National College of Ireland

Automated Machine Learning: Understanding the relation between Data and Neurons

MSc Research Project
Artificial Intelligence

Mugil Sivasamy Kalamani
Student ID: X22165096

School of Computing
National College of Ireland

Supervisor: Muslim Jameel Syed

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Mugil Sivasamy Kalamani
Student ID:	X22165096
Programme:	Artificial Intelligence
Year:	2023 Jan - 2024 Jan
Module:	MSc Research Project
Supervisor:	Muslim Jameel Syed
Submission Due Date:	31/1/2023
Project Title:	Automated Machine Learning: Understanding the relation between Data and Neurons
Word Count:	4978
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mugil Sivasamy Kalamani
Date:	31st January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automated Machine Learning: Understanding the relation between Data and Neurons

Mugil Sivasamy Kalamani
X22165096

Abstract

Automated Machine Learning is one of the fields that can improve productivity in every possible field as it automates most of the ML tasks, allowing everyone to develop AI systems. Neural networks are the base of all AI systems in the current world, and they are very complex, making it difficult to understand the deep workings of the dataset. Without that knowledge, automating neural network model selection and tuning is impossible. This research study mainly focuses on selecting the no of neurons in each layer and the number of layers in the neural network. As a bonus, the analysis gave a deep understanding of the contribution of the Activation function to fitting the curve. The basic straight-line equation y = mx + c is the basis of the neural network, and a neuron holds every straight line or a single-valued function. Thus, from all the experiments, it is found that the minimum no of neurons required in the first layers is (e + 1), where e is the total no of extrema in the data column. For n-dimensional Dataset, the sum of all extremas in all the columns + no of columns. The first layer of the neural network is the most important, and the activation function is chosen Based on how the data relation is; if it is a smooth curve, smooth functions like sigmoid and tanh can be used, and for sharp-edged curves, ReLU and other functions should be used.

1 Introduction

In today's world, Artificial intelligence is developing extremely fast, leading to automating every bit of manpower-intensive task. Artificial intelligence is in every possible field, but it needs domain knowledge and a high-level understanding of data to develop and tune the AI systems. Machine learning and neural networks form the basis of many AI systems. Automated Machine Learning is the field where a multitude of research and development is going on for the automation of ML model training and tuning. Existing AutoML techniques use grid search or random search, where the subset of the primary dataset is fit using an optimal range of parameters, and the best model is selected based on the model's performance on the subgroup. (Grid Search: It is a method of automated Machine learning, where all possible Hyperparameter are given as a grid and models are generated with all the combinations of these hyperparameters in the grid all these models are trained on the Dataset and the function returns the combination with the best performance.) This method usually needs a lot of Computing power and time to train multiple models, so it is traditionally done on cloud servers to decrease time and increase performance. However, the trust in cloud platforms' privacy and data security

is a question mark for many small and large industries. Thus, finding a better and more efficient way to tune the model is essential to improve productivity in all fields.

My focus on AutoML is to understand the basis of Neural networks, how all the weights, biases, activation functions and hyperparameters contribute to the fitting of data and how the fitting of the model to the data is changed with each parameter. In Neural networks, the main parameters are the neurons in each layer, and no layers are needed for the model to fit the particular dataset. These two parameters significantly determine the model's ability to work on the Dataset. The objective is to analyse the function of individual neurons in modelling linear equations of the data and how combinations of neurons, layers and activation functions model non-linear relationships. This research bridges a gap in understanding the direct impact of neural networks and the performance of regression tasks.

The motivation for the project is obtained from personal thought while learning about Automated Machine Learning's existing tools and techniques used in Hyperparameter Optimization in the AutoML tools. It was found that all the AutoML tools use Grid Search, Random Search and a few other HPO methods to find an optimal model for that particular dataset. All these methods start with a blind selection of Hyperparameter space and use all its combinations to test the small sample sub-dataset and select the combination with the best performance to apply to the whole dataset. But from a mathematician's point of view, behind all ML algorithms is pure mathematics, and ML is just fitting a straight or curved line through the training data points, so it is possible to use a different mathematical approach to select the parameters and Hyperparameter.

The Research was narrowed down to a simple Artificial Neural network and its application on regression problems. The best way to develop an AutoML system to select ANN for regression tasks is to start by understanding the basics of neural networks, and how each neuron contributes to the fitting of the dataset. And so all the literature on the development of basic perceptron to current Deep neural networks was reviewed.

2 Related Work

In the article, Hutter et al. (2019) mentioned that AutoML has taken a boost, but many successes still rely on human work. In the article "Deep learning", Giovanni details the Application of deep learning in two fields, RNN and CNN, for text and image-based tasks.

In 2021, He et al. (2021) surveyed all state-of-the-art AutoML tools and methods and discussed all the existing limitations and advantages of those tools.

Many different approaches to automating the neural network have already been taken, and Nagy and Boros (2021), in an article, discussed the use of reinforcement learning for ANN automation.

Perceptors are the primary neural network building blocks, and the article "The Perception" by Rosenblatt (1958) explains the idea from which the neural network started to develop. It gives an essential perspective to understand neural networks.

An introduction to computational geometry by Arbib (1969) discusses the basics of perceptron's mathematical formulation, giving a perfect mathematical relationship on how ANNs work in depth. David E Rumelhart 1986 introduced the backpropagation method to correct the weight and bias in the function to fit the data.

The performance and features of Auto-Sklearn, an automatic tool in the sklearn framework, are reviewed in detail by Rosenblatt (1958). The testing was done rigorously in all

possible ways and pushed the limits of the AutoML tools.

The idea of neurons and Neural networks was started after **?** wrote a paper on how neurons in the brain may work. This paper discussed neurons' multi-dimensional connection and ability to process information efficiently and quickly.

In the attempt to determine the number of hidden layers and neurons in a layer for wind speed prediction,Rachmatullah et al. (2021) wrote a paper on this, which gives a good insight into how to approach the problem of selecting the optimal no of neurons in the layer.

The main thing in machine learning is understanding the data and measuring the complexity of any dataset; there is no proper method to measure the complexity of data. The article by Zubek and Plewczynski (2016) described a way of assessing the complexity of a dataset by looking at the probability distribution of the data.

To understand DNN, Zhang et al. (2023) introduced an approach of using linear separability to separate hidden layers and understand the working deep neural network and all its Hyperparameters. The Research gives a good insight into how to look at the Neurons and the layers and their relationship to the dataset.

Stühler et al. (2023) benchmarked automl performance in price forecasting, and the results were promising, so the AutoML tools could be significantly reliable. Similarly, Warner et al. (2023) benchmarked ELM and Google AutoML and found that ELM is much better than AutoML.

Johannes et al., in 2022, introduced an approach to select models by linking the model performance to the features of the production process.

The usage of AutoML in the medical field is essential. Still, Manfred et al., in an article on the use of AutoML in radio neurology, says that conventional models perform better than AutoML.

With the mathematical approach in mind, all the existing literature on Hyperparameter Optimization and selection of the model is reviewed in search of any mathematical way to select these parameters. Still, all the research work was very narrowed down to a single dataset and used blind testing of Hyperparameters on that particular dataset or mostly performance review of multiple AutoML tools. Thus there is no direct existing literature to motivate the study.

3 Methodology

The Research started with a primary literature review on the problem, how the existing AutoML techniques are functioning currently, and the limitations and issues in the domain faced by both the user and technique development end. A clean and neat systematic approach, as in the Figure 1, was followed throughout the research work.

Before starting with the understanding of basic Neural networks, the Idea was to find a way to measure the dataset or its complexity and train multiple models with different datasets of increasing complexity and map their performance with complexity to obtain a graph of model combination to dataset complexity, thus by interpolating And extrapolating the graph, we can directly choose the model combination by measuring dataset complexity and use that combination. Thus to test this, the study was done on synthetic data and a real-world dataset, to have a basic idea of models and their combination's performance on different datasets.

The practical research started with basic testing of Neural network model performance

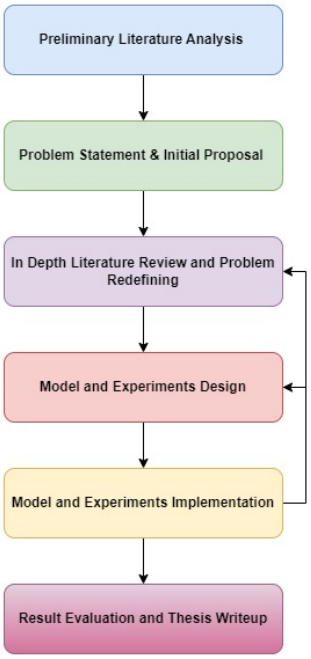

Figure 1: Research Process

for different parameter (Layers, Number of neurons, and activation function) settings for two datasets. This is similar to an AutoML technique in which a grid search method defines the parameter space. Still, the moto is not to tune the model for best performance but to understand how these parameters, when chosen wrong, can take the performance to worst. The experiment gave an excellent insight into all the limitations and issues with the existing techniques, like the time and computational resources it takes. The experiment also gave an outline of the Hyperparameter's contribution to the fitting of the dataset.

For the experiment, The real-world dataset is a pollution dataset under the Google Air View Dublin Project, which is carried one in collaboration with Dublin City Council to improve the air quality of Dublin City *Google Project Air View Data - Dublin City (May 2021 - August 2022) - data.smartdublin.ie* (n.d.). To this dataset, other weather information is added from the Weather website *Met Éireann - the Irish Meteorological Service* (n.d.). The dataset is from a mobile sensor, and the dataset has 5 million data points over one year, with various features contributing to Pollution like traffic count, weather information, etc. This dataset is taken as it has a very complex relationship between the features and the pollution levels. The pollution dataset is preprocessed to remove anomalies, feature extraction for a few columns, and dimensionality reduction for fields like latitude and longitude.

The second dataset is a synthetic dataset containing 15 elements, created using mathematical functions like sin, tan, exponent, log and polynomial equations and a combination of these functions and equations. These functions are simple and can be easily related to y values, so it is taken as an example of a simple dataset.

The Synthetic dataset was generated to test the model's performance in addition to the real-world dataset. The main reason for generating a synthetic dataset is that it is

easy to know the pattern in data as we know the mathematical function that generated the data. Also, this study is to understand the model and not to train a model with the best accuracy to deploy in real-world usage, to understand the model to the fullest, we first have to understand the data on which the model is trained. In this case, real-world data may have an Extreme amount of noise and anomalies, which makes it hard to manually understand the pattern in data. Thus generating synthetic data where the function creating the pattern is known is well-suited for this study. To generate the dataset, multiple mathematical functions were used, and all are listed in the Figure 2.

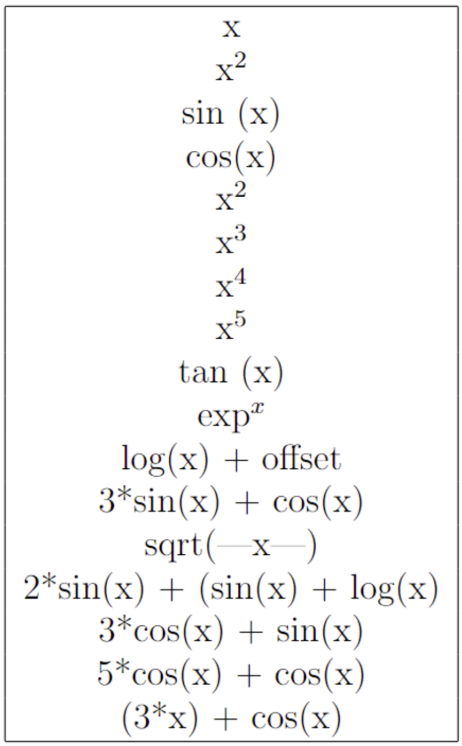

Figure 2: Equations to generate Synthetic Dataset

The models were trained on the same hyperparameter space for both datasets. Complex and simple two datasets give a fundamental relationship between the model's hyperparameters contribution and the data relation to y values Figure 3. This insight also motivates us to understand the depth of neurons, layers, and activation functions and their contribution to fitting linear two-dimensional data to non-linear multi-dimensional data Figure 4.

The idea is to find the optimal no of neurons in each layer and no of layers, activation function, etc. To understand these parameters, the experiment started with a primary single neuron. It went up to multi-layer deep neural network architecture, and the data was synthesised from linear line equation and up to multi-dimensional non-linear equation.

3.1 Equipment and Software

Equipment and Software: The study primarily used Python programming language to develop and test neural networks. The basic neural network structure and functions

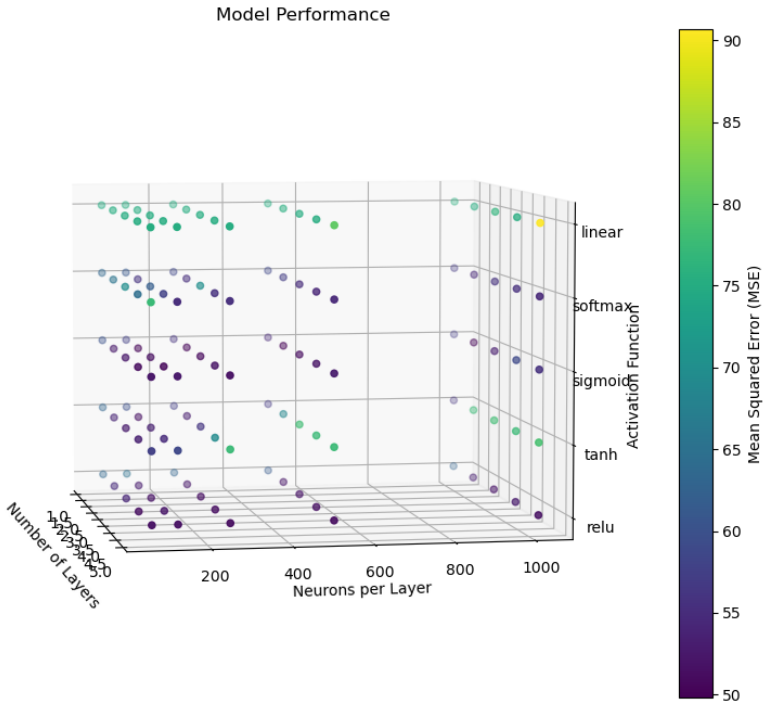

Figure 3: Model Performance on Pollution dataset

were constructed using frameworks like TensorFlow and Keras. The pandas and numpy library were used for synthesising the dataset and Data preprocessing, transformation, etc. The data visualisation framework used was Matplotlib for its simple and efficient visualisation. For the programming Interface, Pycharm IDE and Jupyter Notebook were used.

All the study and experiments were done on a local machine, with a 14-core (20-thread) Intel i9 processor and a DDR5 RAM of 16 GB. The device has an NVIDIA RTX 3060 GPU with dedicated 6 GB RAM for accelerated computing.

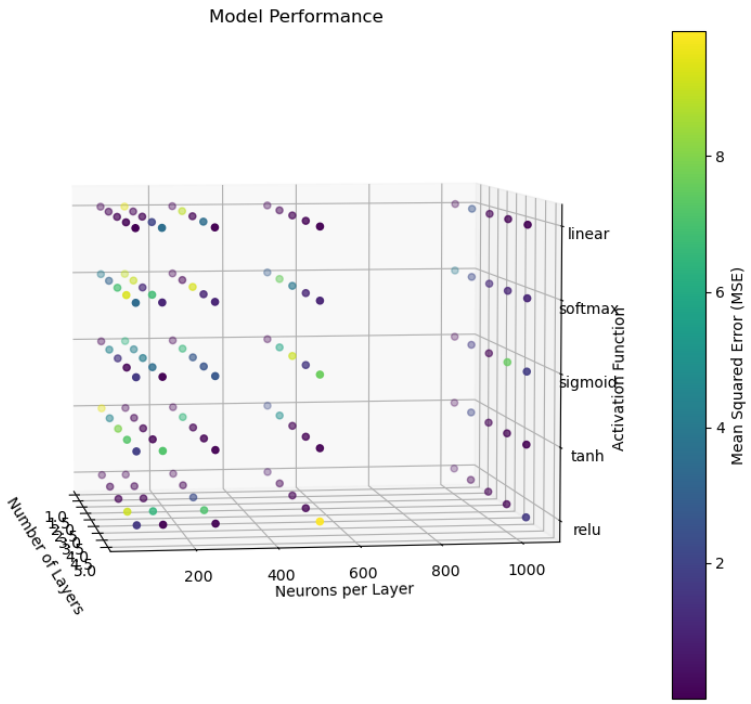

Figure 4: Model Performance on Synthetic Dataset

3.2 Evaluation Methodology

To evaluate these simple neurons, the function or polynomial equation used to generate the dataset is compared with the neural network weight and bias equation by rearranging it as a simple mathematical equation used to create the dataset. The weight and bias of each neuron are extracted after training the model to a mean squared error of 10 power -6 as it is a regression task and is substituted in the weight bias equation of neural networks to get back the mathematical equation used to generate data. Suppose the error is high even after multiple epochs. In that case, that indicates that the model is not able to fit the data, and the weight and bias equation should be able to rearrange in the polynomial equation that generated the data; if not rearrangeable, then this again indicates that the model architecture is not sufficient to fit that data. The model's inability can be visually

seen using the plots generated after the training. This proves the direct link between neural network weight, bias function and dataset generation equation. Also, to visualise the data and model fitting ability, the x values of the dataset are entirely passed to the trained model to predict the y values and plotted together in a graph.

4 Design Specification

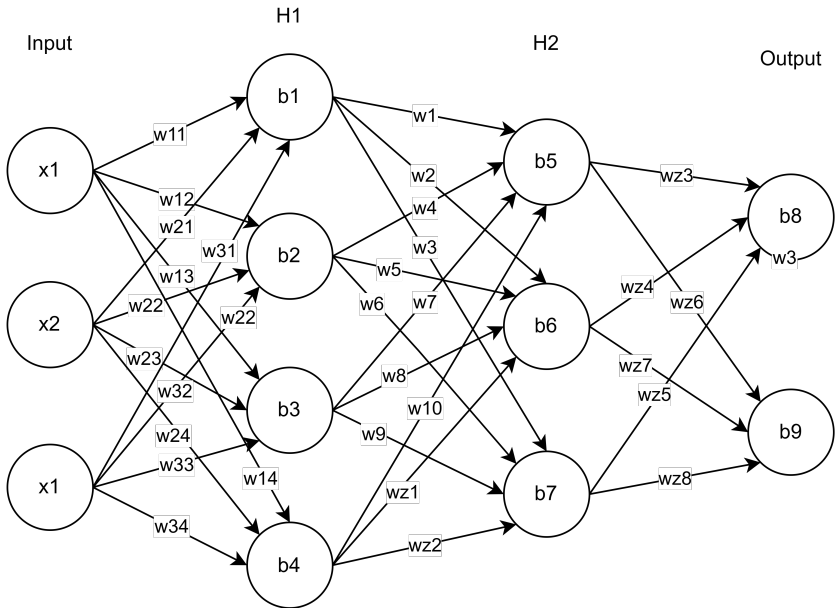

Figure 5: Neural Network

A Neural Network is a massively complex system made of tiny neurons, where each neuron has a weight associated with its connection to the previous layer, a bias, and an activation function. The equation of a simple straight line is given by

$$y = mx + c$$

Where m is the slope of that line in an N-dimensional space, and c is the point where the line intercepts the y-axis. Similarly, when we look at a neural network, weight is compared to the slope, and intercept is compared to the neuron's bias. I.e. weight(w) is multiplied by the input (x), and bias(b) is added to the weight multiplied by input (w . x). Every single neuron in the neural network works in the same way. If a neuron has multiple inputs (x1, x2, x3 . . . xn), then it has a weight corresponding to each input, i.e. (w1, w2, w3 . . . wn). All the inputs are multiplied by their corresponding weights and added to their own bias(b). The activation function is a mathematical function that takes this w. x + b equation is used as input and gives a squished or curved line function as output. The shape of the production depends on the activation function intended to be used. This is the primary single-neuron architecture. Every other Neural network architecture is created by adding the neurons in a vertical or horizontal stack.

For the study of multiple model architecture's performance on the two mentioned datasets, the model architecture is defined using hyperparameter space; the parameters are no of layers, no of neurons and activation function. The model is trained for all the

combinations of hyperparameters in the parameter space, and the mean squared error is saved to a file. Then, multiple plots are plotted to get valuable insight into the model's performance. This experiment also gives an idea of which parameter is inducing the error in the fitting of the data, which tells us which parameter's contribution we have to understand deeply to avoid overfitting or underfitting the model.

To study the individual contribution of each neuron to the fitting of data, the experiment started with a simple single-neuron architecture, and it went up to 4 neurons in the horizontal stack, four neurons in the vertical Stack and all other combinations inside the range.

The basic idea starts by checking the minimum number of neurons and layers required to fit a straight line data and move to a single sin wave and then to multiple sine waves. Then, two-dimensional data is provided from a plane surface to non-linear plane data. To test the fitting, the data is plotted and visually checked. For a considered case, if the defined architecture cannot fit the data even after thousands of epochs, the architecture is insufficient, and the neurons should be added. First, neurons are added vertically (i.e., in the same layer), and if they are still inadequate, new layers are introduced to fit the complexity of the data. This method will give the minimum no of neurons required to provide a particular data, either a straight line or a curve.

4.1 Measuring Dataset Complexity

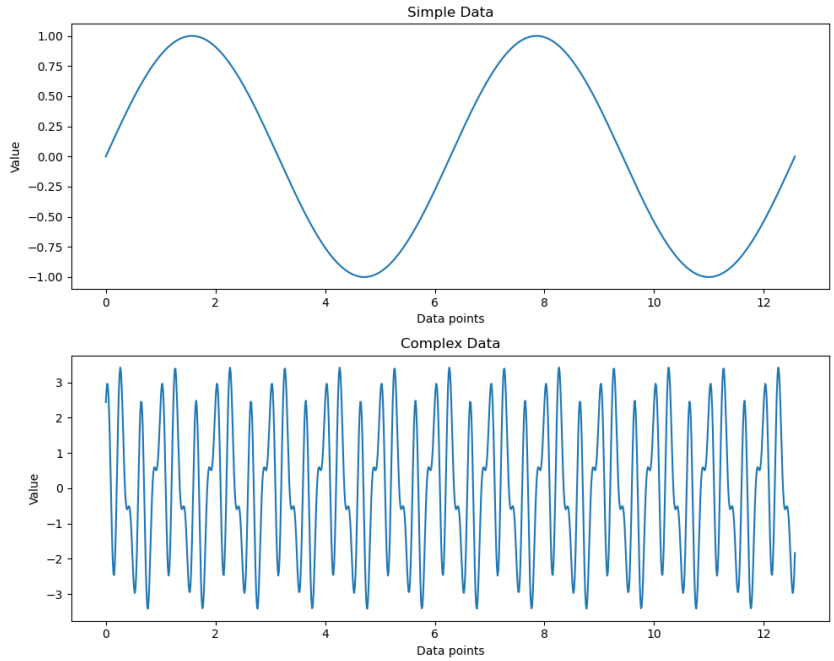

Figure 6: Simple and Complex Data

The simplest way to measure any curve is to find the local maxima and local minima present in the data. The sum of no of maxima and minima in a data is directly proportional to the complexity of the data. If more maxima and minima are present in the data, then the data is more complex. The study uses this method to evaluate the complexity of the data.

5 Implementation

5.1 Straight line

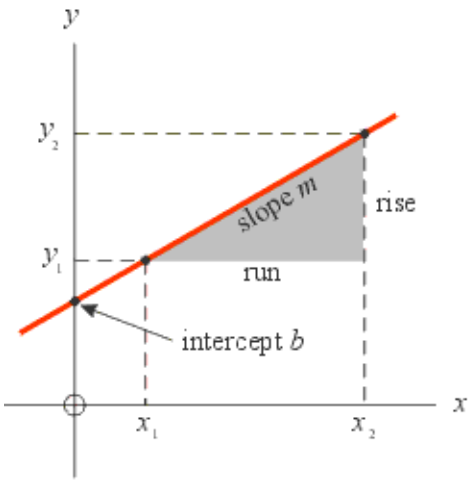

Figure 7: Straight line graph

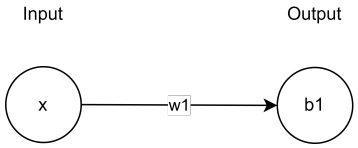

Figure 8: Single Neuron Architecture

The straight line equation is given by

$$y = m.x + c$$

from Figure 7, where m and c are slope and intercept, respectively. For the data generation, the slope value was 0.5, and the intercept was 3. A single neuron architectureFigure 8 is used to fit the straight line, and after training the model, the final weight and bias are extracted. The weight(w) and bias (b) were 0.50229 and 2.99414, respectivelyFigure 9. The model equation is given by,

$$y = w.x + b$$

Comparing these two equations, the model directly fits the straight-line equation, so one neuron is sufficient to provide straight-line data.

For the same straight line, the x and y values were interchanged, and on rearranging the equation, we get

$$x = 2y–6$$

where 2 is the slope and – 6 is the intercept. Weight and bias were found to be 2.00000 and – 5.99998, respectively.

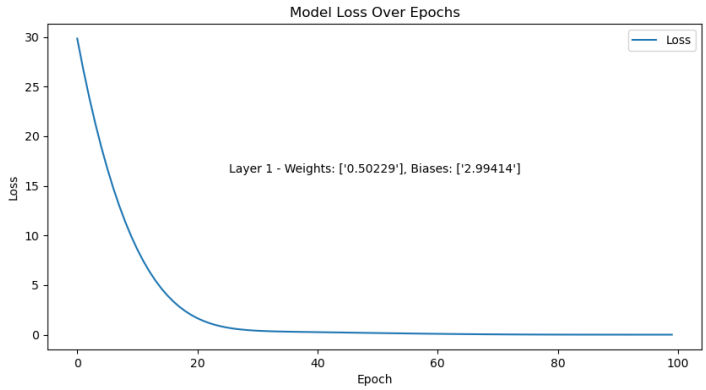

Figure 9: Single Neuron Loss plot

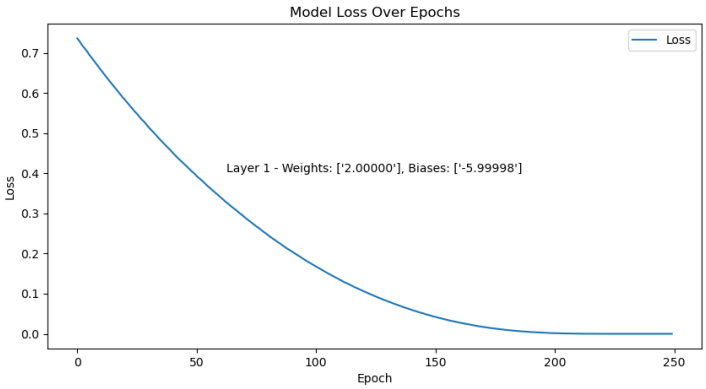

Figure 10: Single Neuron Loss plot for X and Y interchanged

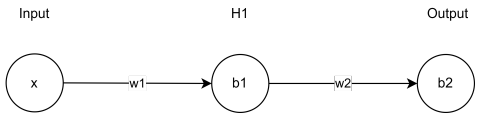

Figure 11: Two layer Architecture

Now, a second layer is introduced with a single neuron, as in the Figure 11, and the same data is fit on the model. The model weight bias equation is given by

$$y = (w1 * x1 + b1)w2 + b2$$

on rearranging, we get

$$y = (w1 * w2) * x1 + (b1 * w2 + b2)$$

which is similar to the straight line equation, where (w1 * w2) is the coefficient of x1, and so it is supposed to be slope and (b1 * w2 + b2) is the intercept. After training the model, we get weight and bias to be [0.34395, 1.45370], [1.21265, 1.23717], respectively; substituting in the model equation, we get weight and bias to be 0.5 and 3, which is the slope and intercept of the line data generated.

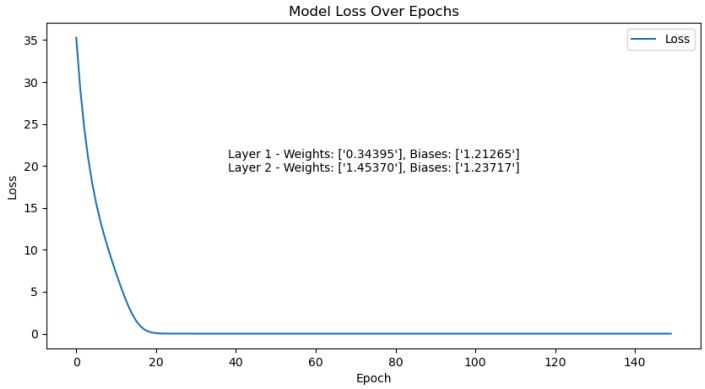

Figure 12: [1,1] architecture Loss plot

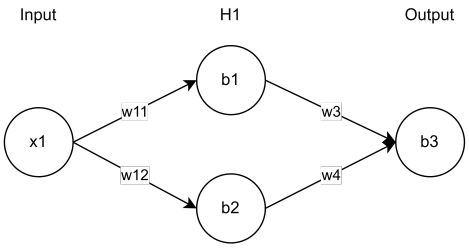

Figure 13: [2, 1] Architecture with one inputs

Two neurons in the first layer and one in the output layer were used for the same line data. As our concentration is on the regression task and the regression task needs only one output, it is mandatory to use a single neuron in the output layer so that it gives a single output. For this model ArchitectureFigure 13, the final rearranged weight bias equation is given by

$$y = (w1 * w3 + w2 * w4) * x1 + (w3 * b1 + w4 * b2 + b3) \quad (1)$$

Weight and bias were found to be Layer 1 = [1.75029, -0.85670], [1.35468, 0.85306], Layer 2 = [0.82297, 1.09774], [0.94868916], respectively. On substituting, We get the slope and intercept of the straight line [0.49053, 3.00115].

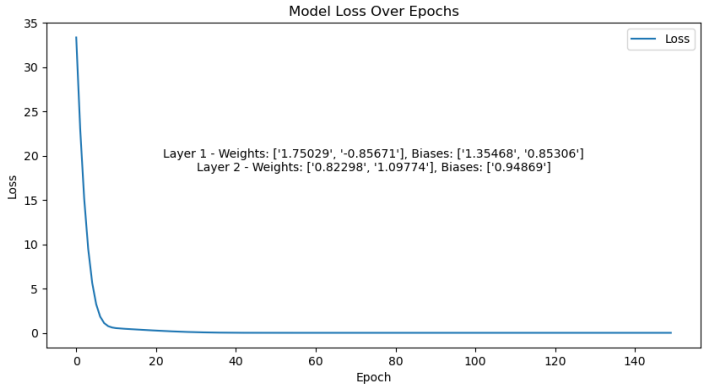

Figure 14: [2, 1] architecture Loss plot

5.2 3D Linear Data

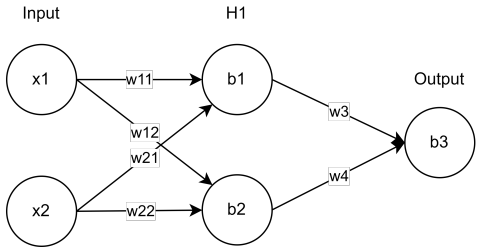

Figure 15: [2, 1] Architecture with two inputs

For plane, two parallel straight line equations make up a 3D plane, so another straight line data is generated using 2 and 3 as slope and intercept, respectively. These two equations combine to form a plane equation; we get

$$x = y + 0.25z–3.73$$

Where the coefficient of y is 1, z is 0.25, and the intercept is -3.75. The model equation on rearranging, we get

$$\begin{aligned} y = &(w11 * w3 + w21 * w4) * x1+ \\ &(w12 * w3 + w22 * w4) * x2+ \\ &(w3 * b1 + w4 * b2 + b3) \end{aligned} \tag{2}$$

The weight and bias after training were extracted and substituted in the equation to get

$$y = 0.924x1 + 0.2685x2–3.5833$$

Comparing this with the plane equation, it is evident that the model could fit the data using two neurons in the Hidden layer and one in the output layer. In all these cases, the linear activation function was used as the data is linear, the linear function wouldn't change the weight and bias according to activation, and the equation was kept simple.

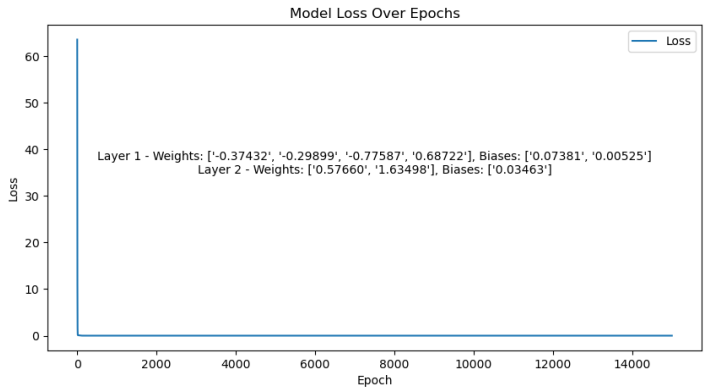

Figure 16: [2, 1] architecture with 2 inputs Loss plot

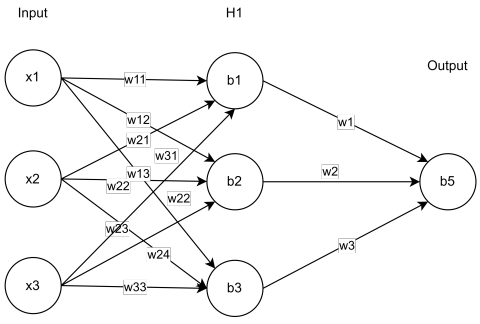

Figure 17: [3, 1] Architecture with three inputs

5.3 4D Linear Data

For 4 Dimensional Data, the simplified weight and bias model equation is given by

$$\begin{aligned} y = &(w11 * wo1 + w21 * wo2 + w31 * wo3) * x1 + \\ &(w12 * wo1 + w22 * wo2 + w32 * wo3) * x2 + \\ &(w13 * wo1 + w23 * wo2 + w33 * wo3) * x3 + \\ &(wo1 * b1 + wo2 * b2 + wo3 * b3 + bo) \end{aligned} \tag{3}$$

and the four-dimensional data mathematical equation is given by

$$a.x + b.y + c.z + d = w$$

Training the model and substituting the bias and weight values found that it could fit the dataset with 3 features.

5.4 Simple Curved lines

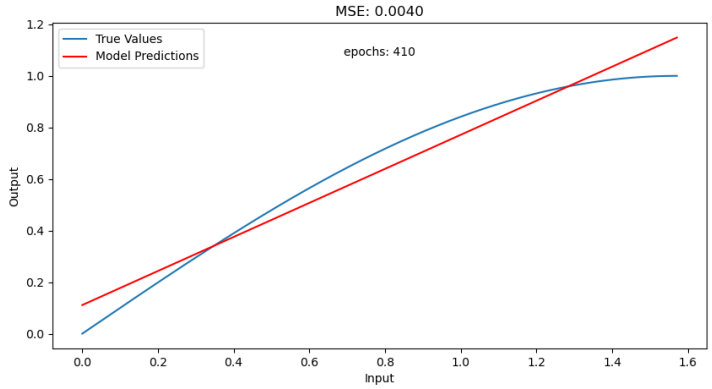

Figure 18: Sin 0 to $\pi/2$ and one neuron neural network

A curved line Figure 18 with zero extrema is generated using a sin function, where the x value ranges from

$$0 - \pi/2$$

A single neuron with a linear activation function was used, and it could not fit the data perfectly as a linear function doesn't introduce a non-linearity in the output. However, the model could fit the curve ideally using the sigmoid activation functionFigure 19.

5.5 Curved line with one maxima

The x range was increased from pi/2 to pi to create data with a single maximum Figure 20. Neural network architecture like [1], [1,1] and [2,1], [3,1], [3, 2,1] were used to fit this parabolic curve. When we look at the simplified mathematical function that defines a single neuron with a linear activation function, is

$$y = m * x + c$$

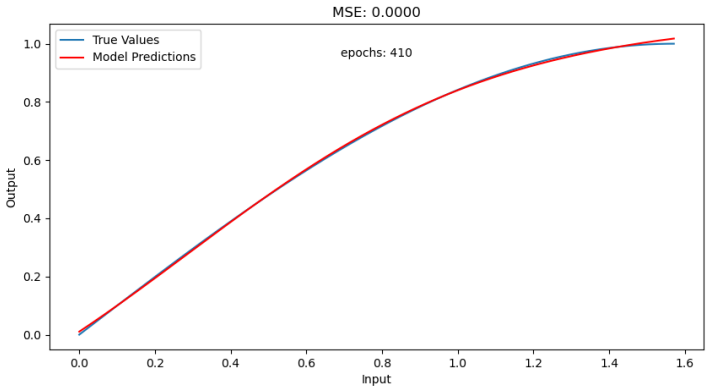

Figure 19: Sin 0 to $\pi/2$ and one neuron neural network

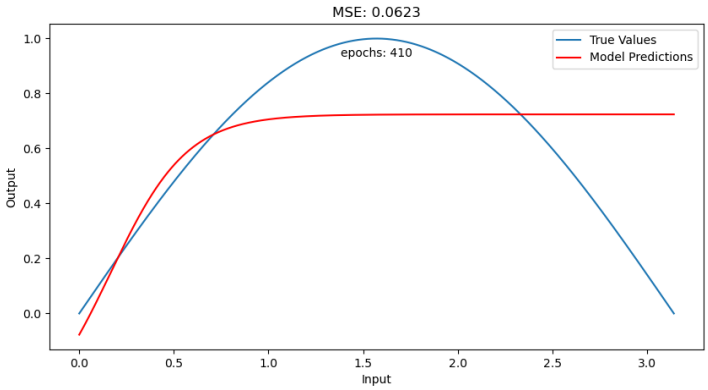

Figure 20: Sin 0 to π and one neuron neural network

which mathematically is an equation of a straight line. All the activation function is used to introduce a slight curvature to the straight line equation, without allowing the line to form a maxima or minima. But for fitting a parabolic curve that has one maxima, a single neuron cannot fit the data, with any activation function, As none of the activation functions is going to introduce a maxima to the line equation of the single neuron. Also increasing the depth of the Neural network is only going to share the weight and bias between depths while remaining as a straight-line equation. When increasing the breadth of the neural network, the equation changes to

$$y =) * x^2 + (w_2 1 * w_1 1 + w_2 2 * w_1 2) * x + (w_2 1 * b_1 + w_2 2 * b_2 + b_3)$$

This is the equation of a parabola,

$$y = a * x^2 + b * x + c$$

thus only increasing breath can fit the parabolic curve and not increasing the depth. Also, here

$$x^2$$

term goes to zero as the linear activation function cannot introduce a curve to the data. Thus only when using a sigmoid activation function can introduce a non-linearity to the data making the Neural network learn that curve. Thus, a neuron is vertically stacked on the first layer, removing the third layer. The model could fit the data perfectly, as in the image below.

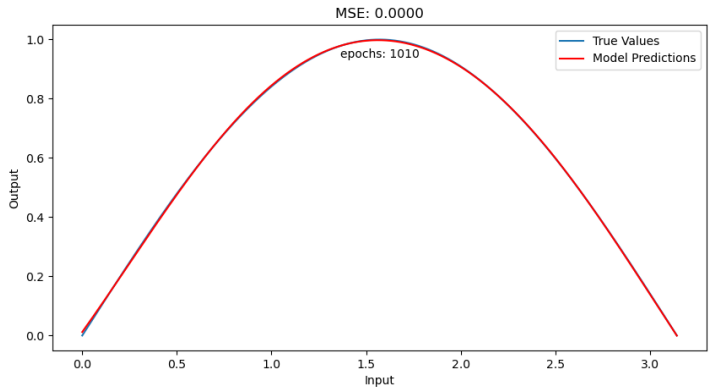

Figure 21: Sin 0 to π and two neuron neural network

5.6 Curved line with one maxima and one minima

Now again, the range of x is increased to (2 . pi), and [2, 1] architecture was used. The model couldn't fit the dataFigure 20, and the horizontal stacking of neurons didn't show any improvement. The model could fit that data when neurons were stacked in [3, 1] architecture, as in the below image.

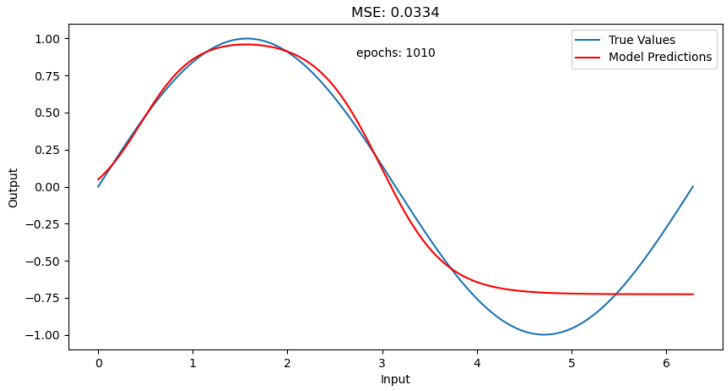

Figure 22: Sin 0 to 2π and two neuron neural network

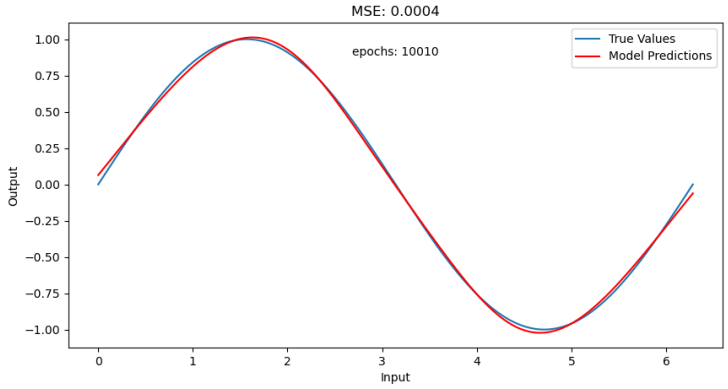

Figure 23: Sin 0 to 2π and three neuron neural network

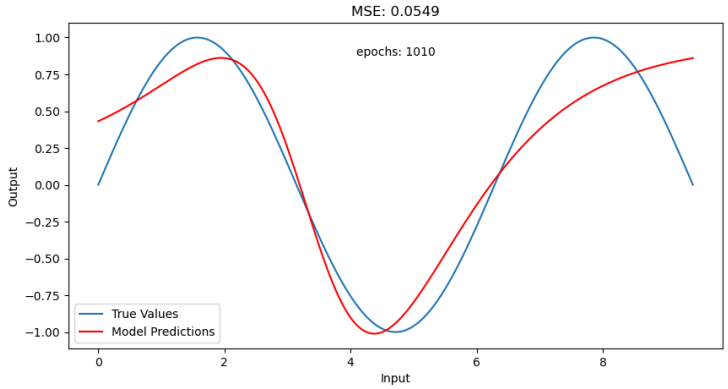

Figure 24: Sin 0 to 3π and three neuron neural network

5.7 Curved line with two maxima and one minima

Again, the x range is increased to (3 . pi), and [3, 1] architecture was used. The model couldn't fit the data, and the horizontal stacking of neurons in layers showed no improvement. The model could fit that data when neurons were stacked in [4, 1] architecture, as in the below image. Also, using other activation functions like "ReLU"

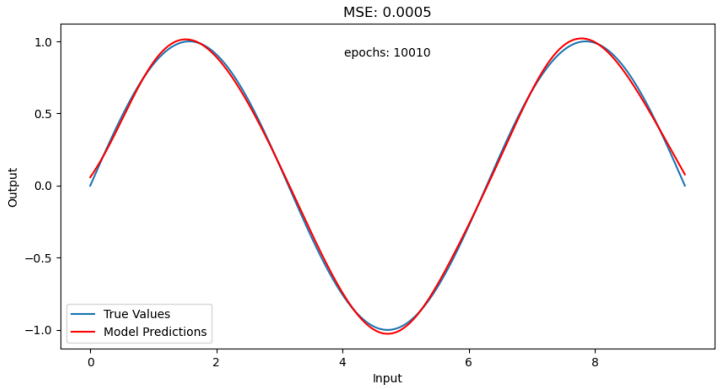

Figure 25: Sin 0 to 3π and four neuron neural network

needed more neurons in vertical stacking to fit the same graphs Figure 26.

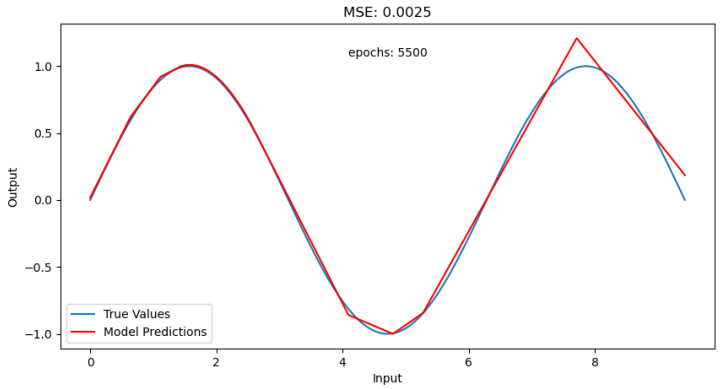

Figure 26: Sin 0 to 3π and 100 neuron neural network

6 Evaluation

The experiments and case studies in the Research not only answered the questions that the answers were sought but also gave many insights into how neural networks work on the data and answered the selection of other hyperparameters. This section discusses all the evaluation results and is only tested for regression tasks.

6.1 No Neurons in each layer

From all the experiments, simple curve with no extrema's (single-valued function), straight line and n-dimensional linear data, as each of the dimensions was able to be fit by a single neuron, the study suggests that the minimum no of neurons in the first layers should n, where n is the no of the dimension of the dataset.

For the data with extrema's, from all the observations, the minimum no of neurons in the first layer is e + 1, Where e is the total no of maxima and minima. And if there are n dimensions in the non-linear data, there must be e + 1 neurons for each dimension in the first layer. Also, stacking neurons vertically in addition to the needed no of neurons will share the weight values while still forming the desired mathematical equation.

6.2 No of layers

From all the experiments, it is evident that the first layer is the most critical layer for the neural network in fitting the curve. No neurons needed for this layer are given by

e + 1, Where e is the total no of extra.

Since our main objective is to focus on the regression tasks, the output layer will always have a single neuron with a linear activation function.

6.3 Activation function

As we have seen, each neuron contributes to fitting a straight line of equation, y = mx + c; the activation function introduces the non-linearity in doing the simple curve. Even though non-linearity is presented, as all the activation functions are single-valued (i.e. the function won't have two values for any point of x), the non-linear curve will also be single-valued. Also, the curve plots show the difference between each activation function and the data they are fitting on. We may need more neurons to fit a smooth curve line using the relu activation function, but the extrema conditions apply to check a zig-zag data using ReLU or leakyReLU. Similarly, a zig-zag curve with a sigmoid activation function may need more neurons. It is best to choose the proper activation function to provide data depending on the fit we seek.

6.4 No of Epochs

Under the condition of selecting the minimum number of neurons in the layer, having more epochs does not overfit the data. Another reason for not overfitting the data even after 1000 epochs was that our data was smooth and continuous. It is always best to use an early stopping callback. Also, fitting happens only if more than the minimum required neurons exist.

6.5 Discussion

The main point to note in the experiment is that it is only done for single-frequency non-linear data. Some expected results may not be achieved if the data has more frequencies. Also, the study gave another hypothesis: converting the data into a frequency domain might answer the no of layers needed for a much better understanding. (i.e. finding the total no of extra for each frequency and stacking neurons vertically for each frequency, and

also giving the first layer to the highest frequency and the last hidden layer to correspond to the lowest frequency.)

As the optimiser uses its epochs to fit the model to the data, having a minimum number of neurons constantly increases the epochs, costing a lot of time. When accelerated processing is used efficiently, the time may be reduced. Also, in some tests, it is seen that adding a layer reduced the epoch drastically.

As a neural network is a substantial complex mathematical equation, it is easy to prove the observation theoretically.

7 Conclusion and Future Work

The research's main objective is to find a better and more efficient way to Automate the ML pipeline by understanding the in-depth contribution of each parameter in a neural network. With findings like (e + 1) no of neurons for the first layer and n(e + 1) for n dimensions, these hyperparameters can be selected very efficiently with a vast per cent decrease in Computing resource usage and time consumed. This also allows it to run on local machines, theory, by making it available for everyone, reducing the dependency on high-performance cloud systems and the data security and privacy risk.

Future work will study more data varieties and observe how the model performs for each data. Also, as said earlier, testing the hypothesis and frequency domain of data may answer a few more questions related to the no of layers and neurons contribution to multiple frequency data.

References

Arbib, M. (1969). Review of 'perceptrons: An introduction to computational geometry' (minsky, m., and papert, s.; 1969), *Information Theory, IEEE Transactions on* **15**: 738–739.

Google Project Air View Data - Dublin City (May 2021 - August 2022) - data.smartdublin.ie (n.d.).
URL: *https://data.smartdublin.ie/dataset/google-airview-data-dublin-city*

He, X., Zhao, K. and Chu, X. (2021). Automl: A survey of the state-of-the-art, *Knowledge-Based Systems* **212**: 106622.

Hutter, F., Kotthoff, L. and Vanschoren, J. (2019). *Automated Machine Learning - Methods, Systems, Challenges*.

Met Éireann - the Irish Meteorological Service (n.d.).
URL: *https://www.met.ie/*

Nagy, A. and Boros, (2021). Improving the sample-efficiency of neural architecture search with reinforcement learning.

Rachmatullah, M. I. C., Santoso, J. and Surendro, K. (2021). Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction, *PeerJ* **7**: e724.
URL: *https://doi.org/10.7717/peerj-cs.724*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain [j], *Psychol. Review* **65**: 386 – 408.

Stühler, H., Zöller, M.-A., Klau, D., Bedrikow, A. B. and Tutschku, C. (2023). Benchmarking automated machine learning methods for price forecasting applications, *ArXiv* **abs/2304.14735**.
URL: *https://api.semanticscholar.org/CorpusID:258418202*

Warner, B., Ratner, E. and Lendasse, A. (2023). *Edammo's Extreme AutoML Technology – Benchmarks and analysis*.
URL: *https://doi.org/10.1007/978-3-031-21678-7*$_1$5

Zhang, C., Chen, X., Li, W., Liu, L., Wu, W. and Tao, D. (2023). Understanding deep neural networks via linear separability of hidden layers, *arXiv (Cornell University)* .
URL: *https://arxiv.org/abs/2307.13962*

Zubek, J. and Plewczynski, D. (2016). Complexity curve: a graphical measure of data complexity and classifier performance, *PeerJ* **2**: e76.
URL: *https://doi.org/10.7717/peerj-cs.76*