SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Analysis of the use of Nguyen Widrow Algorithm in Backpropagation Kidney Disease

Romanus Damanik[1)], Muhammad Zarlis[2)], Zakarias Situmorang[3)]
[1,2,3)] Universitas Sumatera Utara, Medan, Indonesia
[1)]rdfikom@gmail.com, [2)]m.zarlis@yahoo.com, [3)]zakarias65@yahoo.com

Submitted : Jan 14, 2024 | **Accepted :** Feb 15, 2024 | **Published :** Apr 1, 2024

Abstract: Fast and accurate diagnosis is very important for kidney disease. This research conducts and analyzes by using Nguyen Widrow Algorithm in Back Propagation method in artificial neural network for kidney disease diagnosis with the aim to improve the accuracy in predicting and time efficiency in diagnosing. The Nguyen Widrow algorithm is very capable of accelerating convergence and stabilizing the learning process in artificial neural networks, which is also expected to present a meaningful contribution to the handling of health data. This study uses MATLAB as a platform for algorithm implementation and a dataset of medical records of kidney disease patients collected from a hospital that specializes in treating kidney disease patients. The data pre-processing and artificial neural network modelling stages use the Nguyen Widrow algorithm, while the model training process uses the Back Propagation method. The results showed that the Nguyen Widrow algorithm was able to improve the accuracy of predicting someone suffering from kidney disease compared to using only the Back Propagation method. Analysis of the performance of the model shows a significant improvement in stability and convergence speed during the learning process. This indicates that data processing and medical decision making becomes more efficient. On the other hand, this research also studied the challenges and limitations that will be faced in terms of implementation of the Nguyen Widrow algorithm. Also the sensitivity of the initialization parameters, the need for the quality of the dataset to be used in training the model. This research reveals the ability of the Nguyen Widrow algorithm to improve the performance of artificial neural networks in diagnosing kidney disease. By implementing this algorithm in MATLAB, the results show that the use of the latest data processing technology and analysis tools can provide significant improvements in accuracy and efficiency in the medical field. In addition, this research is expected to provide a new direction in the development of machine learning algorithms for applications in the healthcare field, especially for diagnosing kidney disease. By further utilizing this technology, it contributes significantly to improving the quality of healthcare and treatment outcomes for patients suffering from kidney disease.

Keywords: Nguyen Widrow, Kidney Deseases, Back Propagation, Artificial Neural Network, Machine Learning.

INTRODUCTION

Millions of people die every year from kidney disease, which is one of the most severe health problems worldwide. To remove waste and excess fluid from the blood, maintain electrolyte balance, and maintain homeostasis, the kidneys must function properly. Without proper diagnosis and prompt medical treatment, such as dialysis or transplantation, kidney failure can be fatal(Tiastama et al., 2021). However, a major obstacle in the diagnosis of kidney disease is that symptoms are often non-specific and only appear when the disease has progressed. This makes early detection important, albeit difficult. The risk of serious complications increases if diagnosis is delayed, showing how important it is to use more sophisticated and accurate approaches to kidney disease diagnosis(Yudi Setyawan, 2016).

Advances in information technology in recent decades have provided new tools that can aid in medical decision-making. Artificial Neural Networks (ANNs) have emerged as a possibly revolutionary medical data processing technique. The Artificial Neural Network provide new hope for the development of more accurate and

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

efficient kidney disease diagnosis systems due to their unique ability to identify patterns from a variety of clinical data. The Artificial Neural Network have proven useful in a variety of medical applications, including medical image interpretation and biomarker identification for various conditions.
The backpropagation algorithm is one of the main algorithms used for Artificial Neural Network training. With this algorithm, the network can adjust its weights and biases iteratively. The goal of this algorithm is to reduce the difference between real and predicted results. As a result, backpropagation is essential for improving the accuracy of Artificial Neural Network models in classification tasks such as kidney disease diagnosis. However, proper parameter selection is required for successful backpropagation training. Many initialization methods have been proposed to improve this process. These include the Nguyen-Widrow method. This method aims to provide an ideal starting point for Artificial Neural Network training, which in turn can accelerate convergence and improve the performance of the final model (solver, 2024).

Problem Formulation

How can optimizing the backpropagation algorithm by applying the Nguyen-Widrow method improve accuracy and efficiency in diagnosing kidney disease through Artificial Neural Networks?

Research Objectives

The main objective of this research is to develop and validate an optimized Artificial Neural Network (ANN) model with backpropagation algorithm using Nguyen-Widrow initialization approach to improve accuracy in diagnosing kidney disease. Specifically, this research aims to:

1. Assessing the effectiveness of the Nguyen-Widrow algorithm in initializing the weights of Artificial Neural Networks, with the hope of achieving faster convergence and higher accuracy compared to random weight initialization.
2. Integrate and test the optimized Artificial Neural Network model on real clinical datasets to diagnose kidney disease, by comparing its prediction results against the results from conventional diagnostic methods.
3. Identify and analyze the clinical features that contribute most to model accuracy, enabling a deeper understanding of the key factors in kidney disease diagnoses.

Research Benefits

Some of the benefits of this research are theoretical, practical, and technological. The first is the theoretical benefit are as follows:

1. This research will provide empirical evidence on the Nguyen-Widrow method as an initialization technique for the backpropagation algorithm. It will enrich the literature in the field of artificial intelligence and medical data processing and provide new insights into the relationship between known patterns and clinical characteristics of kidney disease.
2. Practical Benefits: Accurate diagnosis of kidney disease can save lives through early detection and more targeted treatment, reduce long-term health impacts and financial burden for patients and health systems, and provide diagnostic tools that can be used by physicians and other medical personnel to increase diagnostic confidence in cases that are difficult to identify through conventional approaches.
3. The potential use of Artificial Intelligence in healthcare will be demonstrated by technological benefits, such as the development of optimized Artificial Neural Network models. The development of these models could drive the development of more advanced automated diagnostic tools and encourage a more data- and technology-focused health system(Mahfuzh et al., 2020).

METHOD

The method used in this research is a qualitative survey. The research location is Rasyida Kidney Specialty Hospital Medan. Data collection was done by observation and interview. Data sources were obtained from one of the hospital's clinical staff.

Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is an adaptive system that uses neurons, a group of small processing units, to mimic the structure and function of human nerves (neptune.ai, 2023). A Artificial Neural Network can change its structure to solve problems due to information flowing through the network or from outside. In other words, a Artificial Neural Network is a non-linear statistical data modeling tool that can be used to model complex relationships between inputs and outputs to find patterns in the data. The Artificial Neural Network architecture understands data through various layers of mathematical processing. Artificial neural networks typically consist of tens to millions of artificial neurons referred to as units. Multiple layers, or layers, including input layers, hidden layers, and output layers, comprise an artificial neural network (datacamp, 2024).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The Artificial Neural Network receive input signals from external sources, in the form of patterns or images represented in vectors. These inputs are then mathematically designated with the notation x(n) for every n number of inputs. Thereafter, each input is multiplied by its corresponding weight, which represents the strength of the interconnections between neurons in the artificial neural network. All the weighted inputs are summarized in the computational unit. One important characteristic of The Artificial Neural Network is fault tolerance, which means that the loss of one or more cells does not affect the performance of the artificial neural network. In addition, the network stores data throughout the network, which means that the network does not produce output even if there are no data pairs. In addition, it is difficult to find the right network structure, which causes the selection of the right artificial neural network architecture can only be achieved through practice and error (machinelearningmastery, 2024).

The Artificial Neural Network aim to mimic the network of neurons that make up the human brain, so that computers can understand things and make decisions in a manner similar to how humans think. In other words, The Artificial Neural Network seek to mimic the mechanisms of the human brain in terms of data distribution, which allows us to extract more than one data from our memory simultaneously if needed.

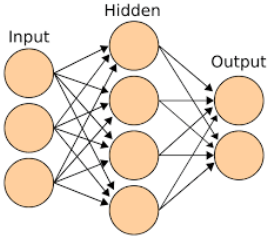

Fig.1 Illustration of Artificial Neural Network

Back Propagation

In machine learning, backpropagation is an algorithm used to optimize the weights of Artificial Neural Networks. The algorithm works in three main phases including:

Phase 1: Feedforward: in this phase, data is fed into the network and processed from the input layer to the output layer. Each neuron calculates its output based on the input and weights.

Phase 2: Backward propagation (Backpropagation). After the output is generated, the error between the generated output and the target is calculated . This error is then passed backwards (from the output layer to the input layer) to calculate the gradient of the loss function against the weights and bias. This gradient is used to change the weights and bias of the network with the aim of reducing the error (Matt Mazur, 2023).

Phase 3: weight change: The gradient calculated in the backward propagation phase is used to change the bias and weights. This is done by subtracting the gradient of the loss against the weights and bias then multiplied by the learning rate. Replica neural networks use the flow of data from input to output (feedforward) and then the error is passed back to the input (backpropagation) to update the weights and biases (Rahayu & Mustafidah, 2022).

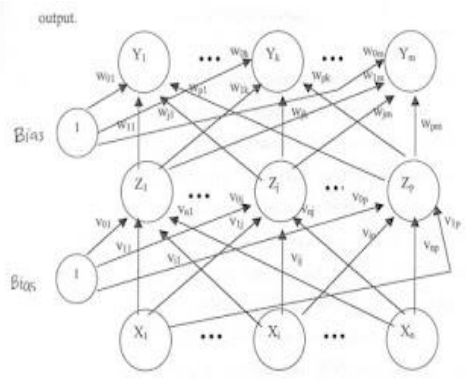

Fig.2 Backpropagation with One Hidden Layer

The architecture of a backpropagation network using one hidden layer unit is shown in the figure. Input layer Xi, hidden layer Zj, and output layer Yk respectively. Each unit has a different weight. Both Vij and Wjk indicate the weight of the input layer unit to the hidden layer unit and the output layer unit respectively.

Nguyen-Widrow Initialization Technique

The Nguyen-Widrow method sets the initial weights and biases in a feedforward network that has one or more hidden layers. This is done so that the initial stimuli for activation of the neurons in the hidden layers are

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

spread evenly across their dynamic range. Evenly distributing the response of the hidden layers for a given input is its main goal to speed up the learning process(Hakim et al., 2021).
This method is used by initializing the neural network with a small random value, which changes the weights and bias. The following formula is used to adjust the bias b and weight W for each neuron located in the hidden layer:

1. The beta (β) value for each hidden layer is calculated based on the number of neurons N inside in that layer: $\beta = 0.7 \times N^{1/I}$ where i is the amount of neurons on the input layers.
2. Normalized each vectors wi from the i neuron so it has the length of vector $\|w_i\|=1$.
3. Do scaling each value of vector that can be normalized with β:
   $w_{i(new)} = \beta \times \frac{wi}{||wi||}$
4. Define the bias b_i for each neurons i on the hidden layer with the value randomly taken from the range of $[-\beta,\beta]$.

The Nguyen-Widrow algorithm changes the biases and weights so that the input space is initially mapped evenly to the activation space of the sigmoid activation function. This allows each neuron to contribute to the output and can cause the network to learn faster. This method is typically used on networks with sigmoidal activation functions. For non-saturating activation functions such as ReLU (Rectified Linear Unit), this method is considered ineffective or irrelevant (Pavelka & Procházka, n.d.). Depending on the method and activation function used, it is one of the initialization methods used in machine learning and is often replaced by newer weight initialization methods (mathworks, 2023).

System Design

This experimental study aims to develop an Artificial Neural Network (ANN) model using an optimized backpropagation algorithm with Nguyen-Widrow initialization. Three main stages are involved in this design: data collection, model development and training, and model validation and testing (Mahfuzh et al., 2020). Each step is divided into steps as follows:

A. Data Collection and Preparation

1. Data Selection: Data will be collected from the electronic medical records of collaborating hospitals within a specific time span to ensure case diversity. The following procedures will be used to support the objectives of this study:

a. Data Sources: To meet the ethical standards of the study, collaboration with hospitals and nephrology clinics will be established to gain access to the renal patient database.
b. Dataset: Information on patient demographics, medical history, clinical laboratory results, and final diagnosis will be part of the dataset. Age, systolic blood pressure, creatinine levels, proteinuria levels, and hemoglobin levels are the specific clinical characteristics captured, which have been identified as important metrics for the evaluation of renal function.

Table 1. Dataset of Patient Diagnosis Results

Age (Year)	Blood Pressure Sistolik (mmHg)	Creatinine Level	Proteinuria Level	Hemoglobin Level	Diagnosis Result
62	175	1,16	1,35	12,5	1
65	169	0,51	0,5	15,97	1
71	157	1,12	0,93	17,14	1
18	142	1,17	1,93	15,87	0
21	141	1,47	1,18	13,21	1
77	133	1,38	1,32	14,18	1
21	145	1,01	1,07	13,2	1
57	165	0,96	0,46	12,17	1
27	138	0,95	0,79	12,11	0
37	140	0,52	1,24	14,01	1
39	144	0,94	0,93	16,81	1
54	113	0,58	0,58	14,08	1
41	145	0,86	1,43	14,95	1
24	161	0,98	0,78	16,89	0
32	146	0,77	0,58	14,14	1

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

42	148	1,38	1,75	14,25	1
20	127	1,42	1,26	16,23	0
36	144	1,37	0,71	15,54	0
19	158	1,07	0,18	17,45	1
76	162	1,02	0,21	12,74	0

Table 2. (Continued) Dataset of Patient Diagnosis Results

Age (Year)	**Blood Pressure Sistolik (mmHg)**	**Creatinine Level**	**Proteinuria Level**	**Hemoglobin Level**	**Diagnosis Result**
56	121	1,37	0,71	15,31	0
51	111	1,42	0,97	16,82	0
67	171	1,42	1,64	15,67	1
54	135	0,54	0,26	16,28	0
41	167	0,78	0,44	12,88	0
65	121	0,51	0,28	14,36	1
32	145	0,34	0,19	12,45	0
35	126	1,15	1,97	12,42	0
41	160	1,34	1,52	12,47	0
26	114	0,76	0,02	17,91	0
37	162	0,9	0,9	15,31	0
29	153	1,05	0,2	17,42	0
68	126	0,66	0,67	11,55	0
53	178	0,87	0,94	11,37	0
33	176	0,65	1,88	14,77	0
65	116	1,07	1,61	16,13	1
33	157	1,2	0,88	17,35	1
35	137	0,73	1,02	15,01	0
68	163	0,99	1,33	14,83	1
56	125	1,26	1,56	16,55	1
32	130	0,9	0,72	15,31	0
43	168	1,34	1,38	14,19	1
67	133	1,14	0,28	15,4	1
69	123	1,19	0,97	14,95	0
47	158	1,34	1,07	15,64	1
37	159	0,95	0,75	14,05	1
37	179	1,48	0,84	13,95	1
37	171	0,62	0,17	16,09	1
57	145	1,27	0,73	11,89	1
50	174	0,91	0,03	12,49	1

In the patient dataset table there are six (6) clinical features and 1 (one) target, namely the Diagnosis Result, where the Diagnosis Result column has a value of 1 (one) and 0 (zero), meaning 1 = Kidney Pain and 0 = No Kidney Pain. The amount of training data is 50 patients.

B. Data Splitting

Retrieve input data and target data from the dataset table as follows.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Table 3. Data Input

Age (Year)	Blood Pressure Sistolik (mmHg)	Creatinine Level	Proteinuria Level	Hemoglobin Level
62	1.750	11.600	13.500	125.000
65	1.190	0.5100	0.5000	159.700
71	1.670	11.200	0.6300	171.400
18	1.420	11.700	19.300	158.700
21	1.410	14.700	11.800	132.100
77	1.330	13.800	13.200	141.800
21	1.450	10.100	10.700	132.000
57	1.650	0.5600	0.4600	121.700
27	1.380	0.950	0.790	121.100
39	1.430	0.520	1.240	141.100
68	1.100	0.940	0.950	166.800
39	1.100	1.440	0.940	171.400
54	1.460	0.860	14.300	149.500
41	1.630	0.980	0.580	148.800
24	1.150	11.900	0.770	160.900
42	1.480	13.800	15.000	142.500
42	1.270	14.200	17.600	113.300
30	1.140	0.720	0.210	155.400
76	1.520	10.700	0.180	174.500
19	1.680	13.700	0.710	127.400
56	1.410	10.100	11.000	156.100
57	1.110	14.200	10.700	164.200
41	1.750	14.200	9.400	158.800
64	1.510	0.580	0.440	172.700
62	1.670	0.780	0.640	146.000
35	1.450	0.510	0.280	123.800
55	1.210	13.400	1.900	124.500
43	1.560	11.500	19.700	124.000
31	1.100	13.400	0.520	147.200
26	1.240	0.760	10.700	139.400
27	1.630	0.900	0.900	173.100
35	1.160	13.400	0.800	124.200
38	1.220	10.500	0.200	154.500
39	1.520	0.660	0.700	115.200
64	1.780	0.870	0.940	141.700
39	1.760	0.650	16.800	163.700
29	1.180	10.700	18.100	147.100
33	1.570	12.000	0.820	173.500
65	1.780	0.790	10.000	160.100
31	1.620	0.930	0.330	145.300
34	1.230	12.600	15.600	165.500
24	1.540	0.940	0.300	158.300

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

32	1.620	11.800	17.200	141.100
27	1.390	0.900	0.500	154.000
46	1.530	0.940	0.750	149.200
34	1.230	0.950	0.280	146.500
37	1.500	0.800	0.900	154.900
47	1.790	14.900	0.750	140.600
69	1.590	11.800	0.840	149.500
57	1.450	0.620	0.700	168.900
52	1.510	0.760	10.300	118.900
32	1.740	12.200	0.730	124.900
50	1.580	0.910	0.800	169.900
37	1.590	0.950	0.750	149.600
37	1.230	0.850	0.800	154.500
69	1.830	14.800	0.290	140.500

Table 5. Target (Diagnose Result)

Age (Year)	Diagnosis Result
62	1
65	1
71	1
18	0
21	1
77	1
21	1
57	1
27	0

Normalization

The process of changing the values of a data set so that they share a common scale without changing the range of values or losing information is known as data normalization. This is important for machine learning and data analysis, especially when working with scale-sensitive methods such as neural networks. Data normalization can speed up convergence during training and improve overall model performance.

Normalization Function

Normalization changes the entered data so that the values are within certain limits, such as [0, 1] or [-1, 1]. It could help in several cases:
Example:
Let's say we have the following dataset:

Feature 1	Feature 2
5	200
10	400
15	600

If we want to normalize this dataset to the range [0, 1], we will use these steps:
1. Calculate the minimum and maximum values for each Fitur.
   - Feature 1: Min = 5, Max = 15
   - Feature 2: Min = 200, Max = 600
2. Apply the normalization formula to each features.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

For each value of x this feature, the value-normalized x^1 calculate using the following formula:

$$x' = \frac{x - \text{Min}}{\text{Max} - \text{Min}}$$

Normalize the Feature 1:
5 becomes (5 - 5) / (15 - 5) = 0
10 becomes (10 - 5) / (15 - 5) = 0.5
15 becomes (15 - 5) / (15 - 5) = 1
Normalize the Feature 2:

200 becomes (200 - 200) / (600 - 200) = 0
400 becomes (400 - 200) / (600 - 200) = 0.5
600 becomes (600 - 200) / (600 - 200) = 1
The dataset that has been normalized is seen like the table below.

Feature 1	Feature 2
0	0
0.5	0.5
1	1

Table 6. Normalized Table of Dataset

Age (Year)	Blood Pressure (mmHg)	Kreatinin Degree	Proteinuria Degree	Hemoglobin Degree
10.027	14.317	0.3956	-0.7591	-11.842
11.785	11.697	-0.1855	-0.6598	-0.7382
15.299	10.601	0.2595	-0.4371	13.765
-14.724	-10.103	0.4297	17.896	0.6756
-13.987	-0.1477	14.512	0.5049	-0.7924
18.813	-0.5194	11.447	0.7447	-0.2571
-13.987	0.0381	-0.1151	0.3165	-0.7929
0.7099	0.9672	-16.473	-0.7283	-13.663
-10.473	-0.2871	-0.3194	-0.1631	-13.995
-0.4615	-0.0084	-17.835	0.6077	-0.3509
0.3444	-15.878	-0.3534	0.1110	11.226
13.542	-15.878	14.852	0.0939	13.765
0.5342	0.0845	-0.6258	0.9331	0.1679
-12.233	-18.843	-0.2178	-0.5228	0.1792
0.2273	0.3756	0.4972	-0.1973	0.7290
-0.1687	0.1775	11.447	10.530	-0.2184
0.1687	-79.781	12.809	14.984	-18.229
-0.8715	-14.020	-11.025	-11.565	0.4935
18.227	0.3633	0.0892	-12.079	15.476
-15.258	0.166	11.007	-0.3001	-15.316
0.6513	-0.1477	-0.1151	0.3679	0.4217
0.7099	154.744	-0.8049	-13.963	10.895
-0.2273	14.317	12.809	18.067	68.811
11.199	0.3168	-15.792	0.4204	14.442
-0.1687	0.601	-0.8982	-0.7626	-0.0253
0.5787	100.381	-181.785	-10.366	-12.504

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

0.5927	-0.768	10.085	-11.908	-12.118
-0.1101	0.5491	0.3616	18.581	-12.184
-0.8130	-15.878	10.985	-0.6255	0.0299
-11.058	-0.9375	-0.9663	0.3165	-0.3895
-10.473	0.8743	-0.4896	0.0253	14.703
0.4030	-10.804	0.0211	-11.736	0.4273
14.127	0.3633	-13.068	-0.3172	-17.085
-0.6373	15.711	-0.5918	0.0939	-18.078
14.127	-13.091	-13.408	13.614	-0.2626
-12.815	15.711	0.0892	15.840	11.502
-0.6958	0.5956	0.5318	-13.792	14.924
11.785	144.856	0.8642	0.2309	0.2010
-15.244	0.8278	-0.3875	-0.9510	-0.0639
0.5701	0.8910	0.7361	-0.5583	10.609
0.6955	-0.5253	0.8808	10.655	0.6535
-11.924	-13.961	-14.097	0.9287	-2.957
-0.1617	0.2130	0.3139	0.0726	-0.7891
-0.3444	-0.0084	-17.835	0.6077	-0.3509
0.4615	161.786	14.852	-0.0774	0.3785
-0.4615	0.378	-0.4830	-13.541	-12.385
0.7099	0.0831	-0.7702	-0.2658	-15.209
0.2999	30.958	0.4556	14.648	-11.897

Building Neural Network (Feedforwardnet)

The feedforwardnet function in MATLAB is used to create a feedforward neural network, which is the most basic type of artificial neural network. In this case, the neural network is created with one hidden layer consisting of 10 neurons, like the picture shown below (*MATLAB: Definisi, Kegunaan, Dan Sistemnya - Glints Blog*, n.d.).

```
1x1 network
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 2
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 10
        sampleTime: 1

    connections:

       biasConnect: [1; 1]
      inputConnect: [1; 0]
      layerConnect: [0 0; 1 0]
     outputConnect: [0 1]

    subobjects:
```

Fig. 3 Neural Network (Feedforwardnet)

Splitting The Dataset

Split the dataset into three main parts: training set, validation set, and testing set. Here are the commands in the Matlab command:

net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

70% of the amount of dataset would be used as the set of training and 15% of the total of dataset also would be used as the set of validation.

Transpose

In data processing and neural networks, the main purpose of the transposition process is to fit the data into the format expected by certain functions and algorithms, especially when used with MATLAB. Matrix transposition, which means flipping rows into columns and vice versa, has several important purposes for data analysis and machine learning. In all these cases, the purpose of transposition is to ensure that the data is managed and processed in the most efficient way, and according to the needs of the analysis or machine learning model being developed, here is a dataset that has been transposed:

Table 6. Transposed Dataset

10.027	11.785	15.299	-15.744	-13.987	18.813
14.317	-11.697	10.601	-0.1013	-0.1477	-0.5194
0.3956	-18.175	0.2595	0.4297	14.512	11.447
-0.7591	-0.6598	-0.4371	17.896	0.5049	0.7447
-11.842	0.7308	13.765	0.6756	-0.7924	-0.2571

Table 7. (Continued) Transposed Dataset

-13.987	0.7099	-10.473	-0.4615	-0.3444	13.542	0.5342
0.0381	0.9672	-0.2871	-0.0084	-15.878	-15.878	0.0845
-0.1151	-16.473	-0.3194	-17.835	-0.3534	14.852	-0.6258
0.3165	-0.7283	-0.1631	0.6077	0.1110	0.0939	0.9331
-0.7979	-13.663	-13.995	-0.3509	11.226	13.765	0.1679

Table 8. (Continued) Transposed Dataset

-12.230	-0.1687	-0.1687	-0.8715	18.227	-15.158	0.6513
-13.556	0.1775	-0.7981	-14.020	0.3633	11.066	-0.1477
0.4978	11.447	12.809	-11.025	0.0892	11.107	-0.1151
-0.1973	10.530	14.984	-11.565	-12.079	-0.3001	0.3679
0.7970	-0.2184	-18.299	0.4935	15.476	-10.518	0.4217

Table 9. (Continued) Transposed Datase

0.7099	-0.2273	11.199	-0.1687	-0.5787	0.5927	-0.1101
-15.414	14.317	0.3168	10.601	0.0381	-10.768	0.5491
12.809	12.809	-15.792	-0.8982	-18.175	10.085	0.3616
-13.963	18.067	-0.4200	-0.7626	-10.366	-11.908	18.581
10.895	0.6811	14.482	-0.0253	-12.504	-12.118	-12.284

Table 10. (Continued) Transposed Dataset

-0.8130	-11.058	-10.473	-0.4030	14.127	-0.6373	14.127
15.878	-0.9375	0.8743	-10.304	0.3633	15.711	-13.091
10.085	-0.9663	-0.4896	0.0211	-13.068	-0.5918	13.614
-0.6255	0.3165	0.0253	-11.736	-0.3172	0.0939	-0.2626
0.0299	-0.3895	14.703	0.4273	-17.085	-18.078	-13.408

Training of Artificial Neural Network

The MATLAB and the Neural Network Toolbox are used to train a neural network, or artificial neural network, with a given dataset. Training is the process of setting the weights and bias of the network based on the input and target data. The goal of training is to reduce the prediction error of the network.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

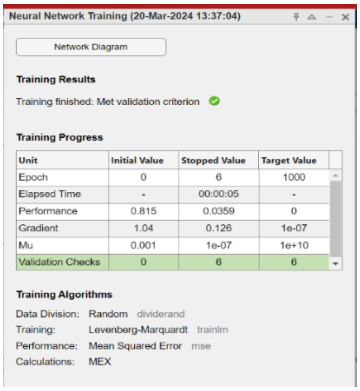

Fig. 4 Training of Artificial Neural Network

RESULTS

According to Fig.4, that shows us about the result of Neural Training Network, we can mention several conclusions as follows:

Training Results:

Training finished:

It indicates the current training status. Validation criterion Met indicates that training stopped because the validation criterion was met, which usually means that the error on the validation set did not decrease over several consecutive epochs (this is a mechanism to prevent overfitting).

Training Progress:

Epoch: It indicates the number of epochs trained during training. Here, the training stops at the sixth (6th) epoch of the targeted 1000 epochs.

Elapsed Time: Time taken for training. This shows that the training lasted for 0.815 seconds before stopping.

Performance: The performance value (usually measured in error such as mean squared error) of the network at the time of training stop. It shows a decrease from the initial value of 0.00916 to 0.00596, which is close to the target value of 0.

Gradient: Magnitude of the gradient of the loss function against the network weights. Training can stop if this gradient is low enough, indicating that learning has converged. The initial value of 0.01 drops to 1e-7, which is lower than the target of 1e-7, indicating good convergence.

Mu: The parameters used for neural network training use Levenberg-Marquardt optimization. To ensure good convergence and an easily adjustable weight update step size. The score changes from 0.001 to the score where the training proces stops.

Training Performance Graph of a Neural Network

The following graph shows how neural network training works. During training, validation, and testing, the Mean Squared Error (MSE) is shown in this graph through the number of epochs that have been passed-in this case, six (6) epochs.

It is explained that the best validation performance is 0.44436 at epoch 0, which indicates that the model has the lowest error on the validation set at the beginning of training. This indicates that the model is good enough with the initialization.

The graph shows training completed at the Met validation criterion. This usually indicates that the premature delay mechanism has been activated. This can happen in MATLAB if the validation error does not improve (or even gets worse) over several consecutive periods. Here, there did not seem to be any significant improvement in the validation error after the first phase; instead, there was a tendency to increase, which caused the training to stop to prevent overfitting (Adzani & Sasongko, 2021; Kurniawan et al., 2019).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13608
e-ISSN : 2541-2019
p-ISSN : 2541-044X

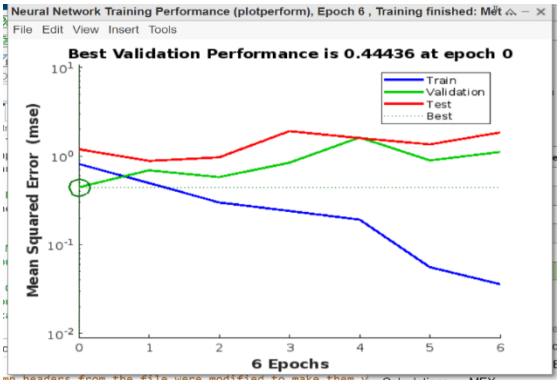

Fig. 5 Training Performance Graph of a Neural Network

DISCUSSION

The selection of appropriate validation criteria in terms of modeling machine learning and artificial neural networks is essential to obtain and build a model and avoid overfitting. Figure 4 shows, when the training process stops because the validation requirements have been met. This indicates that the premature delay mechanism has been activated. This indicates that the model has reached its limit in terms of minimizing the error on the validation set without overfitting. This is important because overfitting can cause the model to not perform well on new data that was not seen during the training process. The training process, shown in the training performance graph, provides significant insight into the dynamics of the learning model. It is clear that the training process was stopped at the 6th epoch out of 1000 targeted epochs, indicating that the training algorithm was effective in detecting when learning no longer provided significant improvement. This informs the authors of the importance of cross-validation process and determination of stopping criteria to avoid redundancy of computational resources and time required.

The decrease in the mean square error (MSE) value from 0.00916 to 0.00596, close to the target of 0, indicates the evaluation of model performance during the training process, but it is worth noting that the significant initial improvement in the best validation at epoch 0 indicates that the model initialization process is very good. The question is about the importance of the advanced training process phase when the model has produced excellent results in the early stage, and the parameter initialization process can affect the speed and efficiency of the training process. It can be concluded that, this study underlines the importance of selecting a suitable training strategy, including the criteria given in the stopping process, in order to ensure that the artificial neural network model not only reduces the training error but also avoids the risk of overfitting. The effectiveness of the premature deferral mechanism in this case demonstrates the value of designing intelligent training algorithms, which can automatically adjust the learning process based on the performance of a given validation and this leads to the creation of robust models that can be relied upon for real-world applications.

REFERENCES

Adzani, W. A., & Sasongko, P. S. (2021). Klasifikasi Status Gizi Balita Menggunakan Metode Backpropagation Dengan Algoritma Levenberg-Marquardt dan Inisialisasi Nguyen Widrow. *JURNAL MASYARAKAT INFORMATIKA*, *12*(1). https://doi.org/10.14710/jmasif.12.1.41020

datacamp. (2024). *Mastering Backpropagation: A Comprehensive Guide for Neural Networks | DataCamp*. https://www.datacamp.com/tutorial/mastering-backpropagation

Hakim, I., Efendi, S., & Sirait, P. (2021). Optimization of The Backpropagation Method with Nguyen-widrow in Face Image Classification. *Randwick International of Social Science Journal*, *2*(2). https://doi.org/10.47175/rissj.v2i2.226

Kurniawan, E., Wibawanto, H., & Widodo, D. A. (2019). Implementasi Metode Backpropogation dengan Inisialisasi Bobot Nguyen Widrow untuk Peramalan Harga Saham. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *6*(1). https://doi.org/10.25126/jtiik.201961904

machinelearningmastery. (2024). *How to Code a Neural Network with Backpropagation In Python (from scratch) - MachineLearningMastery.com*. https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

Mahfuzh, H. F., Widiyanto, D., & Chamidah, N. (2020). Pengaruh Algoritma Inisialisasi Ngiyen-Widrow Terhadap Algoritma Backpropagation Dalam Prediksi Indeks Harga Konsumen (IHK). *Jurnal Seminar Nasional Mahasiswa Bidang Ilmu Komputer Dan Aplikasinya (SENAMIKA)*.

*name of corresponding author

mathworks. (2023). *Nguyen-Widrow layer initialization function - MATLAB initnw*. https://www.mathworks.com/help/deeplearning/ref/initnw.html

MATLAB: Definisi, Kegunaan, dan Sistemnya - Glints Blog. (n.d.). Retrieved March 29, 2024, from https://glints.com/id/lowongan/matlab-adalah/

Matt Mazur. (2023). *A Step by Step Backpropagation*. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

neptune.ai. (2023). *A Comprehensive Guide to the Backpropagation Algorithm in Neural Networks*. https://neptune.ai/blog/backpropagation-algorithm-in-neural-networks-guide

Pavelka, A., & Procházka, A. (n.d.). *ALGORITHMS FOR INITIALIZATION OF NEURAL NETWORK WEIGHTS*.

Rahayu, N., & Mustafidah, H. (2022). Perbandingan Ketepatan Pola Data pada Jaringan Backpropagation Berdasarkan Metode Pembobotan Random dan Nguyen Widrow. *Sainteks*, *19*(1). https://doi.org/10.30595/sainteks.v19i1.12931

solver. (2024). *Training an Artificial Neural Network - Intro | solver*. https://www.solver.com/training-artificial-neural-network-intro

Tiastama, S. A., Laksana, T. G., & Arifa, A. B. (2021). Prediksi Penyakit Ginjal Kronis Menggunakan Hibrid Jaringan Saraf Tiruan Backpropagation dengan Particle Swarm Optimization. *Journal of Innovation Information Technology and Application (JINITA)*, *3*(1), 9–16. https://doi.org/10.35970/JINITA.V3I1.588

Yudi Setyawan. (2016). *View of JARINGAN SYARAF TIRUAN DAN NAIVE BAYES UNTUK MENDETEKSI PENYAKIT GAGAL GINJAL DI RSUD Dr. ADHYATMA TUGUREJO SEMARANG*. https://ejournal.akprind.ac.id/index.php/snast/article/view/1619/1302

*name of corresponding author