J-SISKO TECH
Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD
P:ISSN : 2621-8976 E-ISSN : 2615-5133 ■ 58
Vol.2, No.2, Juli 2019, pp.58-70

Simulasi Permodelan Neural Network dengan Backpropagation pada Pemograman C#

Zahratul Fitri
STKIP Bumi Persada Lhokseumawe
Email: zahratulfitrihumaira@gmail.com

Abstrak

Neural Network atau Jaringan Saraf Tiruan adalah salah satu system tiruan yang memproses informasi yang dilakukan dengan cara mendesaign dengan menirukan cara kerja manusia dalam proses belajarnya sehingga menyelesaikan suatu permasalahan melalui proses belajar pada perubahan bobot sinapsisnya (1).

Penelitian ini mensimulasikan model jaringan saraf tiruan pada suatu simulasi yang terdiri dari satu lapisan input (X1 s/d X120), 2 lapisan tersembunyi/ hidden layer (Z1-1 s/d Z1-241) dan (Z2-1 s/d Z2-241), dan 1 lapisan output dengan 21 neuron.

Hasil simulasi model jaringan saraf tiruan pada inputan learning rate = 0.5, Treshold = 0.5, Setting Error = 0.001 dan Max Epoch = 1000 menghasilkan nilai MSE = 0.24999581738619..

Kata Kunci: Neural Network, Backpropagation, Pemograman C#

1. PENDAHULUAN

Dalam kehidupan sehari – hari sering dilakukan proses identifikasi, mulai dari pelayanan perbankan, akademisi, kesehatan, kependudukan dan lain sebagainya. Ada banyak cara untuk melakukan proses tersebut, seperti menggunakan tanda tangan, finger print, face print dsb. Secara manual proses indentifikasi tersebut biasa digunakan seperti mencocokkan dan membandingkan tanda tangan yang didapat pada saat transaksi dengan spesimen yang sah yang sebelumnya sudah disimpan (2). Untuk membantu mempermudah pengenalan pola tanda tangan dapat menggunakan pola computer, yaitu dengan penerapkan jaringan saraf tiruan. Jaringan saraf tiruan dihasilkan oleh keluaran atau kesimpulan yang ditarik oleh jaringan berdasarkan kepada pengalaman selama mengikuti proses pembelajaran (3).

Dalam penelitian ini menggunakan jaringan saraf tiruan bacpropagation yang mensimulasikan formulasi XOR dengan membangun system pola sehingga menghasilkan nilai MSE yang mendekati nilai pembelajaran. Metode backpropagation telah banyak digunakan dalam model jaringan saraf tiruan, bahkan banyak aplikasi yang telah dibangun diantarannya adalah dalam bidang financial, pola tulisan tangan, pola audio , pola video system kendali dan pengolahan citra digital (4).

Penelitian ini menerapkan jaringan saraf tiruan backpropagation dalam menyelesaikan operasi bolean XOR. Penyelesaian ini mula – mula dibentuk sebuah model simulasi arsitektur neural network yang terdiri dari 1 lapisan input, 2 lapisan tersembunyi, dan 1 lapisan output.

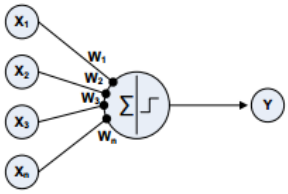
Dalam penyelesaiannya, simulasi ini menggunakan Bahasa pemograman C#, karena Bahasa pemograman tersebut sangat mudah dipahami bagi semua kalangan, baik programmer baru maupun yang telah lama berkecimpung dalam dunia IT. Simulasi ini nantinya diharapkan dapat membangun keilmuan khususnya pada jaringan saraf tiruan backpropagation sehingga menjadi refrensi bagi peneliti awal yang sedang mendalami ilmu neural network backpropagation.

1. LANDASAN TEORI

a. Jaringan saraf tiruan backpropagation

Jaringan saraf tiruan adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh system sel saraf biologi sama seperti otak yang memproses suatu informasi (5). Jaringan saraf tiruan sama seperti halnya manusia yang belajar dari suatu contoh untuk memecahkan suatu masalah yang memiliki pola yang sama dengan contoh yang diberikan. Jaringan saraf tiruan merupakan metode pembelajaran yang bias digunakan untuk menyelesaikan permasalahan yang bernilai diskrit, real maupun vector (5).

Jaringan syaraf tiruan merupakan algoritma pembelajaran yang meniru cara kerja sel syaraf. Selama proses pembelajaran, bobot-bobot dan bias selalu diperbaharui menggunakan algoritma belajar, jika ada error pada keluaran. Untuk proses identifikasi, bobot-bobot yang secara langsung memboboti masukan inilah yang dinamakan sebagai parameter yang dicari, seperti terlihat pada Gambar 2.1, parameter yang dicari adalah harga W1, W2, W3 dan Wn. Dalam identifikasi secara on-line, neuron ataupun jaringan neuron akan selalu ‘belajar’ setiap ada data masukan dan keluaran.



Gambar 2.1: Proses komunikasi antar neuron

Gambar 2.1 memperlihatkan bahwa NN terdiri atas satuan-satuan pemroses berupa neuron. Y sebagai output menerima input dari neuron X1, X2, X3,…, Xn dengan bobot W1, W2, W3, …, Wn. Ketiga impuls neuron yang ada dijumlahkan yaitu net = $x_1w_1 + x_2w_2 + x_3w_3$. Konsep jaringan saraf tiruan bermula pada makalah. Hasil penjumlahan seluruh impuls neuron dibandingkan dengan nilai ambang tertentu melalui fungsi aktivasi f setiap neuron. Fungsi aktivasi digunakan sebagai penentu keluaran suatu neuron.

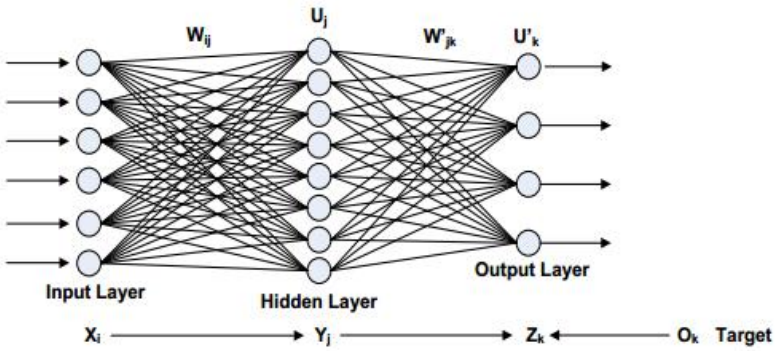
Jaringan saraf tiruan backpropagation merupakan salah satu model dari jaringan saraf tiruan umpan maju dengan menggunakan pembelajran terawasi yang disusun berdasarkan pada algoritma errorbackprogation yang didasarkan pada aturan pembelajaran koreksi kesalahan (error correction learning rule). Arsitekur jaringan saraf tiruan backpropagation terdiri dari unit masukan, unit layer tersembunyi dan unit keluaran.

Algoritma jaringan saraf tiruan backpropagation dapat dibagi ke dalam dua bagian, yaitu (5) :

1) Algoritma pelatihan, terdiri dari tiga tahap yaitu tahap umpan maju pola pelatihan input, tahap mempropagasibalikkan error dan tahap pengaturan bobot.
2) Algoritma aplikasi atau pengujian yang digunakan hanyalah umpan maju saja.

Salah satu metode pelatihan dalam neural network adalah pelatihan terbimbing (supervised learning). Backpropagation Neural Network merupakan salah satu metode yang menggunakan supervised learning. Pada pelatihan terbimbing diperlukan sejumlah masukan dan target yang berfungsi untuk melatih jaringan hingga diperoleh bobot yang diinginkan. Pada setiap kali pelatihan, suatu input diberikan ke jaringan, kemudian jaringan akan memproses dan mengeluarkan keluaran. Selisih antara keluaran jaringan dengan target merupakan kesalahan yang terjadi, dimana jaringan akan memodifikasi bobot sesuai dengan kesalahan tersebut.

Pada supervised learning terdapat pasangan data input dan output yang dipakai untuk melatih Jaringan saraf tiruan hingga diperoleh bobot penimbang (weight) yang diinginkan. Penimbang itu sendiri adalah sambungan antar lapis dalam Jaringan saraf tiruan. Algoritma ini memiliki proses pelatihan yang didasarkan pada interkoneksi yang sederhana, yaitu apabila keluaran memberikan hasil yang salah, maka penimbang dikoreksi agar galat dapat diperkecil dan tanggapan Jaringan saraf tiruan selanjutnya diharapkan dapat mendekati nilai yang benar. Backpropagation Neural Network juga berkemampuan untuk memperbaiki penimbang pada lapis tersembunyi (hidden layer).



Gambar 2.2. Lapis dan Aliran Sinyal dalam algoritma Backpropagation Neural Network

Secara garis besar BPNN terdiri atas tiga lapis (layer) yaitu lapis masukan (input layer) xi, lapis tersembunyi (hidden layer) yj, dan lapis keluaran (output layer) zk. Lapis masukan dan lapis tersembunyi dihubungkan dengan penimbang wij dan antara lapis tersembunyi dan lapis keluaran dihubungkan oleh penimbang w'jk. Pada dasarnya, metode pelatihan backpropagation terdiri dari tiga langkah, yaitu :

1) Data dimasukkan kedalam input node atau jaringan (feedforward);
2) Perhitungan dan propagasi balik dari error yang bersangkutan;
3) Pembaharuan (adjustment) bobot dan bias.

b. Operator Logika/ Bolean

Operand adalah nilai asal yang digunakan di dalam sebuat proses operasi. Sedangkan Operator adalah instruksi yang diberikan untuk mendapatkan hasil dari proses tersebut. Biasanya operator berupa karakter matematis atau perintah singkat sederhana. Sebagai contoh adalah : 8 x 5. Angka 8 dan 5 disebut Operand sedangkan tanda x disebut Operator.

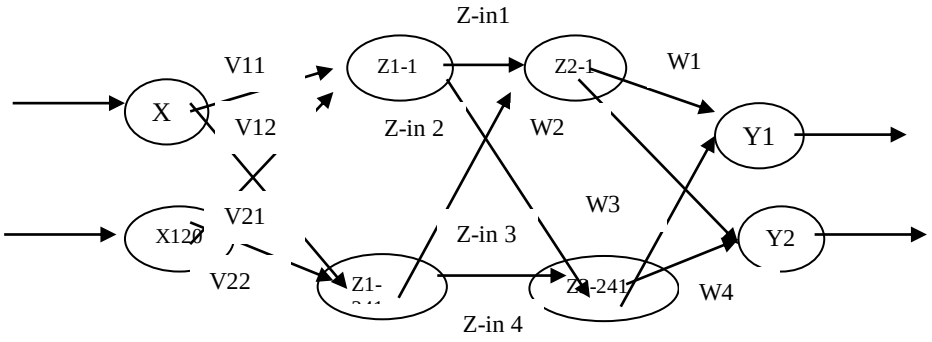
Operator logika digunakan untuk menghasilkan nilai bolean TRUE atau FALSE dari dua kondisi dijelaskan pada table berikut:

Tabel 2.1. Operator Logika nilai Bolean TRUE atau FALSE

No.	Operator	Penjelasan	Contoh
1.	And	Akan menghasilkan TRUE jika kedua operand TRUE	TRUE dan FALSE, hasilnya: FALSE
2.	Or	Akan menghasilkan TRUE jika salah satu operand TRUE	TRUE or FALSE, hasilnya: TRUE
3.	Xor	Akan menghasilkan TRUE jika kedua operand berbeda	TRUE Xor FALSE hasilnya: TRUE
4.	Not	Akan menghasilkan TRUE jika operand FALSE	Not TRUE, hasilnya: FALSE

2. ANALISA DAN HASIL

Pada Bab ini dijelaskan simulasi arsitektur neural network yang disajikan pada gambar berikut:



Gambar 3.1. Arsitektur Neural Network

Terdiri dari :
- 1 lapisan input (X1 s/d X120)
- 2 lapisan tersembunyi / hidden layer (Z1-1 s/d Z1-241) dan (Z2-1 s/d Z2-241)
- 1 lapisan output dengan 21 Neuron

Simulasi :
bobot awal yang menghubungkan neuron-neuron pada lapisan tersembunyi (V11 s/d V12 ; V21

s/d V22 ; W1 s/d W4) dipilih secara acak pada range 0 s/d 1.
serta input sudah ditetapkan sbb :

X1	X2
0	0
0	1
1	0
1	1

sedangkan TARGET menjadi masukan yang bebas ditentukan oleh pengguna.

Algoritma pemrograman :

1. Bangkitkan bilangan acak (random) untuk bobot awal (V11 s/d V12 ; V21 s/d V22 ; W1 s/d W4)
2. Operasi pada hidden layer:
   - Penjumlahan terbobot : hitung Zin1 s/d Zin4
   - Pengaktifan : hitung Z1 s/d Z2
3. Operasi pada Output Layer
   - Perkalian : hitung Yin
   - Pengaktifan : hitung Y
4. Hitung Eror dan Kuadrat Eror (MSE/Mean Square Error)
5. Hitung Faktor Eror (dho)
6. Hitung Perubahan Bobot W (delta w) : dw1 s/d dw4
7. Hitung Perubahan Eror (dho in) : dho in1 s/d dho in4
8. Hitung Factor : dho1 s/d dho4
9. Hitung Perubahan Bobot V (dv)
10. Hitung Perubahan Bobot Bias (db)
11. Bobot Akhir Input (V) sudah ditemukan
12. Bobot Akhir Hidden Layer ke Output (W) sudah ditemukan.

    langkah 1 s/d 4 baru digunakan untuk mengitung bobot akhir pada data pertama (X1 = 0 dan X2 = 0)
    Pada data kedua, juga dilakukan operasi perhitungan yang sama dengan menggunakan bobot-bobot akhir hasil pengolahan data pertama sebagai bobot awal pada data kedua... begitu seterusnya sampai data ke-empat.

Proses perhitungan pada data ke-1 s/d data ke-4 ini dilakukan secara berulang terus sampai tercapai meksimum epoh atau MSE (kuadrat error < terget eror)

Berikut ini adalah Hasil uji yang menggunakan bahasa pemograman C# :

Source Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading.Tasks;
using System.Threading;

namespace NNBackpropagation
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      double[,] acak = new double[4, 5];
      double v11_a, v12_a, v21_a, v22_a, v01_a;
      double w1_a, w2_a, w3_a, w4_a;
      double alpha, t_eror, epoh, max_epoh, treshold;
      double[] x1 = new double[4] { 0, 0, 1, 1 };
      double[] x2 = new double[4] { 0, 1, 0, 1 };

      double[] zin1 = new double[4]; double[] zin2 = new double[4];
      double[] zin3 = new double[4]; double[] zin4 = new double[4];

      double[] z1 = new double[4]; double[] z2 = new double[4];
      double[] z3 = new double[4]; double[] z4 = new double[4];

      double[] dw1 = new double[4]; double[] dw2 = new double[4];
      double[] dw3 = new double[4]; double[] dw4 = new double[4];

      double[] dB = new double[4];

      double[] dhoin1 = new double[4]; double[] dhoin2 = new double[4];
      double[] dhoin3 = new double[4]; double[] dhoin4 = new double[4];
```

```
double[] dho1 = new double[4]; double[] dho2 = new double[4];
double[] dho3 = new double[4]; double[] dho4 = new double[4];

double[] dv11 = new double[4]; double[] dv12 = new double[4];
double[] dv13 = new double[4]; double[] dv14 = new double[4];
double[] dv21 = new double[4]; double[] dv22 = new double[4];
double[] dv23 = new double[4]; double[] dv24 = new double[4];

double[] db1 = new double[4]; double[] db2 = new double[4];
double[] db3 = new double[4]; double[] db4 = new double[4];

double[] v11 = new double[4]; double[] v12 = new double[4];
double[] v13 = new double[4]; double[] v14 = new double[4];
double[] v21 = new double[4]; double[] v22 = new double[4];
double[] v23 = new double[4]; double[] v24 = new double[4];

double[] v01 = new double[4]; double[] v02 = new double[4];
double[] v03 = new double[4]; double[] v04 = new double[4];

double[] w1 = new double[4]; double[] w2 = new double[4];
double[] w3 = new double[4]; double[] w4 = new double[4];
double[] wb = new double[4];

double[] Yin = new double[4];
double[] Y = new double[4];
double[] eror = new double[4];
double[] MSE = new double[4];
double[] dho = new double[4];
double[] T = new double[4];
double[] uji_Yin = new double[4];
double[] uji_Y = new double[4];
double[] uji_T = new double[4];
int a, b;


double rand(double min, double max)
{
   Random random = new Random();
   return random.NextDouble() * (max - min) + min;
}

//** tombol Calculasi untuk nilai bobot akhir
private void button1_Click(object sender, EventArgs e)
{
```

```
//** mengambil nilai dari input masukan
 T[0] = double.Parse(textBox1.Text.ToString());
 T[1] = double.Parse(textBox2.Text.ToString());
 T[2] = double.Parse(textBox3.Text.ToString());
 T[3] = double.Parse(textBox4.Text.ToString());
 alpha = double.Parse(textBox5.Text.ToString());
 t_eror = double.Parse(textBox7.Text.ToString());
 max_epoh = double.Parse(textBox8.Text.ToString());
 treshold = double.Parse(textBox6.Text.ToString());

 //** kalkulasikan nilai yang akan dimunculkan pada bobot akhir
 for (epoh = 0; epoh < max_epoh; epoh++)
 {
    for (a = 0; a < 4; a++)
    {
       // 1. Operasi pada Hiden Layer - Penjumlahan Terbobot (Zin)
       zin1[a] = v01_a + v11_a * x1[a] + v21_a * x2[a];
       zin2[a] = v01_a + v11_a * x1[a] + v21_a * x2[a];
       zin3[a] = v01_a + v12_a * x1[a] + v22_a * x2[a];
       zin4[a] = v01_a + v12_a * x1[a] + v22_a * x2[a];

       // 2. Pengaktifan (Z)
       z1[a] = 1 / (1 + (Math.Exp(-zin1[a])));
       z2[a] = 1 / (1 + (Math.Exp(-zin2[a])));
       z3[a] = 1 / (1 + (Math.Exp(-zin3[a])));
       z4[a] = 1 / (1 + (Math.Exp(-zin4[a])));

       // 3. Operasi Pada Output Layer (Yin)
       Yin[a] = w1_a * z1[a] + w2_a * z2[a] + w3_a * z3[a] + w4_a * z4[a];
       Y[a] = 1 / (1 + (Math.Exp(-Yin[a])));

       // 4. Eror dan MSE (Mean Square Eror / jumlah kuadrat eror)
       eror[a] = T[a] - Y[a];
       MSE[a] = Math.Pow(T[a] - Y[a], 2);

       // 5. Faktor Eror (dho)
       dho[a] = (T[a] - Y[a]) * Y[a] * (1 - Y[a]);

       // 6. dw
       dw1[a] = alpha * dho[a] * z1[a];
       dw2[a] = alpha * dho[a] * z2[a];
       dw3[a] = alpha * dho[a] * z3[a];
       dw4[a] = alpha * dho[a] * z4[a];
      //  dB[a] = alpha * dho[a]; --> hitung W0
```

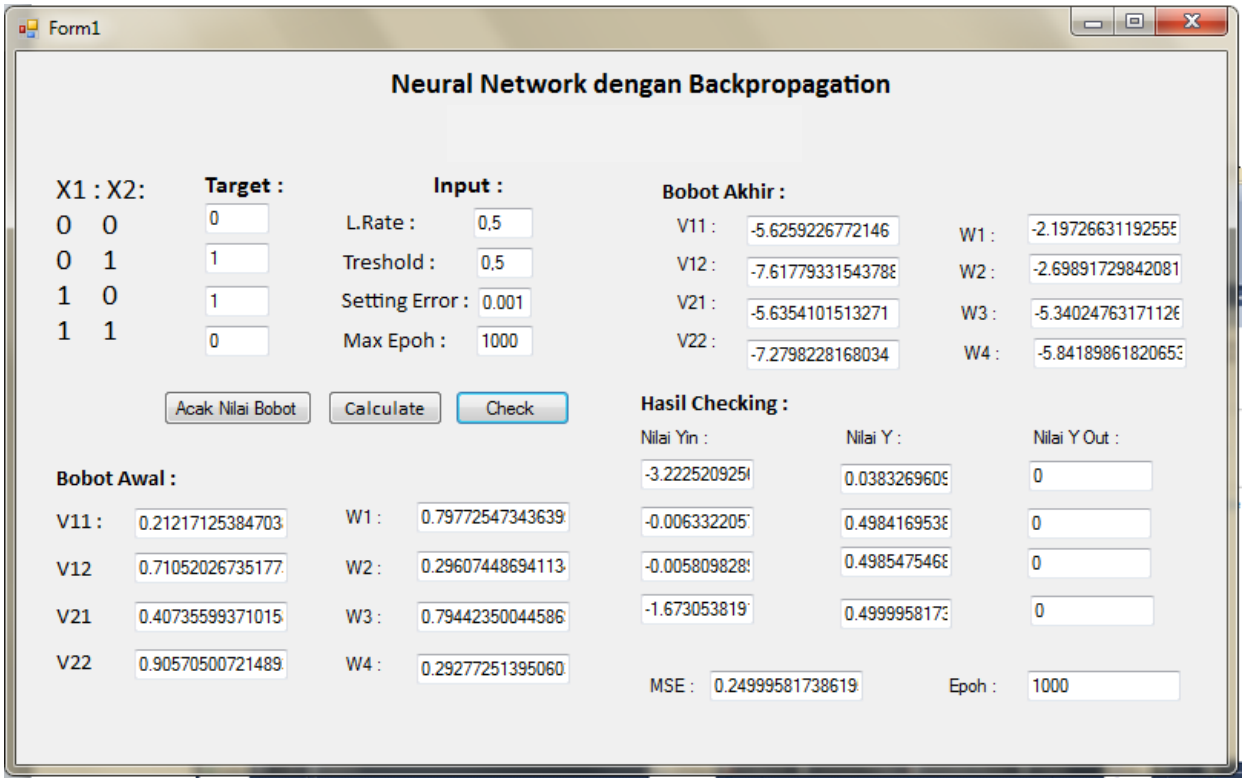
```
// 7. Perubahan Kesalahan (dho in)
dhoin1[a] = dho[a] * w1_a;
dhoin2[a] = dho[a] * w2_a;
dhoin3[a] = dho[a] * w3_a;
dhoin4[a] = dho[a] * w4_a;

// 8. Factor (dho)
dho1[a] = z1[a] * dhoin1[a] * (1 - z1[a]);
dho2[a] = z2[a] * dhoin2[a] * (1 - z2[a]);
dho3[a] = z3[a] * dhoin3[a] * (1 - z3[a]);
dho4[a] = z4[a] * dhoin4[a] * (1 - z4[a]);

// 9. Perubahan Bobot V (dv)
dv11[a] = alpha * x1[a] * dho1[a];
dv12[a] = alpha * x1[a] * dho2[a];
dv13[a] = alpha * x1[a] * dho3[a];
dv14[a] = alpha * x1[a] * dho4[a];
dv21[a] = alpha * x2[a] * dho1[a];
dv22[a] = alpha * x2[a] * dho2[a];
dv23[a] = alpha * x2[a] * dho3[a];
dv24[a] = alpha * x2[a] * dho4[a];

// 10. Perubahan Bobot Bias (db)
db1[a] = alpha * dho1[a];
db2[a] = alpha * dho2[a];
db3[a] = alpha * dho3[a];
db4[a] = alpha * dho4[a];

// 11. Bobot Akhir Input (V)
v11[a] = v11_a + dv11[a];
v12[a] = v12_a + dv12[a];

v21[a] = v21_a + dv21[a];
v22[a] = v22_a + dv22[a];
// Bobot Akhir Bias
v01[a] = v01_a + db1[a];

// 12. Bobot Akhir Hidden ke Output
w1[a] = w1_a + dw1[a];
w2[a] = w2_a + dw2[a];
w3[a] = w3_a + dw3[a];
w4[a] = w4_a + dw4[a];

// Bobot Akhir Bias ke Output
v11_a = v11[a]; v12_a = v12[a];
```

```
            v21_a = v21[a]; v22_a = v22[a];
            v01_a = v01[a];
                        w1_a = w1[a]; w2_a = w2[a]; w3_a = w3[a]; w4_a = w4[a];
        }
        if (MSE[3] < t_eror)
        {

            break;
        }
    }

    //Menampilkan Bobot Akhir
    textBox21.Text = w1[3].ToString();
    textBox22.Text = w2[3].ToString();
    textBox23.Text = w3[3].ToString();
    textBox24.Text = w4[3].ToString();

    textBox49.Text = MSE[3].ToString();
    textBox50.Text = epoh.ToString();

    textBox17.Text = v11[3].ToString();
    textBox18.Text = v12[3].ToString
        textBox19.Text = v21[3].ToString();
    textBox20.Text = v22[3].ToString();
}

//** tombol acak untuk mencari bobot awal
private void button4_Click(object sender, EventArgs e)
{
    for (a = 0; a < 4; a++)
    {
        for (b = 0; b < 5; b++)
        {
            Thread.Sleep(20);
            acak[a, b] = rand(0, 1);
        }
    }


    //**Menampilkan Bobot awal secara acak
    // menampilkan nilai v11-v22 kedalam textbox
    v11_a = acak[0, 0]; textBox9.Text = v11_a.ToString();
    v12_a = acak[1, 0]; textBox10.Text = v12_a.ToString();
    v21_a = acak[0, 1]; textBox11.Text = v21_a.ToString();
    v22_a = acak[1, 1]; textBox12.Text = v22_a.ToString();
```

```
        v01_a = acak[0, 2];

        // menampilkan nilai w1-w4 kedalam textbox
        w1_a = acak[0, 3]; textBox13.Text = w1_a.ToString();
        w2_a = acak[1, 3]; textBox14.Text = w2_a.ToString();
        w3_a = acak[2, 3]; textBox15.Text = w3_a.ToString();
        w4_a = acak[3, 3]; textBox16.Text = w4_a.ToString();



    }


    //** tombol check hasil Y
    private void button3_Click_1(object sender, EventArgs e)
    {
        for (b = 0; b < 4; b++)
        {
uji_Yin[b] = wb[b] + z1[b] * w1[b] + z2[b] * w2[b] + z3[b] * w3[b] + z4[b] * w4[b];
uji_Y[b] = 1 / (1 + (Math.Exp(-uji_Yin[b])));

            if (uji_Y[b] >= treshold)
            {
                uji_T[b] = 1;
            }
            else
            {
                uji_T[b] = 0;
            }
        }

        //**Menampilkan Hasil Check

        //menampilkan hasil Y Out

        textBox41.Text = uji_T[0].ToString();
        textBox42.Text = uji_T[1].ToString();
        textBox43.Text = uji_T[2].ToString();
        textBox44.Text = uji_T[3].ToString();

        //menampilkan hasil Yin0 -Yin3
        textBox25.Text = uji_Yin[0].ToString();
        textBox26.Text = uji_Yin[1].ToString();
        textBox27.Text = uji_Yin[2].ToString();
        textBox28.Text = uji_Yin[3].ToString();
```

```
            //menampilkan hasil Y0-Y3
            textBox33.Text = uji_Y[0].ToString();
            textBox34.Text = uji_Y[1].ToString();
            textBox35.Text = uji_Y[2].ToString();
            textBox36.Text = uji_Y[3].ToString();
        }

        }
}
```

Screen Shoot Program adalah sebagai berikut :



Gambar 3.2 Output Program simulasi Neural Network backpropagation pada operasi logika XOR

3. KESIMPULAN

Berdasarkan hasil yang telah dicapai selama perancangan, implementasi dan pengujian perangkat lunak, maka dapat ditarik kesimpulan sebagai berikut:

1. Implementasi jaringan saraf tiruan backpropagation pada operasi logika XOR berhasil diimplementasikan dengan menggunakan Bahasa pemograman c#.
2. Kesimpulan yang diperoleh dari hasil pembelajaran neural network backpropagation adalah dengan menggunakan operasi logika XOR pada inputan Learning Rate = 0.5, nilai Treshold

= 0.5, Setting Error = 0.001 dan Max Epoch =1000 diperoleh bahwa nilai MSE mencapai 0.24999581738619.

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih kepada suami tercinta Muhammad Faisal selaku Pakar juga dalam bidang IT yang telah banyak memberikan bimbingan sehingga penelitian ini dapat terselesaikan dengan baik.

REFERENSI

1. Fitriani Ma, Mustafidah H. Analisis Perbandingan Algoritma Teknik Heuristik Pada Jaringan Syaraf Tiruan Metode Pembelajaran Backpropagation. :8.
2. Abidin Z. Metoda Backpropagation Neural Network Untuk Mengelompokkan Pola Huruf Tertentu Dalam Bentuk Vektor. 2006;123.
3. Salman Ag. Implementasi Jaringan Syaraf Tiruan Recurrent Dengan Metode Pembelajaran Gradient Descent Adaptive Learning Rate Untuk Pendugaan Curah Hujan. 2011;8.
4. Wardani S, Ramadhan S. Analisis Sistem Pendukung Keputusan Menggunakan Metode Moora Untuk Merekomendasikan Alat Perekam Suara. 2019;2(1):9.
5. Sukarno Nm, Wirawan Pw, Adhy S. Perancangan Dan Implementasi Jaringan Saraf Tiruan Backpropagation Untuk Mendiagnosa Penyakit Kulit. Jurnal Masyarakat Informatika. 2015 Oct 31;5(10):9–18.