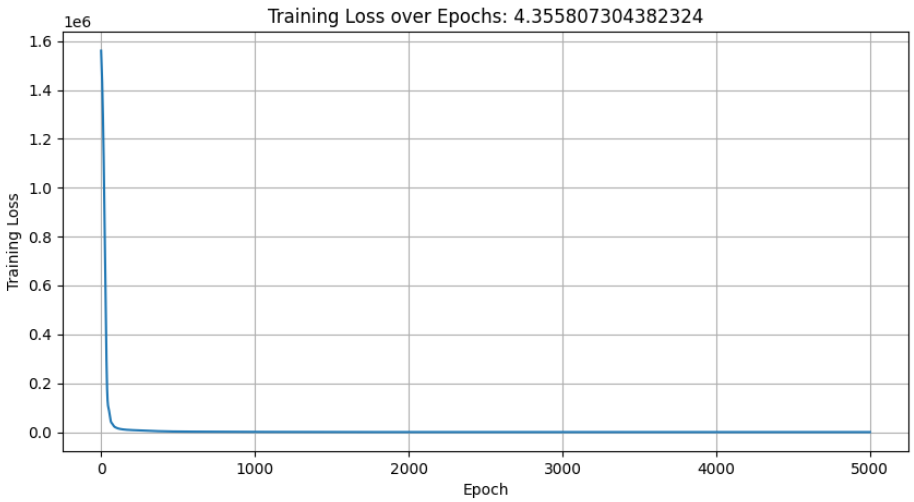


Figure 4.1 : Graph of training loss over number of epochs

Based on the figure 4.1 above, the training loss is 4.355. Thus, the training loss graph depicts that it converges indicating that the model has reached a stable point in the training process. This stability indicates that the model has learned the patterns in the training data to a reasonable extent.

4.2.2 Graphs of predicted values over experimented values

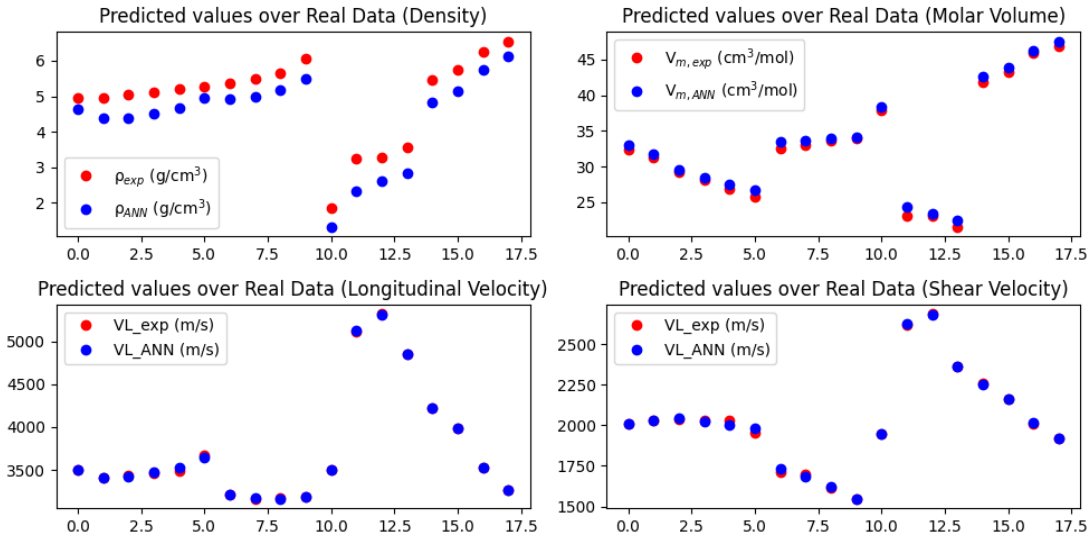


Figure 4.2 : Graphs of predicted values over experimental values

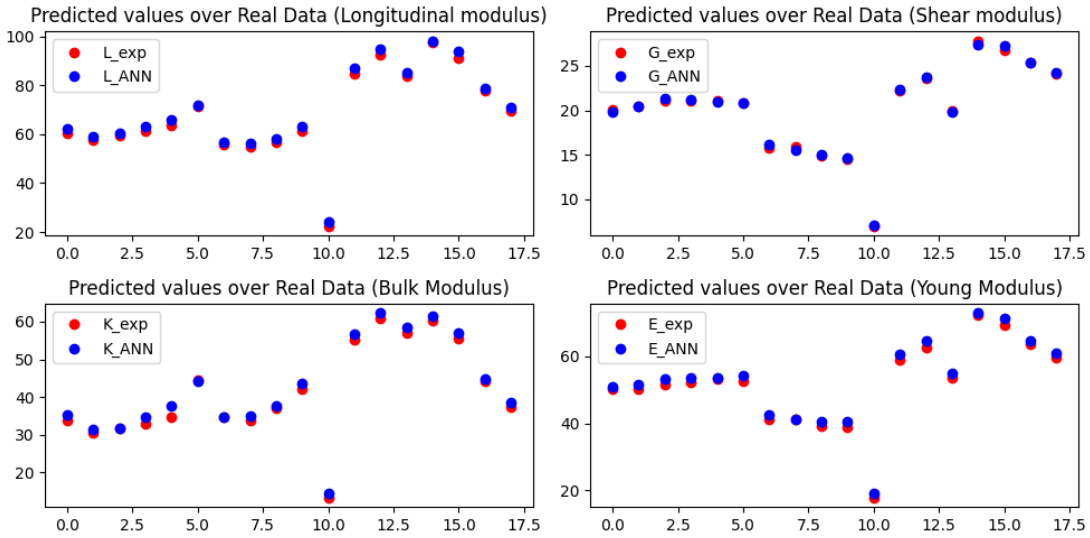


Figure 4.3 : Graphs of predicted values over experimental values

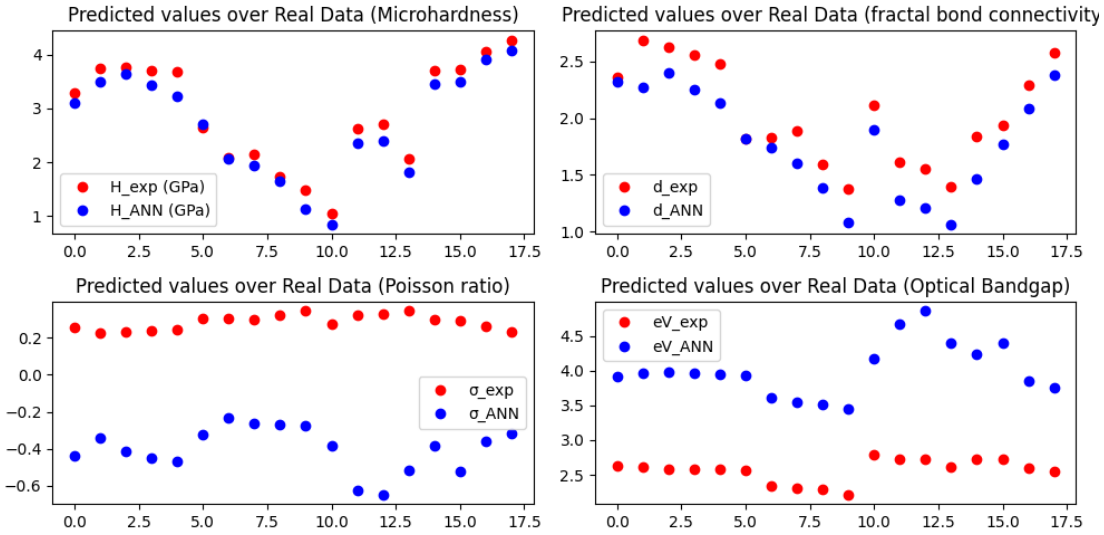


Figure 4.4 : Graphs of predicted values over experimental values

4.2.3 Graphs of R-squared value

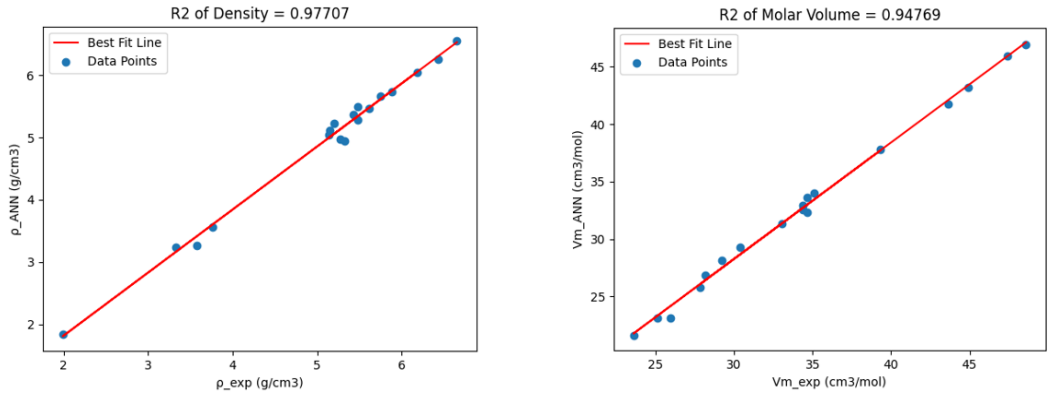


Figure 4.5 : R-squared values of density and molar volume

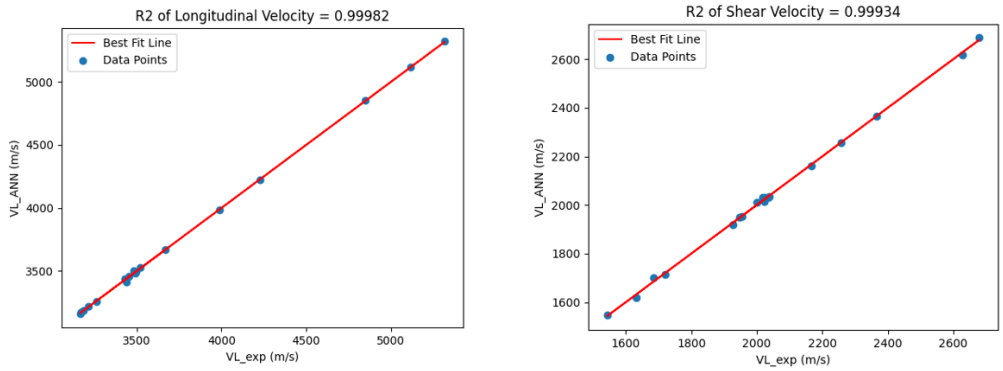


Figure 4.6 : R-squared values of longitudinal velocity and shear velocity

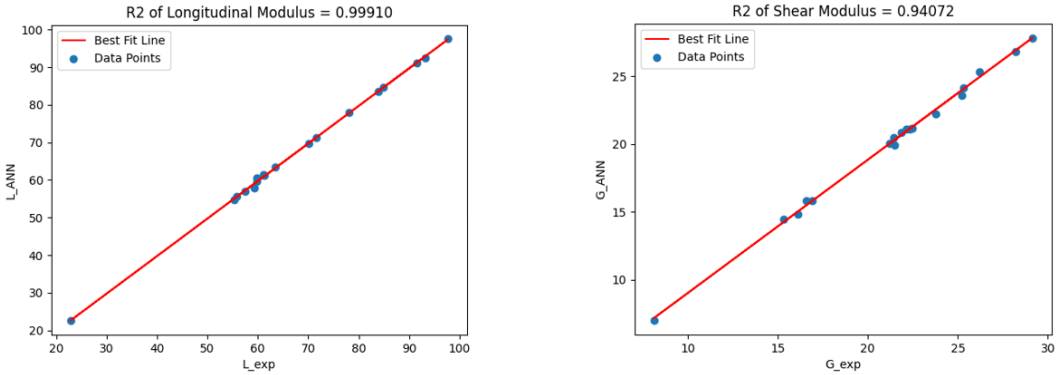


Figure 4.7 : Longitudinal Modulus and Shear Modulus

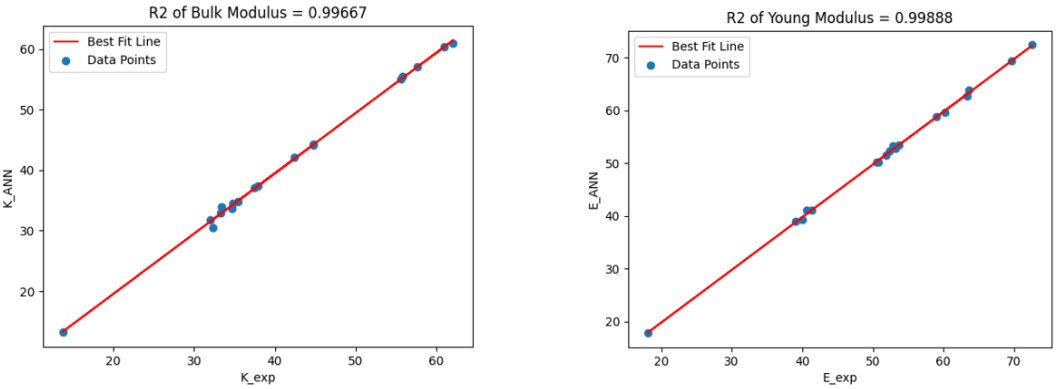


Figure 4.8: R-squared values of bulk modulus and young modulus

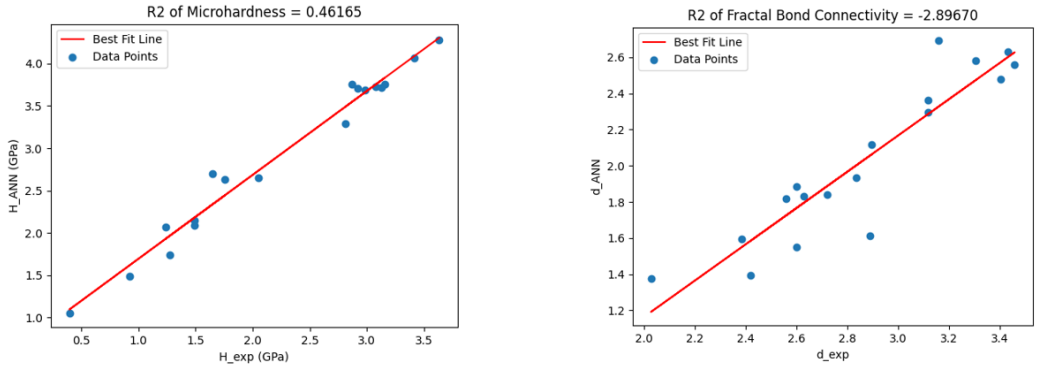


Figure 4.9: R-squared values of microhardness and fractal bond connectivity

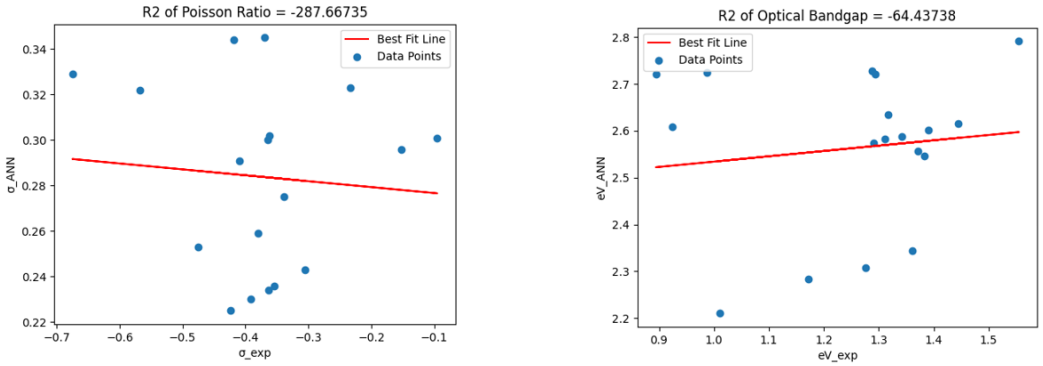


Figure 4.10 : R-squared values of Poisson Ratio and optical bandgap

4.2.4 Evaluation of R-squared values of Model A

Based on the graph plots above, the R-squared value of density, molar volume, longitudinal velocity, shear velocity, longitudinal modulus, shear modulus, bulk modulus and young modulus has an R-squared value in the range of 0.94072 to 0.9988 which indicates that there is a high correlation between the variables and parameters and implies that the model is efficiently predicting the parameters. Besides that, the R-squared value of the microhardness is 0.46165 which indicates a low correlation between the parameters and the performance of model A in predicting the microhardness value is poor and not up to the mark. On the other hand, the R-squared value fractal bond connectivity, Poisson ratio, and optical bandgap are

negative. Thus, in regression modelling a negative R-squared value should be rejected. This is because a negative R-squared value indicates that there is no correlation between the variables and the explanatory variables do not predict the changes in the dependent variable. Thus, model A has not satisfied the performance criteria of having a stable R-squared value as it could not predict certain parameters.

4.2.5 Calculation of MAPE of Model A

Table 4.3: Predicted and Experimental values of physical properties

ZnO	Bi2O3	TeO2	B2O3	Density, ρ_exp (g/cm3)	Density, ρ_ANN (g/cm3)	Percentage error % (ρ)	Molar Volume, Vm_exp (cm3/mol)	Molar Volume, Vm_ANN (cm3/mol)	Percentage error % (Vm)
0	0	100	0	4.939	4.963	0.486	32.314	33.737	4.404
5	0	95	0	4.967	4.852	2.32	31.345	32.054	2.262
15	0	85	0	5.049	4.951	1.94	29.286	29.966	2.322
20	0	80	0	5.114	4.984	2.54	28.149	29.068	3.265
25	0	75	0	5.222	5.024	3.79	26.818	28.096	4.765
30	0	70	0	5.283	5.154	2.44	25.768	26.875	4.296
0	5	95	0	5.372	5.115	4.78	32.561	33.714	3.541
0	7	93	0	5.491	5.235	4.66	32.971	33.83	2.605
0	10	90	0	5.659	5.537	2.16	33.617	34.231	1.826
0	15	85	0	6.052	6.051	0.02	33.965	34.976	2.977
0	0	0	100	1.84	1.742	5.33	37.817	38.642	2.182
45	0	0	55	3.241	2.973	8.27	23.114	24.772	7.173
50	0	0	50	3.267	3.121	4.47	23.11	24.325	5.257
60	0	0	40	3.555	3.415	4.11	21.569	22.837	5.879
0	40	0	60	5.462	5.265	3.74	41.772	43.001	2.942
0	45	0	55	5.736	5.67	1.15	43.231	44.136	2.093
0	55	0	45	6.259	6.141	1.92	45.951	46.839	1.932
0	60	0	40	6.55	6.386	2.51	46.935	48.077	2.433

Mean absolute percentage error (MAPE) of density :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\rho_{ANN} - \rho_{exp}|}{\rho_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\rho_{ANN} - \rho_{exp}|}{\rho_{exp}} = \frac{1}{18} \times 56.626 = 3.146\%$$

Mean absolute percentage error (MAPE) of molar volume :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{Vm}_{ANN} - \mathrm{Vm}_{exp}|}{\mathrm{Vm}_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{V}_{ANN} - \mathrm{V}_{exp}|}{\mathrm{V}_{exp}} = \frac{1}{18} \times 62.154 = 3.453\%$$

Table 4.4: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Longitudinal velocity, VL_exp (m/s)	Longitudinal velocity, VL_ANN (m/s)	Percentage error % (VL)	Shear velocity, VL_exp (m/s)	Shear velocity, VL_ANN (m/s)	Percentage error % (VL)
0	0	100	0	3503.000	3500.533	0.070	2013.000	2017.731	0.235
5	0	95	0	3410.000	3413.021	0.089	2031.000	2028.997	0.099
15	0	85	0	3438.000	3436.844	0.034	2035.000	2031.389	0.177
20	0	80	0	3458.000	3457.952	0.001	2032.000	2035.504	0.172
25	0	75	0	3486.000	3485.877	0.004	2030.000	2029.760	0.012
30	0	70	0	3672.000	3672.049	0.001	1953.000	1953.014	0.001
0	5	95	0	3217.000	3208.601	0.261	1715.000	1726.889	0.693
0	7	93	0	3159.000	3172.481	0.427	1699.000	1679.055	1.174
0	10	90	0	3172.000	3169.051	0.093	1618.000	1624.515	0.403
0	15	85	0	3185.000	3183.392	0.050	1547.000	1547.816	0.053
0	0	0	100	3496.000	3495.868	0.004	1948.000	1947.565	0.022
45	0	0	55	5114.000	5116.765	0.054	2619.000	2623.670	0.178
50	0	0	50	5320.000	5317.134	0.054	2688.000	2683.223	0.178
60	0	0	40	4850.000	4850.398	0.008	2366.000	2366.435	0.018
0	40	0	60	4225.000	4225.073	0.002	2257.000	2256.843	0.007
0	45	0	55	3988.000	3987.983	0.0004	2162.000	2162.550	0.025
0	55	0	45	3528.000	3527.994	0.0002	2011.000	2010.609	0.019
0	60	0	40	3260.000	3259.989	0.0003	1920.000	1919.943	0.003

Mean absolute percentage error (MAPE) of longitudinal velocity :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL}_{ANN} - \mathrm{VL}_{exp}|}{\mathrm{VL}_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL}_{ANN} - \mathrm{VL}_{exp}|}{\mathrm{VL}_{exp}} = \frac{1}{18} \times 1.152 = 0.064\%$$

Mean absolute percentage error (MAPE) of shear velocity :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|VL_{ANN} - VL_{exp}|}{VL_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|VL_{ANN} - VL_{exp}|}{VL_{exp}} = \frac{1}{18} \times 3.592 = 0.200\%$$

Table 4.5: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Longitudinal modulu, L_exp	Longitudinal modulu, L_ANN	Percentage error % (L)	Shear modulus, G_exp	Shear modulus, G_ANN	Percentage error % (G)
0	0	100	0	60.612	60.444	0.277	20.029	19.763	1.328
5	0	95	0	57.781	57.848	0.116	20.494	20.612	0.576
15	0	85	0	59.698	60.023	0.544	21.121	20.867	1.203
20	0	80	0	61.181	61.428	0.404	21.125	21.062	0.298
25	0	75	0	63.476	63.000	0.750	21.137	21.142	0.024
30	0	70	0	71.240	71.526	0.401	20.866	20.849	0.081
0	5	95	0	55.629	55.234	0.710	15.813	16.020	1.309
0	7	93	0	54.800	55.446	1.179	15.858	15.571	1.810
0	10	90	0	56.942	57.414	0.829	14.826	15.051	1.518
0	15	85	0	61.394	61.071	0.526	14.489	14.273	1.491
0	0	0	100	22.511	22.594	0.369	6.986	6.954	0.458
45	0	0	55	84.777	84.772	0.006	22.240	22.061	0.805
50	0	0	50	92.471	92.961	0.530	23.615	23.752	0.580
60	0	0	40	83.654	83.505	0.178	19.904	19.720	0.924
0	40	0	60	97.525	97.179	0.355	27.833	27.608	0.808
0	45	0	55	91.245	91.677	0.473	26.827	26.885	0.216
0	55	0	45	77.909	77.756	0.196	25.336	25.230	0.418
0	60	0	40	69.625	69.914	0.415	24.154	24.160	0.025

Mean absolute percentage error (MAPE) of longitudinal modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|L_{ANN} - L_{exp}|}{L_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|L_{ANN} - L_{exp}|}{L_{exp}} = \frac{1}{18} \times 9.833 = 0.546\ \%$$

Mean absolute percentage error (MAPE) of shear modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|G_{ANN} - G_{exp}|}{G_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|G_{ANN} - G_{exp}|}{G_{exp}} = \frac{1}{18} \times 14.736 = 0.819\%$$

Table 4.6: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Bulk modulus, K_exp	Bulk modulus, K_ANN	Percentage error % (K)	Young modulus, E_exp	Young modulus, E_ANN	Percentage error % (E)
0	0	100	0	33.906	34.185	0.823	50.203	49.629	1.143
5	0	95	0	30.455	30.906	1.481	50.218	50.492	0.546
15	0	85	0	31.81	32.387	1.814	51.467	51.574	0.208
20	0	80	0	33.014	33.379	1.106	52.234	52.233	0.002
25	0	75	0	34.759	34.941	0.524	53.252	52.639	1.151
30	0	70	0	44.351	44.871	1.172	52.737	52.739	0.004
0	5	95	0	34.544	34.102	1.280	41.16	41.275	0.279
0	7	93	0	33.655	35.006	4.014	41.117	40.423	1.688
0	10	90	0	37.173	37.679	1.361	39.26	39.684	1.080
0	15	85	0	42.076	42.19	0.271	38.991	38.775	0.554
0	0	0	100	13.196	13.557	2.736	17.815	17.736	0.443
45	0	0	55	55.124	55.608	0.878	58.81	58.426	0.653
50	0	0	50	60.984	61.791	1.323	62.747	62.876	0.206
60	0	0	40	57.116	57.503	0.678	53.497	53.262	0.439
0	40	0	60	60.415	60.669	0.420	72.382	72.082	0.414
0	45	0	55	55.475	56.044	1.026	69.309	69.411	0.147
0	55	0	45	44.128	44.41	0.639	63.798	63.301	0.779
0	60	0	40	37.419	37.947	1.411	59.632	59.798	0.278

Mean absolute percentage error (MAPE) of bulk modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|K_{ANN} - K_{exp}|}{K_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|K_{ANN} - K_{exp}|}{K_{exp}} = \frac{1}{18} \times 22.175 = 1.231\%$$

Mean absolute percentage error (MAPE) of young modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|E_{ANN} - E_{exp}|}{E_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|E_{ANN} - E_{exp}|}{E_{exp}} = \frac{1}{18} \times 10.365 = 0.576\%$$

Table 4.7: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Microhard-ness, H_exp (GPa)	Microhard-ness, H_ANN (GPa)	Percentage error % (H)	fractal bond connectivity , d_exp	fractal bond connectivity , d_ANN	Percentage error % (d)	Poisson ratio, σ_exp	Poisson ratio, σ_ANN	Percentage error % (d)
0	0	100	0	3.295	2.181	33.809	2.362	2.109	10.711	0.253	0.016	93.676
5	0	95	0	3.754	3.023	19.473	2.691	2.324	13.638	0.225	0.03	86.667
15	0	85	0	3.76	2.909	22.633	2.63	2.245	14.639	0.23	0.163	29.130
20	0	80	0	3.713	2.861	22.946	2.559	2.233	12.739	0.236	0.087	63.136
25	0	75	0	3.686	2.812	23.711	2.478	2.192	11.542	0.243	0.055	77.366
30	0	70	0	2.654	1.692	36.247	1.818	1.521	16.337	0.302	0.322	6.623
0	5	95	0	2.093	1.336	36.168	1.831	1.589	13.217	0.301	0.233	22.591
0	7	93	0	2.152	1.307	39.266	1.884	1.527	18.949	0.296	0.289	2.365
0	10	90	0	1.739	1.056	39.275	1.595	1.341	15.925	0.323	0.286	11.455
0	15	85	0	1.491	0.626	58.015	1.377	1.059	23.094	0.345	0.24	30.435
0	0	0	100	1.048	0.305	70.897	2.118	1.846	12.842	0.275	0.202	26.545
45	0	0	55	2.636	1.37	48.027	1.614	1.2	25.651	0.322	0.301	6.522
50	0	0	50	2.7	1.485	45.000	1.549	1.143	26.210	0.329	0.287	12.766
60	0	0	40	2.071	0.941	54.563	1.394	0.932	33.142	0.344	0.117	65.988
0	40	0	60	3.705	2.758	25.560	1.842	1.445	21.553	0.3	0.095	68.333
0	45	0	55	3.724	2.78	25.349	1.934	1.575	18.563	0.291	0.214	26.460
0	55	0	45	4.069	3.222	20.816	2.296	2.055	10.497	0.259	0.201	22.394
0	60	0	40	4.276	3.367	21.258	2.582	3.367	30.403	0.234	0.207	11.538

Mean absolute percentage error (MAPE) of microhardness:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|H_{ANN} - H_{exp}|}{E_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|H_{ANN} - H_{exp}|}{H_{exp}} = \frac{1}{18} \times 654.809 = 36.378\%$$

Mean absolute percentage error (MAPE) of fractal bond connectivity:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathbf{d}_{ANN} - \mathbf{d}_{exp}|}{\mathbf{d}_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathbf{d}_{ANN} - \mathbf{d}_{exp}|}{\mathbf{d}_{exp}} = \frac{1}{18} \times 292.324 = 16.240\%$$

Mean absolute percentage error (MAPE) of Poisson ratio

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\sigma_{ANN} - \sigma_{exp}|}{\sigma_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\sigma_{ANN} - \sigma_{exp}|}{\sigma_{exp}} = \frac{1}{18} \times 646.920 = 35.94\ \%$$

Table 4.8: Predicted and Experimental values of optical properties

ZnO	Bi2O3	TeO2	B2O3	Optical bandgap (eV_exp)	Optical bandgap (eV_ANN)	Percentage error % (eV)
0	0	100	0	2.634	4.965	88.497
5	0	95	0	2.616	4.955	89.411
15	0	85	0	2.588	5.034	94.513
20	0	80	0	2.582	4.981	92.912
25	0	75	0	2.574	4.949	92.269
30	0	70	0	2.557	5.025	96.519
0	5	95	0	2.343	4.572	95.134
0	7	93	0	2.308	4.522	95.927
0	10	90	0	2.284	4.493	96.716
0	15	85	0	2.21	4.387	98.507
0	0	0	100	2.791	4.822	72.770
45	0	0	55	2.725	5.964	118.862
50	0	0	50	2.721	6.134	125.432
60	0	0	40	2.608	5.726	119.555
0	40	0	60	2.728	5.333	95.491
0	45	0	55	2.721	5.299	94.745
0	55	0	45	2.602	5.044	93.851
0	60	0	40	2.546	4.845	90.299

Mean absolute percentage error (MAPE) of optical bandgap.

$$\boldsymbol{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{eV_{ANN}} - \mathrm{eV_{exp}}|}{\mathrm{eV_{exp}}}$$

$$\boldsymbol{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{eV_{ANN}} - \mathrm{eV_{exp}}|}{\mathrm{eV_{exp}}} = \frac{1}{18} \times 1471.333 = 81.741\%$$

Calculation of total mean average percentage error of model B

$$\boldsymbol{MAPE_A} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{A_{ANN}} - \mathrm{A_{exp}}|}{\mathrm{A_{exp}}}$$

$$\sum_{i=1}^{12}\frac{|\mathrm{A_{ANN}} - \mathrm{A_{exp}}|}{\mathrm{A_{exp}}} = 180.334$$

$$\boldsymbol{MAPE_A} = \frac{1}{12}\sum_{i=1}^{12}\frac{|\mathrm{A_{ANN}} - \mathrm{A_{exp}}|}{\mathrm{A_{exp}}} = \frac{1}{12} \times 233.818 = 15.028\ \%$$

4.2.6 Evaluation of MAPE of Model A

Based on the MAPE calculated above for each parameter the MAPE for the physical properties density and molar volume are 3.146 % and 3.453 % which are less than 10 %. Hence according to the interpretation of MAPE values, as shown in figure 3.9, MAPE values that are less than 10 are considered as high accurate forecasting. Thus, the prediction of the physical properties by model A is considered to be excellent. Besides that, the predictions of the elastic properties which are longitudinal velocity, shear velocity, longitudinal modulus, shear modulus, bulk modulus and young modulus are considered to be excellent as they range from 0.064 % to 0.819% thus indicating that model A predicts their values efficiently. However, the MAPE values of the remaining elastic properties are not up to the mark. The MAPE value for fractal bond connectivity is 16.240 % which according to the interpretation table is good forecasting. On the other hand, the MAPE values of Poisson ratio and microhardness are 35.940 and 36.378 which is considered to be a reasonable forecast. The MAPE value for the optical property, optical bandgap is calculated to be 81.741 % which is an inaccurate forecasting. The overall MAPE of model A is calculated to be 15.028% which is good forecasting. However, model A is still lacks the capability to predict some of the important parameters despite its overall MAPE value.

4.3 Results and analysis of Model B

The ANN model B was designed with 4 inputs and one hidden layer with 4000 neurons in the first layer. The output layer has 12 outputs which are the parameters to be predicted. The TanH activation was implemented in this deep learning model. The number of neurons in the hidden layer was experimented with high and low values before settling the current number of neurons as it gave better output results compared to the other values. The table below summarizes the settings of the parameters for the model B.

Table 4.9 : Parameters and settings of Model B

MODEL B	
Activation function	TanH
Number of hidden layers	1
Number of neurons in the hidden layers	1st layer - 4000 neurons
Learning rate	0.001
Number of Epochs	12000

4.3.1 Graph of training loss over number of epochs

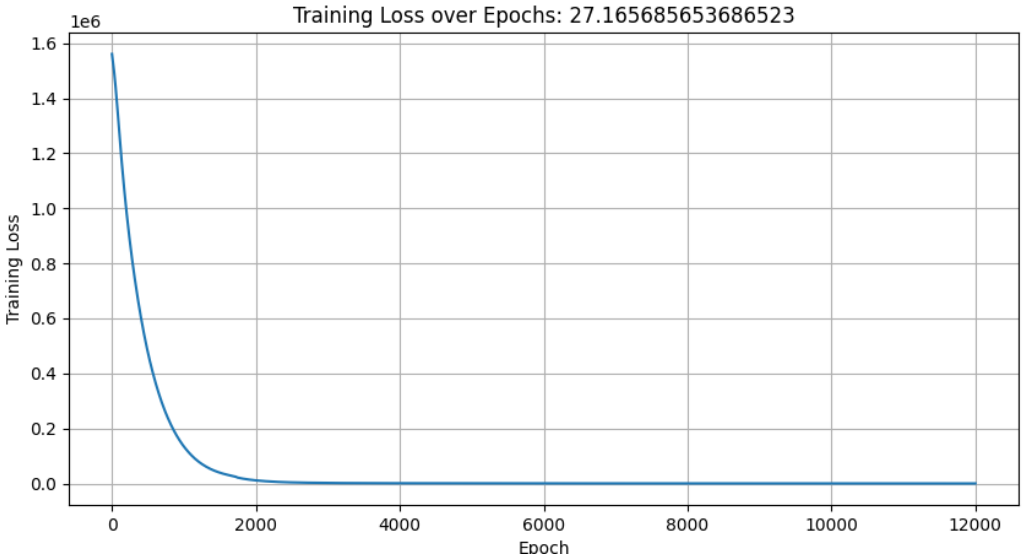


Figure 4.11: Graph of training loss over number of epochs

Based on the figure 4.11 above, the training loss is 27.166 Thus, the training loss graph depicts that it converges indicating that the model has reached a stable point in the training process. This stability indicates that the model has learned the patterns in the training data to a reasonable extent.

4.3.2 Graphs of predicted values over experimental values

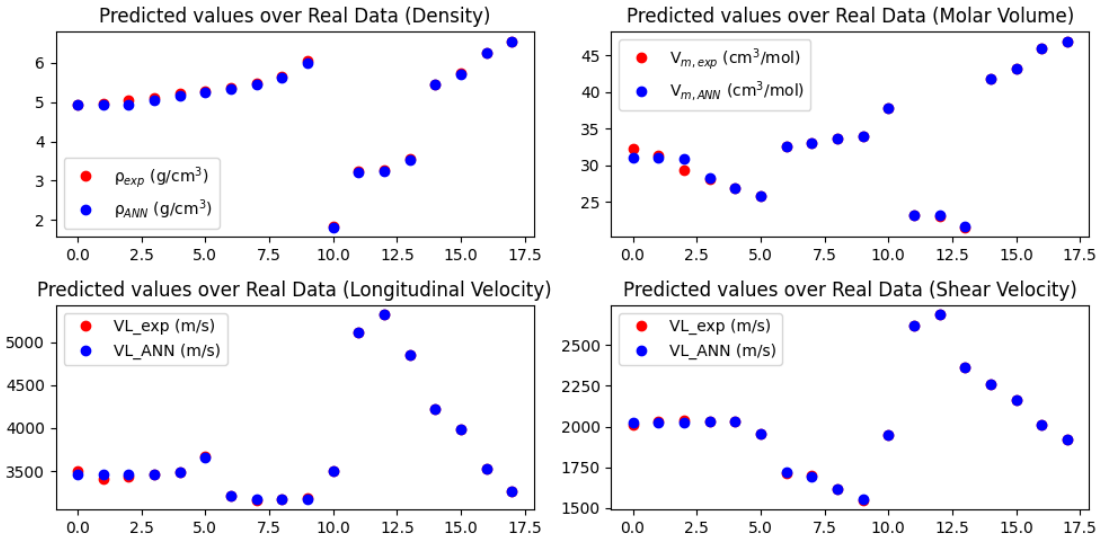


Figure 4.12 : Graphs of predicted values over experimental values

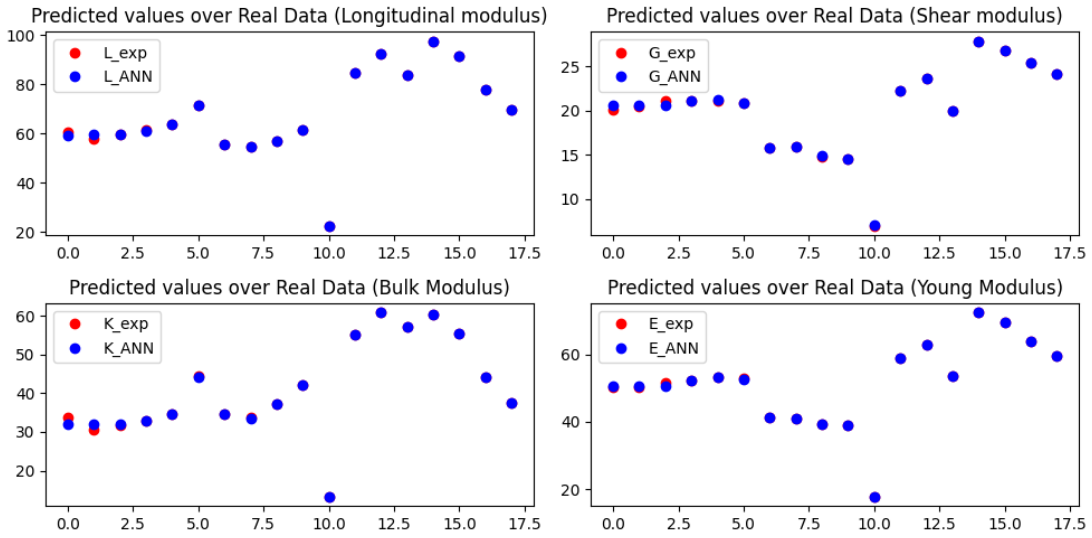


Figure 4.13 : Graphs of predicted values over experimental values

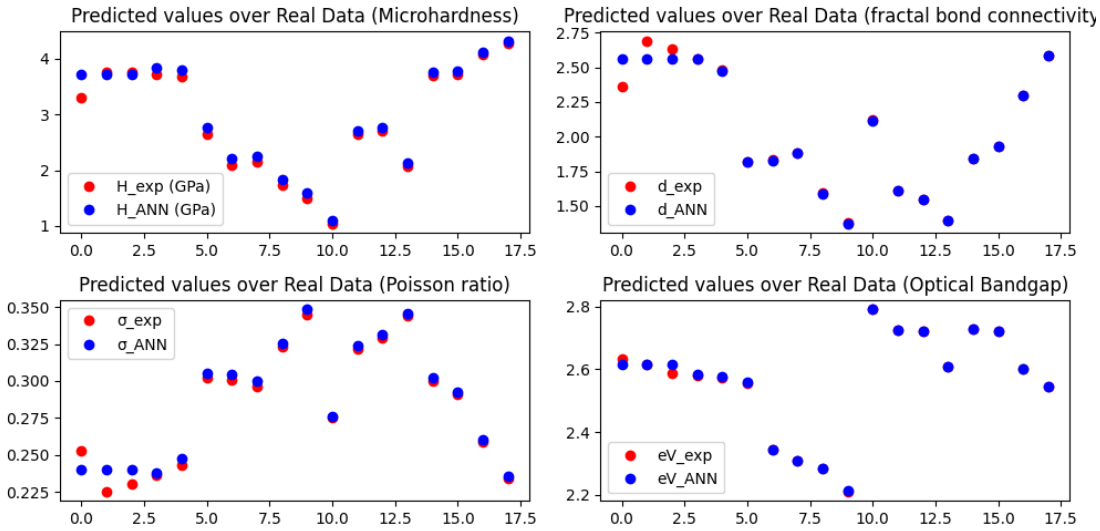


Figure 4.14 : Graphs of predicted values over experimental values

4.3.3 Graphs of R-squared values

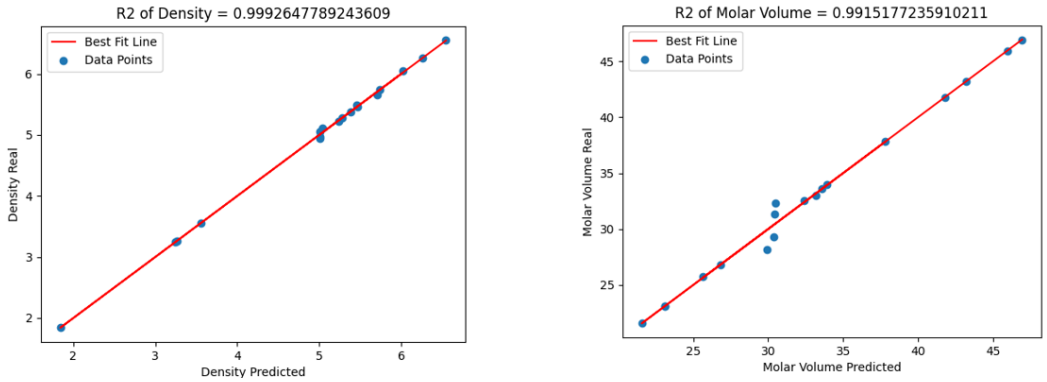


Figure 4.15 : R-squared values of density and molar volume

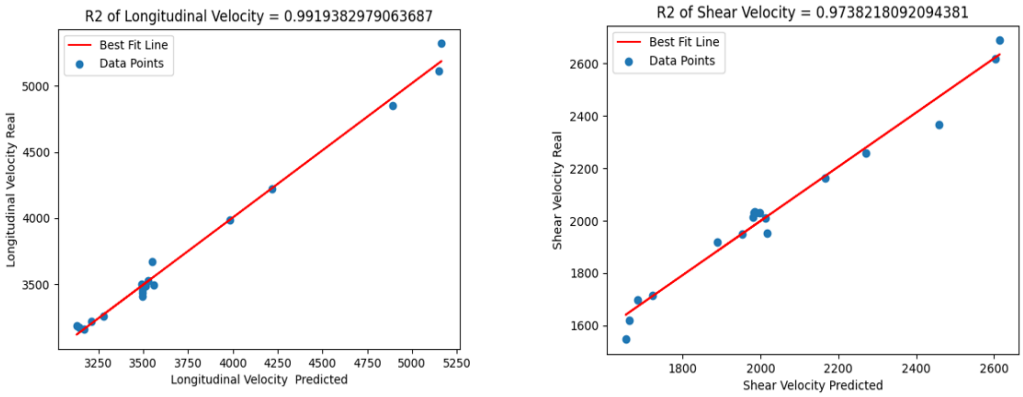


Figure 4.16: R-squared values of longitudinal velocity and shear velocity

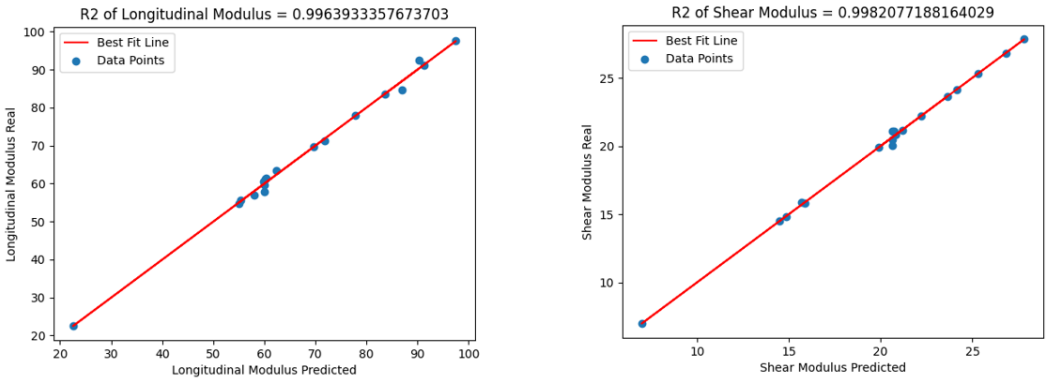


Figure 4.17: Longitudinal Modulus and Shear Modulus

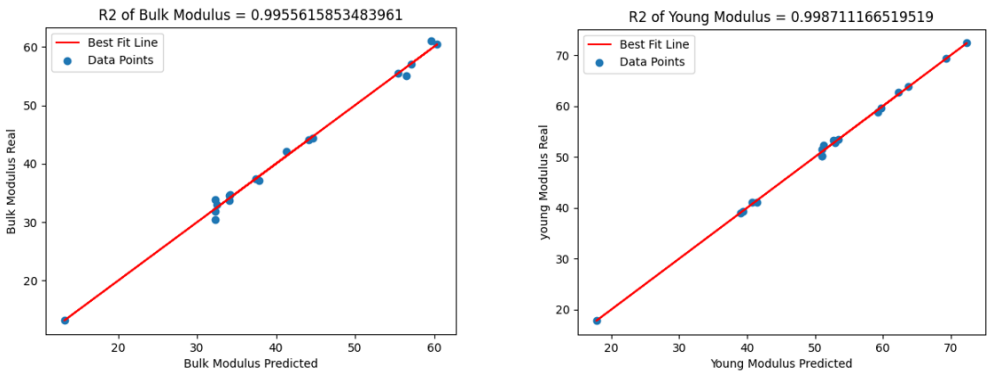


Figure 4.18 : R-squared values of bulk modulus and young modulus

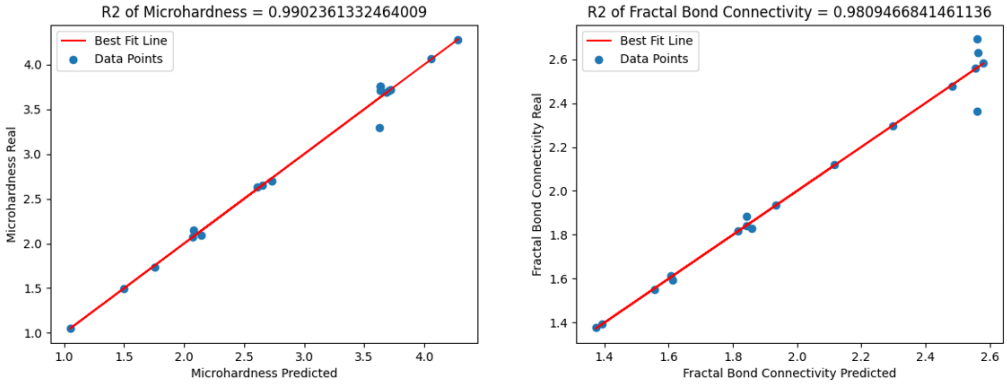


Figure 4.19: R-squared values of microhardness and fractal bond connectivity

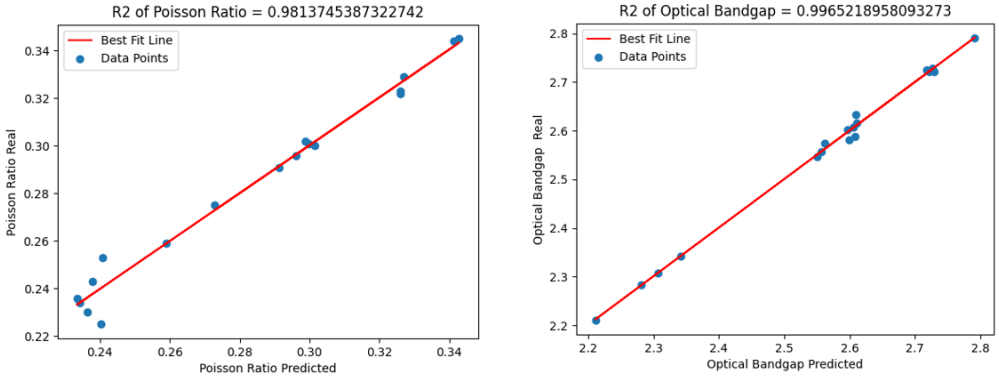


Figure 4.20: R-squared values of Poisson Ratio

4.3.4 Evaluation of R-squared values of Model B

Based on the graph plots above, the R-squared value for all the 12 parameters has an R-squared value in the range of 0.974 to 0.999 which indicates that there is a high correlation between the variables and parameters and implies that the model is efficiently predicting the parameters. Thus, model B has no negative R-squared values which is an indication that the model predicts all the values accurately and there is a solid correlation between the variables and parameters.

4.3.5 Calculation of MAPE for Model B

Table 4.10 : Predicted and Experimental values of physical properties

ZnO	Bi2O3	TeO2	B2O3	Density, ρ_exp (g/cm3)	Density, ρ_ANN (g/cm3)	Percentage error % (ρ)	Molar Volume, Vm_exp (cm3/mol)	Molar Volume, Vm_ANN (cm3/mol)	Percentage error % (Vm)
0	0	100	0	4.939	5.005	1.336	32.314	30.479	5.679
5	0	95	0	4.967	5.005	0.765	31.345	30.427	2.929
15	0	85	0	5.049	5.005	0.871	29.286	30.397	3.794
20	0	80	0	5.114	5.038	1.486	28.149	29.925	6.309
25	0	75	0	5.222	5.238	0.306	26.818	26.825	0.026
30	0	70	0	5.283	5.284	0.019	25.768	25.629	0.539
0	5	95	0	5.372	5.38	0.149	32.561	32.397	0.504
0	7	93	0	5.491	5.461	0.546	32.971	33.179	0.631
0	10	90	0	5.659	5.711	0.919	33.617	33.595	0.065
0	15	85	0	6.052	6.022	0.496	33.965	33.942	0.068
0	0	0	100	1.84	1.84	0.000	37.817	37.819	0.005
45	0	0	55	3.241	3.241	0.000	23.114	23.087	0.117
50	0	0	50	3.267	3.266	0.031	23.11	23.135	0.108
60	0	0	40	3.555	3.557	0.056	21.569	21.568	0.005
0	40	0	60	5.462	5.463	0.018	41.772	41.78	0.019
0	45	0	55	5.736	5.734	0.035	43.231	43.219	0.028
0	55	0	45	6.259	6.262	0.048	45.951	45.944	0.015
0	60	0	40	6.55	6.547	0.046	46.935	46.946	0.023

Mean absolute percentage error (MAPE) of density :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\rho_{ANN} - \rho_{exp}|}{\rho_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\rho_{ANN} - \rho_{exp}|}{\rho_{exp}} = \frac{1}{18} \times 7.128 = 0.396\%$$

Mean absolute percentage error (MAPE) of molar volume :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Vm_{ANN} - Vm_{exp}|}{Vm_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|V_{ANN} - V_{exp}|}{V_{exp}} = \frac{1}{18} \times 20.864 = 1.159\%$$

Table 4.11: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Longitudinal velocity, VL_exp (m/s)	Longitudinal velocity, VL_ANN (m/s)	Percentage error % (VL)	Shear velocity, VL_exp (m/s)	Shear velocity, VL_ANN (m/s)	Percentage error % (VL)
0	0	100	0	3503.000	3492.125	0.310	2013.000	1981.224	1.579
5	0	95	0	3410.000	3494.689	2.484	2031.000	1983.544	2.337
15	0	85	0	3438.000	3495.647	1.677	2035.000	1984.371	2.488
20	0	80	0	3458.000	3498.175	1.162	2032.000	1986.375	2.245
25	0	75	0	3486.000	3513.087	0.777	2030.000	1996.919	1.630
30	0	70	0	3672.000	3549.550	3.335	1953.000	2017.126	3.283
0	5	95	0	3217.000	3213.643	0.1044	1715.000	1722.794	0.454
0	7	93	0	3159.000	3168.927	0.314	1699.000	1683.259	0.926
0	10	90	0	3172.000	3142.347	0.9348	1618.000	1662.604	2.757
0	15	85	0	3185.000	3128.172	1.784	1547.000	1653.367	6.876
0	0	0	100	3496.000	3559.584	1.819	1948.000	1953.515	0.283
45	0	0	55	5114.000	5150.854	0.721	2619.000	2604.045	0.571
50	0	0	50	5320.000	5165.446	2.905	2688.000	2615.387	2.701
60	0	0	40	4850.000	4892.532	0.877	2366.000	2457.914	3.885
0	40	0	60	4225.000	4218.775	0.147	2257.000	2270.576	0.602
0	45	0	55	3988.000	3986.177	0.046	2162.000	2166.564	0.211
0	55	0	45	3528.000	3527.365	0.0180	2011.000	2102.245	4.537
0	60	0	40	3260.000	3277.869	0.548	1920.000	1889.402	1.594

Mean absolute percentage error (MAPE) of longitudinal velocity :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL_{ANN}} - \mathrm{VL_{exp}}|}{\mathrm{VL_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL_{ANN}} - \mathrm{VL_{exp}}|}{\mathrm{VL_{exp}}} = \frac{1}{18} \times 19.963 = 1.109\%$$

Mean absolute percentage error (MAPE) of shear velocity :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL_{ANN}} - \mathrm{VL_{exp}}|}{\mathrm{VL_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{VL_{ANN}} - \mathrm{VL_{exp}}|}{\mathrm{VL_{exp}}} = \frac{1}{18} \times 38.959 = 2.164\%$$

Table 4.12 : Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Longitudinal modulu, L_exp	Longitudinal modulu, L_ANN	Percentage error % (L)	Shear modulus, G_exp	Shear modulus, G_ANN	Percentage error % (G)
0	0	100	0	60.612	59.904	1.168	20.029	20.624	2.971
5	0	95	0	57.781	59.960	3.771	20.494	20.648	0.751
15	0	85	0	59.698	59.952	0.425	21.121	20.653	2.216
20	0	80	0	61.181	60.111	1.749	21.125	20.740	1.822
25	0	75	0	63.476	62.338	1.793	21.137	21.209	0.341
30	0	70	0	71.240	71.795	0.779	20.866	20.840	0.125
0	5	95	0	55.629	55.277	0.633	15.813	15.896	0.525
0	7	93	0	54.800	55.010	0.383	15.858	15.708	0.946
0	10	90	0	56.942	58.052	1.949	14.826	14.862	0.243
0	15	85	0	61.394	60.352	1.697	14.489	14.490	0.007
0	0	0	100	22.511	22.507	0.018	6.986	6.983	0.043
45	0	0	55	84.777	86.918	2.525	22.240	22.210	0.135
50	0	0	50	92.471	90.299	2.349	23.615	23.643	0.119
60	0	0	40	83.654	83.691	0.044	19.904	19.892	-0.060
0	40	0	60	97.525	97.506	0.019	27.833	27.828	0.018
0	45	0	55	91.245	91.271	0.028	26.827	26.827	0.000
0	55	0	45	77.909	77.867	0.054	25.336	25.320	0.063
0	60	0	40	69.625	69.666	0.059	24.154	24.162	0.033

Mean absolute percentage error (MAPE) of longitudinal modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|L_{ANN} - L_{exp}|}{L_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|L_{ANN} - L_{exp}|}{L_{exp}} = \frac{1}{18} \times 19.445 = 1.08\%$$

Mean absolute percentage error (MAPE) of shear modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|G_{ANN} - G_{exp}|}{G_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|G_{ANN} - G_{exp}|}{G_{exp}} = \frac{1}{18} \times 10.297 = 0.57\%$$

Table 4.13 : Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Bulk modulus, K_exp	Bulk modulus, K_ANN	Percentage error % (K)	Young modulus, E_exp	Young modulus, E_ANN	Percentage error % (E)
0	0	100	0	33.906	32.318	4.684	50.203	50.995	1.578
5	0	95	0	30.455	32.338	6.183	50.218	51.07	1.697
15	0	85	0	31.81	32.336	1.654	51.467	51.061	0.789
20	0	80	0	33.014	32.492	1.581	52.234	51.233	-1.916
25	0	75	0	34.759	34.203	1.600	53.252	52.718	-1.003
30	0	70	0	44.351	44.664	0.706	52.737	53.009	-0.516
0	5	95	0	34.544	34.112	1.251	41.16	41.505	0.838
0	7	93	0	33.655	34.09	1.293	41.117	40.694	1.029
0	10	90	0	37.173	37.853	1.829	39.26	39.353	0.237
0	15	85	0	42.076	41.336	1.759	38.991	39.004	-0.033
0	0	0	100	13.196	13.195	0.008	17.815	17.814	-0.006
45	0	0	55	55.124	56.457	2.418	58.81	59.284	0.806
50	0	0	50	60.984	59.624	2.230	62.747	62.29	-0.728
60	0	0	40	57.116	57.141	0.044	53.497	53.484	-0.024
0	40	0	60	60.415	60.412	0.005	72.382	72.364	0.025
0	45	0	55	55.475	55.481	0.011	69.309	69.335	0.038
0	55	0	45	44.128	44.117	0.025	63.798	63.717	0.127
0	60	0	40	37.419	37.429	0.027	59.632	59.699	0.112

Mean absolute percentage error (MAPE) of bulk modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{K_{ANN}} - \mathrm{K_{exp}}|}{\mathrm{K_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{K_{ANN}} - \mathrm{K_{exp}}|}{\mathrm{K_{exp}}} = \frac{1}{18} \times 27.305 = 1.517\%$$

Mean absolute percentage error (MAPE) of young modulus:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{E_{ANN}} - \mathrm{E_{exp}}|}{\mathrm{E_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{E_{ANN}} - \mathrm{E_{exp}}|}{\mathrm{E_{exp}}} = \frac{1}{18} \times 3.048 = 0.170\%$$

Table 4.14: Predicted and Experimental values of elastic properties

ZnO	Bi2O3	TeO2	B2O3	Microhard-ness, H_exp (GPa)	Microhard-ness, H_ANN (GPa)	Percentage error % (H)	fractal bond connectivity, d_exp	fractal bond connectivity, d_ANN	Percentage error % (d)	Poisson ratio, σ_exp	Poisson ratio, σ_ANN	Percentage error % (d)
0	0	100	0	3.295	3.629	10.137	2.362	2.559	8.340	0.253	0.241	4.743
5	0	95	0	3.754	3.633	3.223	2.691	2.56	4.868	0.225	0.24	6.667
15	0	85	0	3.76	3.632	3.404	2.63	2.562	2.586	0.23	0.236	2.609
20	0	80	0	3.713	3.632	2.182	2.559	2.556	0.117	0.236	0.233	1.271
25	0	75	0	3.686	3.684	0.054	2.478	2.483	0.202	0.243	0.238	2.058
30	0	70	0	2.654	2.651	0.113	1.818	1.815	0.165	0.302	0.299	0.993
0	5	95	0	2.093	2.146	2.532	1.831	1.86	1.584	0.301	0.3	0.332
0	7	93	0	2.152	2.079	3.392	1.884	1.843	2.176	0.296	0.296	0.000
0	10	90	0	1.739	1.759	1.150	1.595	1.614	1.191	0.323	0.326	0.929
0	15	85	0	1.491	1.494	0.201	1.377	1.373	0.290	0.345	0.343	0.580
0	0	0	100	1.048	1.05	0.191	2.118	2.116	0.094	0.275	0.273	0.727
45	0	0	55	2.636	2.61	0.986	1.614	1.607	0.434	0.322	0.326	1.242
50	0	0	50	2.7	2.729	1.074	1.549	1.557	0.516	0.329	0.327	0.608
60	0	0	40	2.071	2.068	0.145	1.394	1.392	0.143	0.344	0.341	0.872
0	40	0	60	3.705	3.705	0.000	1.842	1.843	0.054	0.3	0.301	0.333
0	45	0	55	3.724	3.724	0.000	1.934	1.935	0.052	0.291	0.291	0.000
0	55	0	45	4.069	4.061	0.197	2.296	2.298	0.087	0.259	0.259	0.000
0	60	0	40	4.276	4.283	0.164	2.582	2.58	0.077	0.234	0.234	0.000

Mean absolute percentage error (MAPE) of microhardness:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{H_{ANN}} - \mathrm{H_{exp}}|}{\mathrm{E_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{H_{ANN}} - \mathrm{H_{exp}}|}{\mathrm{H_{exp}}} = \frac{1}{18} \times 29.145 = 1.619\%$$

Mean absolute percentage error (MAPE) of fractal bond connectivity:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{d_{ANN}} - \mathrm{d_{exp}}|}{\mathrm{d_{exp}}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{d_{ANN}} - \mathrm{d_{exp}}|}{\mathrm{d_{exp}}} = \frac{1}{18} \times 22.978 = 1.277\%$$

Mean absolute percentage error (MAPE) of poisson ratio

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\sigma_{ANN} - \sigma_{exp}|}{\sigma_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\sigma_{ANN} - \sigma_{exp}|}{\sigma_{exp}} = \frac{1}{18} \times 23.964 = 1.33\%$$

Table 4.15: Predicted and Experimental values of optical properties

ZnO	Bi2O3	TeO2	B2O3	Optical bandgap (eV_exp)	Optical bandgap (eV_ANN)	Percentage error % (eV)
0	0	100	0	2.634	2.609	0.949
5	0	95	0	2.616	2.61	0.229
15	0	85	0	2.588	2.608	0.773
20	0	80	0	2.582	2.599	0.658
25	0	75	0	2.574	2.562	0.466
30	0	70	0	2.557	2.557	0.000
0	5	95	0	2.343	2.342	0.043
0	7	93	0	2.308	2.307	0.043
0	10	90	0	2.284	2.282	0.088
0	15	85	0	2.21	2.211	0.045
0	0	0	100	2.791	2.792	0.036
45	0	0	55	2.725	2.718	0.257
50	0	0	50	2.721	2.729	0.294
60	0	0	40	2.608	2.606	0.077
0	40	0	60	2.728	2.727	0.037
0	45	0	55	2.721	2.722	0.037
0	55	0	45	2.602	2.596	0.231
0	60	0	40	2.546	2.55	0.157

Mean absolute percentage error (MAPE) of optical bandgap.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|eV_{ANN} - eV_{exp}|}{eV_{exp}}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|eV_{ANN} - eV_{exp}|}{eV_{exp}} = \frac{1}{18} \times 4.419 = 0.2455\%$$

Calculation of total mean average percentage error of model B

$$MAPE_A = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_{ANN} - A_{exp}|}{A_{exp}}$$

$$\sum_{i=1}^{12}\frac{|A_{ANN} - A_{exp}|}{A_{exp}} = 12.6365\%$$

$$MAPE_B = \frac{1}{12}\sum_{i=1}^{12}\frac{|A_{ANN} - A_{exp}|}{A_{exp}} = \frac{1}{12} \times 12.6365 = 1.053\ \%$$

4.3.6 Evaluation of MAPE of Model B

Based on the MAPE calculated above for each parameter the MAPE for the physical properties density and molar volume are 0.396% and 1.159% respectively which are less than 10%. Hence according to the interpretation of MAPE values, as shown in figure 3.9, MAPE values that are less than 10 is considered as highly accurate forecasting. Thus, the prediction of the physical properties by model B is considered to be excellent. Besides that, the predictions of the elastic properties which are longitudinal velocity, shear velocity, longitudinal modulus, shear modulus, bulk modulus, young modulus, microhardness, and fractal bond connectivity, Poisson ratio are considered to be excellent as they range from 0.170 % to 2.164% thus indicating that model B predicts their values efficiently. Besides that, the MAPE value for the optical property, the optical bandgap is 0.2455% which is also accurate forecasting. The overall MAPE of model B is calculated to be 1.053% which is a highly accurate forecasting.

4.4 Comparison between models

In this section, all the two models will be compared and analysed based on prediction accuracy and their performance metrics will be discussed. Furthermore, the model with the best performance metric will be selected as the official model for this project. The table 4.16 summarizes the metrices of the two models A and B.

Table 4.16: Comparison between model A and B

Model / Parameters	A	B
Training Loss	4.356	27.166
MAPE Error	15.028%	1.053%
No of negative R-squared parameters	3	0

The Model' A training loss is lower compared to Model B. Thus, this indicates that model A fits the training data well and has a lower error compared to model B. Model B has the highest training loss which is 27.166 and has a higher error on the training data among the two models which may lead to the overfitting of the data. Furthermore, the percentage error for Model A is the highest which is 15.028% but it is still considered as a good forecasting as it is in the range of (10-20) % according to the interpretation table. Model B has the lowest MAPE error which is 1.053% which is an indication of accurate forecasting. Furthermore, there are several negative R-squared values which are present in Model A which indicates that there is no correlation between the parameters and variables to be predicted while there are no negative R-squared present in Model B.

This case could be explained by the relationship of the activation functions and data that is used in the deep learning model. Based on the graphs plotted it can be observed that the negative R-squared value occurs during the prediction of the optical bandgap, fractal bond connectivity, and Poisson's ratio. By referring to the data set of the training values provided it can be observed that the values of the parameters of these three parameters are near zero, hence the ReLU activation function might not be an appropriate activation function to predict the values of these parameters because it is not a zero centric function (Ali, 2023). Thus, it is difficult for the ReLU function to predict numbers that are closer to zero. On the other hand, the Tanh function which is used in Model B is more efficient in dealing with numbers nearer to zero as they have an output centered around zero which is beneficial for optimization. Thus, based on reviewing the two models, model B is more suited for the prediction of elastic and optical properties of binary glass systems as it has an upper hand in its performance metrics. However, due to its large training loss error, overfitting might occur. Hence, more dataset is required to prevent overfitting occurring in this model.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The project prediction of elastic and optical properties of binary glass systems using artificial intelligence approach which aims to develop an artificial neural network using Python software to predict elastic and optical properties of some binary borate compounds and tellurite glass systems has been conducted successfully.

Two deep learning models, models A and B were developed and tested to predict and simulate the elastic and optical properties of the binary glass systems. Thus, several evaluation metrics was used to test the accuracy of each model. The analysis of each model and its advantages and drawbacks were discussed. Furthermore, each of the model's predicted outputs was plotted and compared with each other. After that, performance of evaluation metrics on each model was done and model B was found to be the most accurate in predicting the parameter of the binary glasses.

In a nutshell, despite the ANN model not being tested with real data, the deep learning model's performance was assessed, and the findings were satisfactory. However, the ANN model still has significant limitations and some constraints that needs to be noted and discussed. Furthermore, some other improvements and changes can be conducted so that this deep learning model can be implemented in the manufacturing industry in the future.

5.2 Limitations and Recommendations

The lack of data for deep learning training can pose a lot of challenges and limitations. One of the problems that can occur is overfitting where the model tends to memorize the training data sets rather than learning generalizable patterns hence affecting poor performance. Furthermore, due to the lack of data, there is an inability to conduct validation and hyperparameter tuning. Thus, several other methods such as cross validation, K-fold might be ineffective. On the other hand, deep learning models have high capacity to learn complex patterns from data. However, this capacity may be underutilized with a small dataset leading to a poor performance. In addition, with sufficient data, the train test split for evaluating the ANN can be conducted. The train-test split procedure is a deep learning technique used to evaluate the performance of the deep learning model when they do predictions on data that is not used to train the model. Thus, this procedure allows us to make comparisons about the performance of the deep learning model and allows us to adjust the parameters accordingly to get a better output from the deep learning model.

In conclusion, to overcome the drawbacks discussed above and to increase the efficiency of the ANN model for predicting the elastic and optical properties of binary glass systems to perform with real-world and unseen data, the recommendations stated above shall be conducted in the future to achieve a more optimal result.

REFERENCES

Anon, (2020). *Young's modulus or longitudinal modulus of elasticity - Servosis*. [online] Available at: https://www.servosis.com/en/el-modulo-de-young-o-modulo-de-elasticidad-longitudinal/.[Acessed 21 April 2024]

Banerjee, S., Seth, S., Dey, T., Pal, D. and Student (n.d.). *PYTHON PROGRAMMING LANGUAGE AND ITS SCOPE IN FUTURE*. [online] *International Research Journal of Modernization in Engineering Technology and Science*, Peer-Reviewed, Open Access, pp.2582–5208. Available at: https://www.irjmets.com/uploadedfiles/paper//issue_8_august 2022/29067/ final/fin_irjmets1659536271.pdf. [Accessed 21 April 2024]

Barrett, P., Hunter, J., Miller, J.T., Hsu, J.-C. and Greenfield, P. (2005). matplotlib – A Portable Python Plotting Package. In: *ASTRONOMICAL DATA ANALYSIS SOFTWARE AND SYSTEMS XIV*.

BYJU'S (2024). *Unit of Density - Density Definition, SI unit, Solved Examples*. [online] BYJUS. Available at: https://byjus.com/physics/unit-of-density/#:~:text=Density%20Definition%3A%20Density%20is%20the. [Accessed 21 April 2024].

BYJU'S (n.d.). *Percent Error - Definition, Formula, and Solved examples*. [online] BYJUS. Available at: https://byjus.com/maths/percent-error/#:~:text=Percent%20error%20is%20the%20difference. [Accessed 23 April 2024].

BYJUS. (n.d.). *Shear Modulus (Modulus Of Rigidity) - Definition, Formula, Units, Examples*. [online] Available at: https://byjus.com/physics/shear-modulus-elastic-moduli/#:~:text=Shear%20modulus%2C%20also%20known%20as. [Accessed 21 April 2024].

Chemistry LibreTexts. (2016). *3.5: Differences in Matter- Physical and Chemical Properties*. [online] Available at: https://chem.libretexts.org/Bookshelves/Introductory_Chemistry/Introductory_Chemistry/03%3A_Matter_and_Energy/3.05%3A_Differences_in_Matter-_Physical_and_Chemical_Properties#:~:text=Summary-. [Accessed 20 April 2024].

Condurache-Bota, S., 2017. *intechopen.* [Online] Available at: http://dx.doi.org/10.5772/intechopen.7510 [Accessed 17 April 2024].

David L. Griscom (25 March 1991). Optical Properties and Structure of Defects in Silica Glass. In *Naval Research Laboratory.*

D. B. Beniz and A. M. Espindola, (2016). USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS. In: *Brazilian Synchrotron Light Laboratory*.

Dongare, A., Kharde, R. and Kachare, A. (2008). Introduction to Artificial Neural Network. *Certified International Journal of Engineering and Innovative Technology (IJEIT)*, [online] 9001(1), pp.2277–3754. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06.[Accessed 17 April 2024].

Effendy, N., Hj Ab Aziz, S., Mohamed Kamari, H., Mohd Zaid, M.H. and Wahab, S.A. (2020). Ultrasonic and artificial intelligence approach: Elastic behavior on the influences of ZnO in tellurite glass systems. *Journal of Alloys and Compounds*, 835.

Houcque, D. (2007). *INTRODUCTION TO MATLAB FOR ENGINEERING STUDENTS*. [online] Available at: https://www.mccormick.northwestern.edu/documents/students/undergraduate/introduction-to-matlab.pdf. [Accessed 22 April 2024].

Hussain, H. (2019). *Data Science: Explaining R^2 in Statistics*. [online] Medium. Available at: https://towardsdatascience.com/data-science-explaining-r%C2%B2-in-statistics-6f34e7f0a9bb.[Accessed 23 April 2024].

IBM (2023). *What is Deep Learning?* [online] www.ibm.com. Available at: https://www.ibm.com/topics/deep-learning. [Accessed 19 April 2024].

Kaijanaho, A.-J. (n.d.). *Evidence-Based Programming Language Design A Philosophical and Methodological Exploration*. [online] Available at: https://jyx.jyu.fi/bitstream/handle/123456789/47698/1/978-951-39-6388-0_vaitos04122015.pdf [Accessed 22 April 2024].

Khan, Z., Tasnim, S., Alin and Hussain, M. (2011). Price Prediction of Share Market using Artificial Neural Network (ANN). *International Journal of Computer Applications*, [online] 22(2), pp.975–8887. Available at: https://www.ijcaonline.org/volume22/number2/pxc3873497.pdf [Accessed 22 April 2024].

Kim, S. and Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 669-679.

Mckinney, W. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics.

Moez Ali (2023). *Introduction to Activation Functions in Neural Networks-Radar* [online] Available at : https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks. [Accessed 19 April 2024].

Özgür, Ü., Hofstetter, D. and Morkoç, H. (2010). ZnO Devices and Applications: A Review of Current Status and Future Prospects. *Proceedings of the IEEE*, 98(7), pp.1255–1268. doi:https://doi.org/10.1109/jproc.2010.2044550.

Pedregosa, F., Gael Varoquaux, G., Gramfort, A., Michel, V. and Thirion , B. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research 12*, 2825-2830.

Rada, S., Culea, E., Rada, M. *et al.* Structural and electronic properties of tellurite glasses. *J Mater Sci* **44**, 3235–3240 (2009). https://doi.org/10.1007/s10853-009-3433-8

Sharma, S., Sharma, S. and Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology, 2020* , [online] 4(12), pp.310–316. Available at: http://www.ijeast.com/. [Accessed 26 April 2024].

Team, T. (2020). *Artificial Neural Network - Applications, Algorithms and Examples*. [online] TechVidvan. Available at: https://techvidvan.com/tutorials/artificial-neural-network/. [Accessed 23 April 2024].

Terven, J.R., Cordova-Esparza, D.M., Perdraza, A.R. and Chavez-Urbiola, E.A. (2023). LOSS FUNCTIONS AND METRICS IN DEEP LEARNING. A REVIEW. *UNDER REVIEW IN COMPUTER SCIENCE REVIEW*.

APPENDICES

APPENDIX A: Computer Code Model B

```
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
import tkinter as tk
from tkinter import ttk
class Binary_Glass_Simulator():

    def click(self):
        self.alpha_grid.destroy()
        self.Model_Training()

    def click2(self, event=None):
        self.beta_grid.destroy()
        self.Model_Training(event)

    def Model_Training(self, event = None):
        self.training_data = 'C:\\Users\\Prasad
Soundrarajan\\Desktop\\ANN\\Training Dataset.xlsx'
        self.Dataset_Training = pd.read_excel(self.training_data)

        # Initializing input & output data for training set
        self.Binary_Compositions_Training = self.Dataset_Training.iloc[:, 0:4]  #
input data
        self.Parameters_Training = self.Dataset_Training.iloc[:, 4:16]  # output data

        # Converting to tensor
        self.x_train =
torch.tensor(self.Binary_Compositions_Training.values).float()
        self.y_train = torch.tensor(self.Parameters_Training.values).float()

        # Building ANN model
        ANN_Model = nn.Sequential(
```

```
            nn.Linear(4, 4000),
            nn.Tanh(),
            nn.Linear(4000, 12),


# Learning Rate
        learning_rate = 0.001

        # Loss Rate
        loss_function = nn.MSELoss()

        # optimizer
        optimizer = torch.optim.Adam(ANN_Model.parameters(), lr=learning_rate)

        # Initializing the training epochs
        self.numepochs = 12000
        self.losses = torch.zeros(self.numepochs)

        # Model Training
        for epochi in range(self.numepochs):
            self.yHat = ANN_Model(self.x_train)

            # Compute Loss
            self.loss = loss_function(self.yHat, self.y_train)
            self.losses[epochi] = self.loss

            # Backpropagation
            optimizer.zero_grad()
            self.loss.backward()
            optimizer.step()

self.Training_Model_Evaluation_Grid()

    def Training_Model_Evaluation_Grid(self):
        self.beta_grid = tk.Tk()
        self.beta_grid.title("Training Model Evaluation")
        self.beta_grid.geometry('450x550')

Label_beta = tk.Label(self.beta_grid, text="Training Model Evaluation",
                          font=('Helvetica', 15, 'bold'), fg='blue4')
        button_plot_TVE = tk.Button(self.beta_grid, text='Graph of Losses'
                    ,command=self.plot_TVE, width=33, padx=25,
pady=10,font=('Helvetica', 15, 'bold'), bg='cyan')
        button_plot_RVP = tk.Button(self.beta_grid, text='Predicted vs Actual Data'
                    ,command=self.plot_RVP, width=33, padx=25,
pady=10,font=('Helvetica', 15, 'bold'), bg='cyan')
```

```
button_plot_R2 = tk.Button(self.beta_grid, text='Plot of R2 graph'
                ,command=self.plot_R2, width=33, padx=25,
pady=10,font=('Helvetica', 15, 'bold'), bg='cyan')
        button_retrain = tk.Button(self.beta_grid, text='Re-Train Model'
                    , command=self.click2, width=33, padx=25,
pady=10,font=('Helvetica', 15, 'bold'), bg='cyan')
        button_compare = tk.Button(self.beta_grid, text='Real Data Set'
              , command= self.compare_real, width=33, padx=25,pady=10,
font=('Helvetica', 15, 'bold'), bg='cyan')
        button_next = tk.Button(self.beta_grid, text='Predicted Data'
              , command= self.compare_pred, width=33, padx=25,
pady=10,font=('Helvetica', 15, 'bold'), bg='cyan')


Label_beta.grid(column=1, row=0, padx=(10, 10), pady=(10, 5))
        button_plot_TVE.grid(column=1, row=1, pady=(20, 20))
        button_plot_TVE.bind('<Return>', self.plot_TVE)
        button_plot_RVP.grid(column=1, row=2, pady=(10, 10))
        button_plot_RVP.bind('<Return>', self.plot_RVP)
        button_plot_R2.grid(column=1, row=4, pady=(10, 10))
        button_plot_R2.bind('<Return>', self.plot_R2)
        button_retrain.grid(column=1, row=5, pady=(10, 10))
        button_retrain.bind('<Return>', self.click2)
        button_compare.grid(column=1, row=6, pady=(10, 10))
        button_compare.bind('<Return>', self.compare_real)
        button_next.grid(column=1, row=7, pady=(10, 10))
       # button_next.bind('<Return>', self.Binary_Simulator)
        self.beta_grid.mainloop()


def plot_TVE(self):
        # Plot of losses
        plt.figure(figsize=(10, 5))
        plt.plot(self.losses.detach().numpy())  # Convert losses tensor to NumPy
array for plotting
        plt.xlabel('Epoch')
        plt.ylabel('Training Loss')
        plt.title(f'Training Loss over Epochs: {self.losses[-1].item()}')
        plt.grid(True)  # Add grid for better visualization
        plt.show()

def plot_RVP(self):
        # Plot of Real vs Predicted data ( training )
```

```
    # Density , Molar Volume , Longitudinal velocity , Shear Velocity
      self.Density_Training = self.Parameters_Training['Density, ρ (g/cm3)']
      self.Density_Prediction = self.yHat[:, 0]

      self.MolarV_Training = self.Parameters_Training['Molar Volume, Vm
(cm3/mol)']
      self.MolarV_Prediction = self.yHat[:, 1]

      self.LongitudinalV_Training = self.Parameters_Training['Longitudinal
velocity, VL (m/s)']
      self.LongitudinalV_Prediction = self.yHat[:, 2]

      self.ShearV_Training = self.Parameters_Training['Shear velocity, VL (m/s)']
      self.ShearV_Prediction = self.yHat[:, 3]

      fig, ax = plt.subplots(2, 2, figsize=(10, 5))

      # Plot Density
      ax[0, 0].plot(self.Density_Training.values, 'ro', label='ρ$_{exp}$
(g/cm$^3$)')
      ax[0, 0].plot(self.Density_Prediction.detach().numpy(), 'bo',
label='ρ$_{ANN}$ (g/cm$^3$)')
      ax[0, 0].set_title('Predicted values over Real Data (Density)')
      ax[0, 0].legend()

      # Plot Molar Volume
      ax[0, 1].plot(self.MolarV_Training.values, 'ro', label='V$_{m,exp}$
(cm$^3$/mol)')
      ax[0, 1].plot(self.MolarV_Prediction.detach().numpy(), 'bo',
label='V$_{m,ANN}$ (cm$^3$/mol)')
      ax[0, 1].set_title('Predicted values over Real Data (Molar Volume)')
      ax[0, 1].legend()

  # Plot Longitudinal Velocity
      ax[1, 0].plot(self.LongitudinalV_Training.values, 'ro', label='VL_exp (m/s)')
      ax[1, 0].plot(self.LongitudinalV_Prediction.detach().numpy(), 'bo',
label='VL_ANN (m/s)')
      ax[1, 0].set_title('Predicted values over Real Data (Longitudinal Velocity)')
      ax[1, 0].legend()
```

```
# Plot Shear Velocity
        ax[1, 1].plot(self.ShearV_Training.values, 'ro', label='VL_exp (m/s)')
        ax[1, 1].plot(self.ShearV_Prediction.detach().numpy(), 'bo', label='VL_ANN
(m/s)')
        ax[1, 1].set_title('Predicted values over Real Data (Shear Velocity)')
        ax[1, 1].legend()

        plt.tight_layout()
        plt.show()

        # Longitudinal modulus, Shear modulus, Bulk modulus, Young modulus
        self.LongitudinalM_Training = self.Parameters_Training['Longitudinal
modulu, L']
        self.LongitudinalM_Prediction = self.yHat[:, 4]

        self.ShearM_Training = self.Parameters_Training['Shear modulus,G']
        self.ShearM_Prediction = self.yHat[:, 5]

        self.BulkM_Training = self.Parameters_Training['Bulk modulus, K']
        self.BulkM_Prediction = self.yHat[:, 6]

        self.YoungM_Training = self.Parameters_Training['Young modulus, E']
        self.YoungM_Prediction = self.yHat[:, 7]

        fig, ax = plt.subplots(2, 2, figsize=(10, 5))

        # Plot Longitudinal modulus
        ax[0, 0].plot(self.LongitudinalM_Training.values, 'ro', label='L_exp')
        ax[0, 0].plot(self.LongitudinalM_Prediction.detach().numpy(), 'bo',
label='L_ANN')
        ax[0, 0].set_title('Predicted values over Real Data (Longitudinal modulus)')
        ax[0, 0].legend()

    # Plot Shear modulus
        ax[0, 1].plot(self.ShearM_Training.values, 'ro', label='G_exp')
        ax[0, 1].plot(self.ShearM_Prediction.detach().numpy(), 'bo',
label='G_ANN')
        ax[0, 1].set_title('Predicted values over Real Data (Shear modulus)')
        ax[0, 1].legend()
```

```
# Plot Shear modulus
        ax[0, 1].plot(self.ShearM_Training.values, 'ro', label='G_exp')
        ax[0, 1].plot(self.ShearM_Prediction.detach().numpy(), 'bo',
label='G_ANN')
        ax[0, 1].set_title('Predicted values over Real Data (Shear modulus)')
        ax[0, 1].legend()

        # Plot Bulk modulus
        ax[1, 0].plot(self.BulkM_Training.values, 'ro', label='K_exp')
        ax[1, 0].plot(self.BulkM_Prediction.detach().numpy(), 'bo', label='K_ANN')
        ax[1, 0].set_title('Predicted values over Real Data (Bulk Modulus)')
        ax[1, 0].legend()

        # Plot Young modulus
        ax[1, 1].plot(self.YoungM_Training.values, 'ro', label='E_exp')
        ax[1, 1].plot(self.YoungM_Prediction.detach().numpy(), 'bo',
label='E_ANN')
        ax[1, 1].set_title('Predicted values over Real Data (Young Modulus)')
        ax[1, 1].legend()

        plt.tight_layout()
        plt.show()

        # Microhardness, fractal bond connectivity, d, Poisson ratio, σ, Optical
bandgap (eV)
        self.Microhardness_Training = self.Parameters_Training['Microhardness, H
(GPa)']
        self.Microhardness_Prediction = self.yHat[:, 8]

        self.fractalB_Training = self.Parameters_Training['fractal bond connectivity,
d']
        self.fractalB_Prediction = self.yHat[:, 9]

        self.Poisson_Training = self.Parameters_Training['Poisson ratio, σ']
        self.Poisson_Prediction = self.yHat[:, 10]

        self.OpticalB_Training = self.Parameters_Training['Optical bandgap (eV)']
        self.OpticalB_Prediction = self.yHat[:, 11]

        fig, ax = plt.subplots(2, 2, figsize=(10, 5))
```

```
# Plot Microhardness
        ax[0, 0].plot(self.Microhardness_Training.values, 'ro', label='H_exp (GPa)')
        ax[0, 0].plot(self.Microhardness_Prediction.detach().numpy(), 'bo',
label='H_ANN (GPa)')
        ax[0, 0].set_title('Predicted values over Real Data (Microhardness)')
        ax[0, 0].legend()

        # fractal bond connectivity, d
        ax[0, 1].plot(self.fractalB_Training.values, 'ro', label='d_exp')
        ax[0, 1].plot(self.fractalB_Prediction.detach().numpy(), 'bo', label='d_ANN')
        ax[0, 1].set_title('Predicted values over Real Data (fractal bond
connectivity)')
        ax[0, 1].legend()

        # Plot Poisson ratio, σ
        ax[1, 0].plot(self.Poisson_Training.values, 'ro', label='σ_exp')
        ax[1, 0].plot(self.Poisson_Prediction.detach().numpy(), 'bo', label='σ_ANN')
        ax[1, 0].set_title('Predicted values over Real Data (Poisson ratio)')
        ax[1, 0].legend()

        # Plot Optical bandgap (eV)
        ax[1, 1].plot(self.OpticalB_Training.values, 'ro', label='eV_exp')
        ax[1, 1].plot(self.OpticalB_Prediction.detach().numpy(), 'bo',
label='eV_ANN')
        ax[1, 1].set_title('Predicted values over Real Data (Optical Bandgap)')
        ax[1, 1].legend()

        plt.tight_layout()
        plt.show()


def plot_R2(self):
        r2_density = r2_score(self.Density_Training,
self.Density_Prediction.detach().numpy())
        coefficients_density = np.polyfit(self.Density_Prediction.detach().numpy(),
self.Density_Training,1)
        # Fit a 1st degree polynomial (a line)n
        best_fit_line_1 = np.poly1d(coefficients_density)
        plt.plot(self.Density_Prediction.detach().numpy(),
```

```
 best_fit_line_1(self.Density_Prediction.detach().numpy()), 'r',label="Best Fit
Line")
      plt.scatter(self.Density_Prediction.detach().numpy(), self.Density_Training,
label='Data Points')
      plt.xlabel("Density Predicted")
      plt.ylabel("Density Real")
      plt.title(f"R2 of Density = {r2_density}")
      plt.legend()
      plt.show()

      # Plot of R2 Molar Volume
      r2_density = r2_score(self.MolarV_Training,
self.MolarV_Prediction.detach().numpy())
      coefficients_density = np.polyfit(self.MolarV_Prediction.detach().numpy()
                                 , self.MolarV_Training,1)  # Fit a 1st degree
polynomial (a line)n
      best_fit_line_2 = np.poly1d(coefficients_density)
      plt.plot(self.MolarV_Prediction.detach().numpy()
            , best_fit_line_2(self.MolarV_Prediction.detach().numpy()),
'r',label="Best Fit Line")
      plt.scatter(self.MolarV_Prediction.detach().numpy(), self.MolarV_Training,
label='Data Points')
      plt.xlabel("Molar Volume Predicted")
      plt.ylabel("Molar Volume Real")
      plt.title(f"R2 of Molar Volume = {r2_density}")
      plt.legend()
      plt.show()
# Plot of R2 Longitudinal Velocity
      r2_LongitudinalV = r2_score(self.LongitudinalV_Training,
self.LongitudinalV_Prediction.detach().numpy())
      coefficients_LongitudinalV =
np.polyfit(self.LongitudinalV_Prediction.detach().numpy()
                           , self.LongitudinalV_Training,1)  # Fit a 1st degree
polynomial (a line)n
      best_fit_line_3 = np.poly1d(coefficients_LongitudinalV)
      plt.plot(self.LongitudinalV_Prediction.detach().numpy()
            , best_fit_line_3(self.LongitudinalV_Prediction.detach().numpy()),'r',
label="Best Fit Line")
      plt.scatter(self.LongitudinalV_Prediction.detach().numpy()
              , self.LongitudinalV_Training, label='Data Points')
      plt.xlabel("Longitudinal Velocity  Predicted")
      plt.ylabel("Longitudinal Velocity Real")
      plt.title(f"R2 of Longitudinal Velocity = {r2_LongitudinalV}")
      plt.legend()
      plt.show()
```

```
   # Plot of R2 Shear Velocity
      r2_ShearV = r2_score(self.ShearV_Training,
self.ShearV_Prediction.detach().numpy())
      coefficients_ShearV = np.polyfit(self.ShearV_Prediction.detach().numpy()
                              , self.ShearV_Training,1)  # Fit a 1st degree polynomial
(a line)n
      best_fit_line_4 = np.poly1d(coefficients_ShearV)
      plt.plot(self.ShearV_Prediction.detach().numpy()
            , best_fit_line_4(self.ShearV_Prediction.detach().numpy()),
'r',label="Best Fit Line")
      plt.scatter(self.ShearV_Prediction.detach().numpy(), self.ShearV_Training,
label='Data Points')
      plt.xlabel("Shear Velocity Predicted")
      plt.ylabel("Shear Velocity Real")
      plt.title(f"R2 of Shear Velocity = {r2_ShearV}")
      plt.legend()
      plt.show()

      # Plot of R2 Longitudinal Modulus
      r2_LongitudinalM = r2_score(self.LongitudinalM_Training,
self.LongitudinalM_Prediction.detach().numpy())
      coefficients_LongitudinalM =
np.polyfit(self.LongitudinalM_Prediction.detach().numpy()
                        , self.LongitudinalM_Training,1)  # Fit a 1st degree
polynomial (a line)n
      best_fit_line_5 = np.poly1d(coefficients_LongitudinalM)
      plt.plot(self.LongitudinalM_Prediction.detach().numpy()
            , best_fit_line_5(self.LongitudinalM_Prediction.detach().numpy()),'r',
label="Best Fit Line")
      plt.scatter(self.LongitudinalM_Prediction.detach().numpy(),
self.LongitudinalM_Training, label='Data Points')
      plt.xlabel("Longitudinal Modulus Predicted")
      plt.ylabel("Longitudinal Modulus Real")
      plt.title(f"R2 of Longitudinal Modulus = {r2_LongitudinalM}")
      plt.legend()
      plt.show()

  # Plot of R2 Shear Modulus
      r2_ShearM = r2_score(self.ShearM_Training,
self.ShearM_Prediction.detach().numpy())
      coefficients_ShearM = np.polyfit(self.ShearM_Prediction.detach().numpy(),
self.ShearM_Training, 1)
      # Fit a 1st degree polynomial (a line)n
      best_fit_line_6 = np.poly1d(coefficients_ShearM)
      plt.plot(self.ShearM_Prediction.detach().numpy()
```

```
        , best_fit_line_6(self.ShearM_Prediction.detach().numpy()), 'r',label="Best
Fit Line")
        plt.scatter(self.ShearM_Prediction.detach().numpy(), self.ShearM_Training,
label='Data Points')
        plt.xlabel("Shear Modulus Predicted")
        plt.ylabel("Shear Modulus Real")
        plt.title(f"R2 of Shear Modulus = {r2_ShearM}")
        plt.legend()
        plt.show()

        # Plot of R2 Bulk Modulus
        r2_BulkM = r2_score(self.BulkM_Training,
self.BulkM_Prediction.detach().numpy())
        coefficients_BulkM = np.polyfit(self.BulkM_Prediction.detach().numpy(),
self.BulkM_Training,1)
        # Fit a 1st degree polynomial (a line)n
        best_fit_line_7 = np.poly1d(coefficients_BulkM)
        plt.plot(self.BulkM_Prediction.detach().numpy()
                , best_fit_line_7(self.BulkM_Prediction.detach().numpy()),
'r',label="Best Fit Line")
        plt.scatter(self.BulkM_Prediction.detach().numpy(), self.BulkM_Training,
label='Data Points')
        plt.xlabel("Bulk Modulus Predicted")
        plt.ylabel("Bulk Modulus Real")
        plt.title(f"R2 of Bulk Modulus = {r2_BulkM}")
        plt.legend()
        plt.show()

        # Plot of R2 Young Modulus
        r2_YoungM = r2_score(self.YoungM_Training,
self.YoungM_Prediction.detach().numpy())
        coefficients_YoungM =
np.polyfit(self.YoungM_Prediction.detach().numpy(), self.YoungM_Training, 1)
        # Fit a 1st degree polynomial (a line)n
        best_fit_line_8 = np.poly1d(coefficients_YoungM)
        plt.plot(self.YoungM_Prediction.detach().numpy()
                , best_fit_line_8(self.YoungM_Prediction.detach().numpy()), 'r',
label="Best Fit Line")
        plt.scatter(self.YoungM_Prediction.detach().numpy(),
self.YoungM_Training, label='Data Points')
        plt.xlabel("Young Modulus Predicted")
        plt.ylabel("young Modulus Real")
        plt.title(f"R2 of Young Modulus = {r2_YoungM}")
        plt.legend()
        plt.show()
```

```
,
        # Plot of R2 Microhardness
        r2_Microhardness = r2_score(self.Microhardness_Training,
self.Microhardness_Prediction.detach().numpy())
        coefficients_Microhardness =
np.polyfit(self.Microhardness_Prediction.detach().numpy()
                        , self.Microhardness_Training,1)  # Fit a 1st degree
polynomial (a line)n
        best_fit_line_9 = np.poly1d(coefficients_Microhardness)
        plt.plot(self.Microhardness_Prediction.detach().numpy()
              , best_fit_line_9(self.Microhardness_Prediction.detach().numpy()),'r',
label="Best Fit Line")
        plt.scatter(self.Microhardness_Prediction.detach().numpy(),
self.Microhardness_Training, label='Data Points')
        plt.xlabel("Microhardness Predicted")
        plt.ylabel("Microhardness Real")
        plt.title(f"R2 of Microhardness = {r2_Microhardness}")
        plt.legend()
        plt.show()

        # Plot of R2 Fractal Bond Connectivity
        r2_fractalB = r2_score(self.fractalB_Training,
self.fractalB_Prediction.detach().numpy())
        coefficients_fractalB = np.polyfit(self.fractalB_Prediction.detach().numpy()
                        , self.fractalB_Training,1)  # Fit a 1st degree
polynomial (a line)n
        best_fit_line_10 = np.poly1d(coefficients_fractalB)
        plt.plot(self.fractalB_Prediction.detach().numpy()
              , best_fit_line_10(self.fractalB_Prediction.detach().numpy()),
'r',label="Best Fit Line")
        plt.scatter(self.fractalB_Prediction.detach().numpy(), self.fractalB_Training,
label='Data Points')
        plt.xlabel("Fractal Bond Connectivity Predicted")
        plt.ylabel("Fractal Bond Connectivity Real")
        plt.title(f"R2 of Fractal Bond Connectivity = {r2_fractalB}")
        plt.legend()
        plt.show()
# Plot of R2 Poisson Ratio
        r2_Poisson = r2_score(self.Poisson_Training,
self.Poisson_Prediction.detach().numpy())
        coefficients_Poisson = np.polyfit(self.Poisson_Prediction.detach().numpy()
                        ,
, best_fit_line_11(self.Poisson_Prediction.detach().numpy()), 'r',label="Best Fit
Line")
```

```
plt.scatter(self.Poisson_Prediction.detach().numpy(), self.Poisson_Training,
label='Data Points')
        plt.xlabel("Poisson Ratio Predicted")
        plt.ylabel("Poisson Ratio Real")
        plt.title(f"R2 of Poisson Ratio = {r2_Poisson}")
        plt.legend()
        plt.show()

        # Plot of R2 Optical Bandgap
        r2_OpticalB = r2_score(self.OpticalB_Training,
self.OpticalB_Prediction.detach().numpy())
        coefficients_OpticalB =
np.polyfit(self.OpticalB_Prediction.detach().numpy()
                                , self.OpticalB_Training,1)  # Fit a 1st degree
polynomial (a line)n
        best_fit_line_12 = np.poly1d(coefficients_OpticalB)
        plt.plot(self.OpticalB_Prediction.detach().numpy()
                , best_fit_line_12(self.OpticalB_Prediction.detach().numpy()),
'r',label="Best Fit Line")
        plt.scatter(self.OpticalB_Prediction.detach().numpy(),
self.OpticalB_Training, label='Data Points')
        plt.xlabel("Optical Bandgap Predicted")
        plt.ylabel("Optical Bandgap  Real")
        plt.title(f"R2 of Optical Bandgap = {r2_OpticalB}")
        plt.legend()
        plt.show()

    def compare_real(self):
        self.gamma_grid = tk.Tk()
        self.gamma_grid.title("Comparison Between Real & Predicted Values")
        self.gamma_grid.geometry('1080x200')

        comparison_table = ttk.Treeview(self.gamma_grid)

        comparison_table['columns'] = (
            'ZnO', 'Bi2O3', 'TeO2', 'B2O3', 'Density, ρ (g/cm3)', 'Molar Volume, Vm
(cm3/mol)',
            'Longitudinal velocity, VL (m/s)', 'Shear velocity, VL (m/s)', 'Longitudinal
modulu, L',
            'Shear modulus,G', 'Bulk modulus, K', 'Young modulus, E',
'Microhardness, H (GPa)',
            'fractal bond connectivity, d', 'Poisson ratio, σ', 'Optical bandgap (eV)')

        comparison_table.column("#0", width=20, minwidth=25,stretch=tk.NO)
```

```
comparison_table.column("ZnO", anchor=tk.CENTER, width=100,
minwidth=25)
      comparison_table.column("Bi2O3", anchor=tk.CENTER, width=100,
minwidth=25)
      comparison_table.column("TeO2", anchor=tk.CENTER, width=100,
minwidth=25)
      comparison_table.column("B2O3", anchor=tk.CENTER, width=100,
minwidth=25)
      comparison_table.column("Density, ρ (g/cm3)", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Molar Volume, Vm (cm3/mol)",
anchor=tk.CENTER, width=120, minwidth=25)
      comparison_table.column("Longitudinal velocity, VL (m/s)",
anchor=tk.CENTER, width=120, minwidth=25)
      comparison_table.column("Shear velocity, VL (m/s)", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Longitudinal modulu, L", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Shear modulus,G", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Bulk modulus, K", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Young modulus, E", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Microhardness, H (GPa)", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("fractal bond connectivity, d",
anchor=tk.CENTER, width=120, minwidth=25)
      comparison_table.column("Poisson ratio, σ", anchor=tk.CENTER,
width=120, minwidth=25)
      comparison_table.column("Optical bandgap (eV)", anchor=tk.CENTER,
width=120, minwidth=25)

      comparison_table.heading("#0", text="", anchor=tk.CENTER)
      comparison_table.heading("ZnO", text="ZnO", anchor=tk.CENTER)
      comparison_table.heading("Bi2O3", text="Bi2O3", anchor=tk.CENTER)
      comparison_table.heading("TeO2", text="TeO2", anchor=tk.CENTER)
      comparison_table.heading("B2O3", text="B2O3", anchor=tk.CENTER)
      comparison_table.heading("Density, ρ (g/cm3)", text="Density, ρ (g/cm3)",
anchor=tk.CENTER)
      comparison_table.heading("Molar Volume, Vm (cm3/mol)", text="Molar
Volume, Vm (cm3/mol)"
                        , anchor=tk.CENTER)
```

```
                ,
                        , anchor=tk.CENTER)
        comparison_table.heading("Longitudinal velocity, VL (m/s)",
text="Longitudinal velocity"
                                                    ", VL (m/s)",anchor=tk.CENTER)
        comparison_table.heading("Shear velocity, VL (m/s)", text="Shear velocity,
VL (m/s)", anchor=tk.CENTER)
        comparison_table.heading("Longitudinal modulu, L", text="Longitudinal
modulu, L", anchor=tk.CENTER)
        comparison_table.heading("Shear modulus,G", text="Shear modulus,G",
anchor=tk.CENTER)
        comparison_table.heading("Bulk modulus, K", text="Bulk modulus",
anchor=tk.CENTER)
        comparison_table.heading("Young modulus, E", text="Young modulus, E",
anchor=tk.CENTER)
        comparison_table.heading("Microhardness, H (GPa)", text="Microhardness,
H (GPa)", anchor=tk.CENTER)
        comparison_table.heading("fractal bond connectivity, d", text="fractal bond
connectivity", anchor=tk.CENTER)
        comparison_table.heading("Poisson ratio, σ", text="Poisson ratio, σ",
anchor=tk.CENTER)
        comparison_table.heading("Optical bandgap (eV)", text="Optical bandgap
(eV)", anchor=tk.CENTER)

        comparison_table.insert(parent='', index='end', iid=0, text="
                        , values=(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0))
for i in range(1, 19):
            comparison_table.insert(parent='', index='end', iid=i, text='', values=(
                self.x_train[i - 1][0].numpy(), self.x_train[i - 1][1].numpy()
                , self.x_train[i - 1][2].numpy(),self.x_train[i - 1][3].numpy(),
                self.y_train[i - 1][0].numpy(), self.y_train[i - 1][1].numpy()
                , self.y_train[i - 1][2].numpy(),self.y_train[i - 1][3].numpy(),
                self.y_train[i - 1][4].numpy(), self.y_train[i - 1][5].numpy()
                , self.y_train[i - 1][6].numpy(),self.y_train[i - 1][7].numpy(),
                self.y_train[i - 1][8].numpy(), self.y_train[i - 1][9].numpy()
                , self.y_train[i - 1][10].numpy(),self.y_train[i - 1][11].numpy())),

comparison_table.pack(pady=20)
        self.gamma_grid.mainloop()
```

```
def compare_pred(self):
        self.delta_grid = tk.Tk()
        self.delta_grid.title("Comparison Between Real & Predicted Values")
        self.delta_grid.geometry('1080x200')

        comparison_table2 = ttk.Treeview(self.delta_grid)

        comparison_table2['columns'] = (
            'ZnO', 'Bi2O3', 'TeO2', 'B2O3', 'Density, ρ (g/cm3)', 'Molar Volume, Vm
(cm3/mol)',
            'Longitudinal velocity, VL (m/s)', 'Shear velocity, VL (m/s)', 'Longitudinal
modulu, L',
            'Shear modulus,G', 'Bulk modulus, K', 'Young modulus, E',
'Microhardness, H (GPa)',
            'fractal bond connectivity, d', 'Poisson ratio, σ', 'Optical bandgap (eV)')

comparison_table2.column("#0", width=20, minwidth=25,stretch=tk.NO)
        comparison_table2.column("ZnO", anchor=tk.CENTER, width=100,
minwidth=25)
        comparison_table2.column("Bi2O3", anchor=tk.CENTER, width=100,
minwidth=25)
        comparison_table2.column("TeO2", anchor=tk.CENTER, width=100,
minwidth=25)
        comparison_table2.column("B2O3", anchor=tk.CENTER, width=100,
minwidth=25)
        comparison_table2.column("Density, ρ (g/cm3)", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Molar Volume, Vm (cm3/mol)",
anchor=tk.CENTER, width=120, minwidth=25)
        comparison_table2.column("Longitudinal velocity, VL (m/s)",
anchor=tk.CENTER, width=120, minwidth=25)
        comparison_table2.column("Shear velocity, VL (m/s)",
anchor=tk.CENTER, width=120, minwidth=25)
        comparison_table2.column("Longitudinal modulu, L", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Shear modulus,G", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Bulk modulus, K", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Young modulus, E", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Microhardness, H (GPa)", anchor=tk.CENTER,
width=120, minwidth=25)
```

```
        ,
    comparison_table2.column("fractal bond connectivity, d",
anchor=tk.CENTER, width=120, minwidth=25)
        comparison_table2.column("Poisson ratio, σ", anchor=tk.CENTER,
width=120, minwidth=25)
        comparison_table2.column("Optical bandgap (eV)", anchor=tk.CENTER,
width=120, minwidth=25)

        comparison_table2.heading("#0", text="", anchor=tk.CENTER)
        comparison_table2.heading("ZnO", text="ZnO", anchor=tk.CENTER)
        comparison_table2.heading("Bi2O3", text="Bi2O3", anchor=tk.CENTER)
        comparison_table2.heading("TeO2", text="TeO2", anchor=tk.CENTER)
        comparison_table2.heading("B2O3", text="B2O3", anchor=tk.CENTER)
        comparison_table2.heading("Density, ρ (g/cm3)", text="Density, ρ (g/cm3)",
anchor=tk.CENTER)
        comparison_table2.heading("Molar Volume, Vm (cm3/mol)", text="Molar
Volume, Vm (cm3/mol)"
                                  , anchor=tk.CENTER)
        comparison_table2.heading("Longitudinal velocity, VL (m/s)",
text="Longitudinal velocity, VL (m/s)"
                                  ,anchor=tk.CENTER)
        comparison_table2.heading("Shear velocity, VL (m/s)", text="Shear
velocity, VL (m/s)", anchor=tk.CENTER)
        comparison_table2.heading("Longitudinal modulu, L", text="Longitudinal
modulu, L", anchor=tk.CENTER)
        comparison_table2.heading("Shear modulus,G", text="Shear modulus,G",
anchor=tk.CENTER)
        comparison_table2.heading("Bulk modulus, K", text="Bulk modulus",
anchor=tk.CENTER)
        comparison_table2.heading("Young modulus, E", text="Young modulus, E",
anchor=tk.CENTER)
        comparison_table2.heading("Microhardness, H (GPa)",
text="Microhardness, H (GPa)", anchor=tk.CENTER)
        comparison_table2.heading("fractal bond connectivity, d", text="fractal
bond connectivity", anchor=tk.CENTER)
        comparison_table2.heading("Poisson ratio, σ", text="Poisson ratio, σ",
anchor=tk.CENTER)
        comparison_table2.heading("Optical bandgap (eV)", text="Optical bandgap
(eV)", anchor=tk.CENTER)

comparison_table2.insert(parent='', index='end', iid=0, text="
                         , values=(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0))
```

```
for i in range(1, 19):
            comparison_table2.insert(parent='', index='end', iid=i, text='', values=(
                "{:.3f}".format(self.x_train[i - 1][0].numpy()),
                "{:.3f}".format(self.x_train[i - 1][1].numpy()),
                "{:.3f}".format(self.x_train[i - 1][2].numpy()),
                "{:.3f}".format(self.x_train[i - 1][3].numpy()),
                "{:.3f}".format(self.yHat[i - 1][0].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][1].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][2].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][3].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][4].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][5].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][6].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][7].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][8].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][9].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][10].detach().numpy()),
                "{:.3f}".format(self.yHat[i - 1][11].detach().numpy())
            ))

        comparison_table2.pack(pady=20)
        self.delta_grid.mainloop()


    def alpha_grid(self):
        self.alpha_grid = tk.Tk()
        self.alpha_grid.title("Elastic & Optical Properties of Binary Glass System
Simulator")
        self.alpha_grid.geometry('600x150')

        Label_alpha1 = tk.Label(self.alpha_grid,
     text="Elastic & Optical Properties of Binary Glass System Simulator",
font=('Helvetica', 15, 'bold'), fg= 'blue4')
        button_train = tk.Button(self.alpha_grid,
     text='Train Model',command= self.click, padx=20, pady=10,font=('Helvetica',
15, 'bold'), bg= 'cyan')

        Label_alpha1.grid(column=1, row=0, padx=(10, 10), pady=(10, 5))
        button_train.grid(column=1, row=2, pady=(10, 10))
        button_train.bind('<Return>', self.click)

            self.alpha_grid.mainloop()
if __name__ == "__main__":
    event_handler = Binary_Glass_Simulator()
    event_handler.alpha_grid()
```