UNIVERSITAS INDONESIA

SISTEM BERBASIS PENGETAHUAN

LAPORAN PROYEK JARINGAN SYARAF TIRUAN MENGGUNAKAN ALGORITMA PROPAGASI BELAKANG

Ahmad Akbar Habibilah (1806147804)
George (1806194883)
Hansel Matthew (1806194914)
Kemas Muhammad Rizki Fadhila (1806195072)

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO

DEPOK
APRIL 2021

DAFTAR ISI

BAB I
PENDAHULUAN

2.1 Latar Belakang

Sistem kendali adalah cabang ilmu yang fokus mempelajari pengendalian respon dari sebuah sistem. Sistem tersebut dapat datang dari banyak ruang lingkup seperti pada sistem proses kimia, produksi tenaga listrik, bidang manufaktur, produksi minyak dan gas, dan masih banyak lagi. Sistem kendali pertama yaitu sistem kendali klasik PID dikembangkan pada tahun 1940 untuk pengendali dalam proses industri. 90% dari lup pengendalian menggunakan pengendali PID (Proporsional – Integral – Diferensial). Sistem kendali klasik ini Sebagian besar masih bekerja secara manual dengan parameter ditentukan secara trial and error. Penalaan ulang dari parameter ini menghabiskan waktu yang lama serta memakan biaya yang tinggi. Sering sekali industri menggunakan pengendali dengan nilai parameter *default* sehingga memberikan kinejar yang tidak sama untuk perubahan dari titik operasi.

Seiring berkembangnya zaman, sistem kendali menjadi semakin dibutuhkan, hal itu dikarenakan kemampuan manusia tidak dapat lagi mengalahkan kemampuan mesin dalam hal mengendalikan dengan akurat dan presisi. Sistem kendali pun mengalamni beberapa perkembangan menjadi berbagai jenis seperti optimal control, robust control, non linear control, dan lain lain. Salah satu perkembangannya adalah intelligent control, yaitu suatu pengendali adaptif yang berbasis pengetahuan. Terciptalah suatu sistem yang memiliki cara kerja mirip dengan sistem kerja jaringan saraf manusia yang dinamakan *Artificial Neural Network. Artificial Neural Network*. *Artificial Neural Network* (ANN) atau Jaringan Saraf Tiruan adalah suatu sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Sistem ini tidak menggunakan sebuah pemodelan matematis karena menggunakan pemodelan *blackbox*. Pemodelan ini dilakukan dengan cara menganalisa perilaku sistem jika diberikan variasi pasangan input dan output. Dari perilaku inilah maka jaringan saraf tiruan akan berusaha untuk menghasilkan output yang diinginkan.

2.2 Rumusan Masalah

a. Bagaimana konsep dasar jaringan saraf tiruan?
b. Bagaimana cara kerja metode pembelajaran jaringan saraf tiruan dengan menggunakan algoritma *backpropagation*?
c. Bagaimana suatu jaringan saraf tiruan dapat mengklasifikasikan suatu dataset?

2.3Tujuan Penelitian

Berdasarkan permasalahan yang telah dipaparkan, maka tujuan dari laporan ini adalah untuk menganalisis dan mempelajari cara kerja dari jaringan saraf tiruan pada dataset yang diberikan.

BAB II
DASAR TEORI

2.1Artificial Neural Network

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan lahir dari usaha manusia yang ingin memodelkan otak manusia untuk membuat suatu sistem yang paling sempurna. Model ANN pertama kali dikenalkan oleh Mc. Culloh dan Pitts sebagai komputasi dari aktivitas syarat. Hasil penelitian mereka menjadi dasar bagi penelitian di bidang ANN pada masa berikutnya.

Pada tahun 1958, Rosenblat, Widrow dan Hoff pertama kali menemukan aturan pembelajaran pada perceptrons. Minski dan Papert (1969) meyakini bahwa penggunaan perceptrons sangat terbatas yaitu hanya sebagai metode perhitungan dalam kehidupan nyata. Bernard Widrow menemukan neuron sederhana yang mirip dengan perceptron yang disebut ADALINE (neuron linier adaptif), dan jaringan multilayernya yang disebut MADALINE (multiple adaline). Selanjutnya, Widrow juga mengembangkan program pembelajaran terbimbing yang disebut dengan metode pembelajaran Least Mean Square (LMS) atau Widrow Hoff. Di era berikutnya, jaringan syaraf tiruan berkembang sedemikian rupa sehingga menemukan berbagai metode pembelajaran dan aturan pembelajaran.

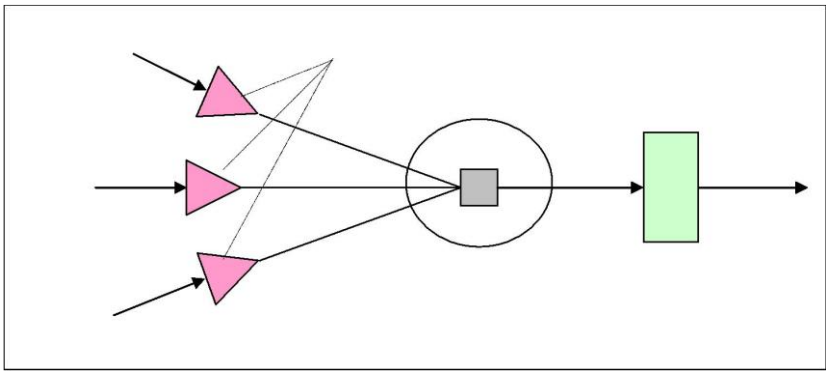
ANN merupakan jaringan yang dibuat dengan meniru jaringan syaraf manusia dengan diilhami oleh struktur dan cara kerja otak dan sel syaraf manusia. Sebuah neuron memiliki sebuah badan sel, pengirim sinyal dan penerima sinyal. Neuron biologis pada manusia dapat dilihat pada gambar 2.1

Gambar 2.1. Ilustrasi Neuron Biologis Manusia

Cara kerja dari sebuah neuron adalah akan bereaksi apabila potensial listrik mencapai suatu batasan tertentu. Neuron akan menjumlahkan sinyal yang masuk melalui dendrite yang dikalikan dengan pembobot sinapsis[1]. Proses pembelajaran terjadi dengan perubahann yang

terjadi pada sinasis. Sinyal yang masuk akan dijumlahkan dan dikonversi dengan suatu fungsi aktivitas yang kemudian akan mengeluarkan suatu sinyal pemicu yang dialirkan ke neuron lain.

Prinsip kerja dari neuron ini kemudian dimodelkan secara matematis. Model matematik orde pertama dari neuron dapat dilihat pada Gambar 2.2



Gambar 2.2. Model Matematik dari Neuron

Pemodelan matematika inilah yang menjadi dasar dari ANN. Elemen dasar dari sebuah ANN adalah sebuah neuron. Neuron ini akan mengubah sinyal msukan menjadi sebuah keluaran. Setiap neuron mempunyai suatu inputan yang memiliki bobotnya masing – masing. Sinyal kemudian akan dikonversi menggunakan sebuah fungsi aktivasi. Fungsi aktivasi yang digunakan dapat beragam seperti *Relu, sigmoid, tanh* dan lain lain

2.2 *Backpropagation*

Backpropagation adalah suatu algoritma ANN yang memiliki kekuatan utama pada klasifikasi suatu pola atau disebut sebagai *pattern recognition*. Jaringan syaraf *backpropagation* dapat digunakan untuk memprediksi luaran dari sebuah sistem. Pada mulanya data akan diberikan pada sistem. Jika sistem menghasilkan luaran yang tidak sesuai dengan yang diharapkan, maka *backpropagation* akan memodifikasi bobot pada hubungan neuron.[2]

Backpropagation menggunakan gradient descent untuk mengoptimasi nilai dari bobot tersebut. Gradient descent bertujuan adalah untuk terus mengambil sampel gradien parameter model ke arah yang berlawanan berdasarkan bobot w serta memperbarui secara konsisten hingga ANN dapat mencapai fungsi minimum global J (w). Gradient descent memiliki suatu parameter yaitu Learning Rate (α) yang mengatur seberapa besar perubahan bobot pada setiap epoch.

Pada gradient descent dikenal juga istilah momentum. Momentum adalah suatu metode agar mempertahankan arah dari gradient descent. Hal ini diperoleh dengan mengkombinasikan

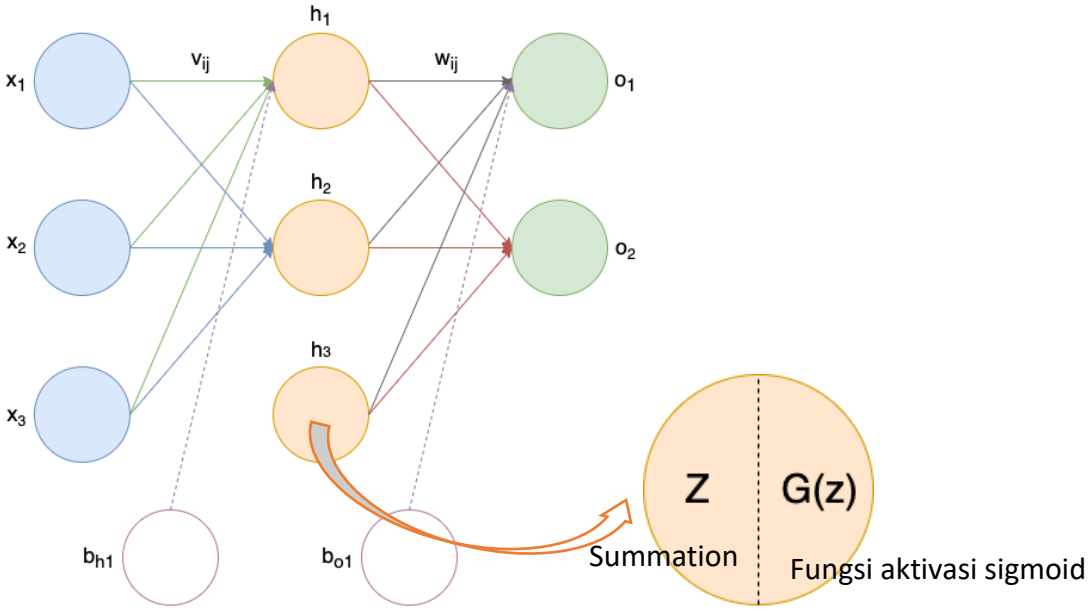
arah yang dikomputasi pada iterasi sekarang dengan iterasi sebelumnya. Besar pengaruh arah iterasi sebelumnya dapat diatur dengan mengatur koefisien momentum (μ**).**

ANN terdiri dari 3 layer, yaitu *input layer* yang memiliki neuron sebanyak I buah, *hidden layer* yang memiliki neuron sebanyak J buah, dan *output layer* yang memiliki neuron sebanyak K buah. Antara layer dihubungkan dengan bobot. Antara *input layer* dengan *hidden layer* dihubungkan dengan bobot **w**. Antara layer *hidden layer* dengan *output layer* dihubungkan dengan bobot **v**. Pada layer input dan hidden ditambah sebuah neuron bobot bias, yang berguna sebagai konstanta.[3]

Algoritma *backpropagation* dapat dibagi menjadi dua buat proses yaitu training dan testing. Dimana kedua proses tersebut dapat dibagi lagi menjadi:

1. Training
   a. Proses pengolahan data input (*feedforward*).
   b. Perhitungan error (*backpropagation*).
   c. Pembaruan bobot.
2. Testing
   a. Pengelompokkan nilai output (*Quantizing*).
   b. Perhitungan *Recognition Rate*.

Diagram sederhana pemodelan jaringan saraf tiruan yang digunakan dapat dilihat pada Gambar 2.3.



Gambar 2.3. Struktur Neural Networks Sederhana

Keterangan:

∂ x → Input layer ∂ h → Hidden layer

∂ o → Output layer

∂ i → Ukuran input layer

∂ j → Ukuran hidden layer

∂ v → Bobot hidden layer

∂ w → Bobot output layer

Dalam konteks machine learning, hyperparameter adalah parameter yang nilainya ditetapkan sebelum dimulainya proses pembelajaran. Berbeda dengan parameter nilai hyperparameter tidak berubah seiring berjalannya proses pelatihan. Beberapa jenis hyperparameter:

1. Learning Rate
2. Momentum Coeficient
3. Layer Size
4. Loss function
5. Jumlah epoch

Untuk mencari nilai hyperparameter yang terbaik untuk model. Kita dapat melakukan hyperparameter tuning. Proses ini adalah mengubah – ubah nilai hyperparameter untuk memilih satu set hyperparameter yang optimal untuk algoritma pembelajaran. Hyperparameter adalah parameter yang nilainya digunakan untuk mengontrol proses. Pembelajaran. Terdapat beberapa metode hyperparameter tuning:

- *Random Search*, yaitu memilih nilai secara **acak** didalam domain nilai hyperparameter tersebut.
- *Grid Search*, yaitu memilih setiap nilai pada grid secara beraturan dimana nilai tersebut merupakan nilai didalam domain nilai hyperparameter tersebut.

Pada laporan ini, digunakan data Electrical Grid Stability Simulated Data Dataset pada *https://archive.ics.uci.edu/ml/datasets/Electrical+Grid+Stability+Simulated+Data+* dengan jumlah *instance* sebanyak 10000 data. Ada 11 parameter yang terdiri dari atribut daya listrik.

Kombinasi ini membuat matrix input memiliki ukuran 10000x12. Dari 10000 data, data tersebut dipecah dengan perbandingan 70:30 dimana 7000 data akan digunakan untuk pembelajaran dan 3000 data sisanya akan digunakan untuk proses *testing*.

Berikut merupakan algoritma pemrograman *backpropagation*:

A. Inisialisasi bobot

Inisialisasi awal dari tiap bobot hubungan antar neuron dapat dilakukan menggunakan dua buah metode yaitu random dan nguyen widrow.

1. Random

Metode inisialisasi random adalah metode dimana bobot awal ditentukan secara acak. Hal ini dapat kita lakukan menggunakan fungsi random pada MATLAB. Inisialisasi awal yang acak dibutuhkan untuk menghasilkan symmetry breaking. Suatu strategi yang efektif untuk diterapkan pada inisialisasi random adalah menentukan batas atas dan batas bawah dari nilai acak tersebut. Hal ini dilakukan untuk memastikan bahwa bobot neuron tidak terlalu besar sehingga proses pembelajaran lebih efisien.

Berikut merupakan kode MATLAB untuk inisialisasi random:

```
%Random Init
epsilon_init = 0.5;
weight_xz = rand(x_size, z_size) * 2 * epsilon_init - epsilon_init;
weight_zy = rand(z_size, y_size) * 2 * epsilon_init - epsilon_init;
bias_xz = rand(1,z_size) * 2 * epsilon_init - epsilon_init;
bias_zy = rand(1,y_size) * 2 * epsilon_init - epsilon_init;
```

2. Nguyen Widrow

Metode nguyen widrow adalah sebuah algoritma modifikasi sederhana dari bobot yang mampu meningkatkan kecepatan proses pembelajaran. Algoritma nguyen-widrow adalah sebagai berikut:

- Menentukan besar faktor skala (β)

  $\beta = 0{,}7 * J^{\frac{1}{I}}$

  Dengan: J = Ukuran *hidden layer*; I = Ukuran *input layer*.
- Inisialisasi bobot secara random pada rentang -0.5 sampai 0.5
- Menghitung norm dari vektor bobot

  $\|v_j\| = \sqrt{\sum_{i=1}^{I} v_{ij}^2}$ dan $\|w_k\| = \sqrt{\sum_{j=1}^{J} w_{jk}^2}$

- Memperbaharui bobot

  $v_{ij} = \frac{\beta v_{ij}}{\|v_j\|}$ $\quad$ $w_{jk} = \frac{\beta w_{jk}}{\|w_k\|}$

- Mengatur bias sebagai bilangan acak antara β sampai - β

  Berikut merupakan kode MATLAB untuk inisialisasi nguyen-widrow:

```
%Nguyen Widrow
beta = 0.7 * z_size^(1/x_size);

for i = 1:z_size
    norm(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norm(i));
end
```

```
bias_xz = rand(1, z_size) * 2 * beta - beta;
bias_zy = rand(1, y_size) * 2 * beta - beta;
```

B. Data Preprocessing

Preprocessing pertama adalah normalisasi data yaitu proses mengubah data pada suatu feature agar mempunyai distribusi yang sama. Hal ini dilakukan agar seluruh feature mempunyai rentang yang sama sehingga tidak ada feature yang mendominasi satu dengan yang lainnya. Terdapat 2 jenis normalisasi yang digunakan yaitu Normalisasi Min-Max dan Normalisasi Z-Score.

Berikut merupakan kode MATLAB untuk normalisasi data:

```
%% Normalisasi
%Tanpa Normalisasi
feature_norm_test = feature_test;

%Normalisasi Min-Max
feature_norm_test = zeros(size(feature_test));
for m = 1 : feature_row_test
    for n_test = 1 : feature_col_test
       feature_norm_test(m,n_test) = ((feature_test(m,n_test) -
min(feature_test(:,n_test)))/(max(feature_test(:,n_test)) -
min(feature_test(:,n_test))));
    end
end

%Normalisasi Z-Score
feature_norm_test = zscore(feature_test,[],1);
```

C. Algoritma Pembelajaran

Algoritma untuk proses pembelajaran adalah sebagai berikut:

Selama kondisi stopping FALSE, maka untuk setiap pasangan pelatihan:

Proses *forward pass*

1. Komputasi *Input Layer*. Setiap x_i, i = 1,2,...,I
   - Menerima input x_i
   - Mengirimkannya ke semua *Hidden Layer* diatasnya.
2. Komputasi *Hidden Layer*. Setiap h_j, j = 1,2,...,J
   - $z\_in_j = v_{0j} + \Sigma x_{pi} v_{ij}$
   - $z_j = \frac{1}{1+e^{-z\_in_j}}$
3. Komputasi *Output Layer*. Setiap o_k, k = 1,2,...,K
   - $o\_in_k = w_{0k} + \Sigma z_j w_{jk}$

- $o = \frac{1}{1+e^{-y\_in_k}}$
- Kuantisasi output
  - $0 <= O_k < 0.3 \rightarrow O_k = 0$
  - $0.7 < O_k <= 1 \rightarrow O_k = 1$
  - Else $\rightarrow O_k = O_k$.

Pada layer output diterapkan suatu threshold untuk membantu proses pembelajaran. Hal ini dilakukan karena sebuah output yang bernilai 1 atau 0 pada neuron output nyaris mustahil untuk tercapai tanpa adanya fungsi matematika tambahan. Threshold yang dipilih adalah 0.7 untuk batas atas dan 0.3 untuk batas bawah. Sehingga output yang bernilai lebih besar dari 0.7 akan dikonversi menjadi nilai biner 1 dan output yang bernilai lebih kecil dari 0.3 akan dikonversi menjadi nilai biner 0

Proses *backward pass*

1. Komputasi di *Output Layer*. Setiap o_k, k = 1,2,...,K
   - Menghitung informasi error:
     $\delta_k = (t_{pk} - y_{pk}).f'(y\_in_k)$
   - Menghitung besarnya koreksi bobot *output layer*:
     $\Delta w_{jk} = \alpha \delta_k z_j$
   - Menghitung besarnya koreksi bias output:
     $\Delta w_{0k} = \alpha \delta_k$
4. Komputasi di *Hidden Layer*. Setiap h_j, j = 1,2,...,J
   - Menghitung semua koreksi error:
     $\delta\_in_j = \Sigma \delta_k w_{jk}$
   - Menghitung nilai aktivasi koreksi error :
     $\delta_j = \delta_{in_j} f'(z\_in_j)$
   - Menghitung koreksi bobot unit hidden :
     $\Delta v_{ij} = \alpha \delta_j x_i$
   - Menghitung koreksi error bias unit hidden :
     $\Delta v_{0j} = \alpha \delta_j$

Proses memperbaharui bobot

1. Bobot ke *hidden layer*:
   $v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$

$$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j}$$

2. Bobot ke *output layer*:

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$$

$$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k}$$

Proses perhitungan error

Error dihitung dengan menggunakan fungsi kesalahan fungsi error kuadratis:

$$E = \frac{1}{2}\Sigma\left(t_{pk} - y_{pk}\right)^2$$

Berikut merupakan kode MATLAB untuk threshold data output:

```
%Threshold
for s=1:3
    if y(n,s) >= 0.7
        y(n,s) = 1;
    end
    if y(n,s) <= 0.3
        y(n,s) = 0;
    end
end
```

Program MATLAB proses pembelajaran

```
clear;
close all;
%% Data Input
load('data.mat')
feature = dataset(1:7000,1:12);
feature = feature{:,:};
class = dataset(1:7000,13);
class = class{:,:};

[feature_row,feature_col] = size(feature);
[class_row,class_col] = size(class);
%% Normalisasi
%Tanpa Normalisasi
feature_norm = feature;

%Normalisasi Min-Max
feature_norm = zeros(size(feature));
for m = 1 : feature_row
    for n_train = 1 : feature_col
       feature_norm(m,n_train) = ((feature(m,n_train) -
min(feature(:,n_train)))/(max(feature(:,n_train)) -
min(feature(:,n_train))));
    end
end

%Normalisasi Z-Score
feature_norm = zscore(feature,[],1);
%% Hyperparameter & Declare Variable
```

```
%Hyperparameter
x_size  = feature_col; %Input layer
z_size = 7;  %Hidden layer
y_size = class_col; %Output layer

alpha = 0.1;
epoch = 2000;
miu = 0;

%Variable
stop = 0;
error_target = 0.00001;
data_count = feature_row;
epoch_count = 1;

delta_zy_old = 0;
delta_xz_old = 0;
delta_zy_bias_old = 0;
delta_xz_bias_old = 0;
%% Initialisasi

%Random Init
rng(2) %Seed
epsilon_init = 0.5; %Range random number

%Randomize weight and bias
weight_xz = rand(x_size, z_size) * 2 * epsilon_init - epsilon_init;
weight_zy = rand(z_size, y_size) * 2 * epsilon_init - epsilon_init;
bias_xz = rand(1,z_size) * 2 * epsilon_init - epsilon_init;
bias_zy = rand(1,y_size) * 2 * epsilon_init - epsilon_init;

%Nguyen Widrow
beta = 0.7 * z_size^(1/x_size);

for i = 1:z_size
    norm(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norm(i));
end

bias_xz = rand(1, z_size) * 2 * beta - beta;
bias_zy = rand(1, y_size) * 2 * beta - beta;
%% Backpropagation
while stop == 0 && epoch_count <= epoch
    true_count_training = 0;
    for n_train = 1:data_count
        %Forward Pass
        %Input -> Hidden
        x_train = feature_norm(n_train,:);
        t_train = class(n_train,:);

        z_in_train = bias_xz + x_train*weight_xz;

        for m=1:z_size
            z_train(1,m) = 1/(1+exp(-z_in_train(1,m)));
        end

        %Hidden -> Output
```

```
        y_in_train = bias_zy + z_train*weight_zy;

        for l=1:y_size
            y_train(n_train,l) = 1/(1+exp(-y_in_train(1,l)));
        end

        %Threshold
        for s=1:1
            if y_train(n_train,s) >= 0.7
                y_train(n_train,s) = 1;
            end
            if y_train(n_train,s) <= 0.3
                y_train(n_train,s) = 0;
            end
        end

        %Backward Pass
        %Output->Hidden
        for l=1:y_size
            do_k(1,l) = (y_train(n_train,l) - t_train(1,l)) *
(y_train(n_train,l)*(1-y_train(n_train,l)));
        end

        delta_zy = (alpha .* z_train' * do_k);
        delta_zy_bias = alpha .* do_k;

        %Hidden->Input
        sigma_j = do_k * weight_zy';
        for m=1:z_size
            do_j(1,m) = (sigma_j(1,m)) .* (z_train(1,m)*(1-
z_train(1,m)));
        end

        delta_xz = (alpha .* x_train' * do_j);
        delta_xz_bias = alpha .* do_j;

        %Momentum calculation
        momentum_zy = miu*delta_zy_old;
        momentum_xz = miu*delta_xz_old;
        momentum_bias_zy = miu*delta_zy_bias_old;
        momentum_bias_xz = miu*delta_xz_bias_old;

        %Weight Update
        weight_zy = weight_zy - delta_zy - momentum_zy;
        weight_xz = weight_xz - delta_xz - momentum_xz;
        bias_zy = bias_zy - delta_zy_bias - momentum_bias_zy;
        bias_xz = bias_xz - delta_xz_bias - momentum_bias_xz;

        %Cost function
        error(1,n_train) = 0.5.*sum((t_train-
y_train(n_train,:)).^2);

        %Momentum update
        delta_zy_old = delta_zy;
        delta_xz_old = delta_xz;
        delta_zy_bias_old = delta_zy_bias;
```

```
        delta_xz_bias_old = delta_xz_bias;

        %Recognition Rate
        if y_train(n_train,:) == t_train
           true_count_training = true_count_training + 1;
        end
    end
    error_per_epoch(1,epoch_count) = sum(error)/feature_row;
    acc_training = true_count_training/data_count*100;
    acc_training_per_epoch(1,epoch_count) = acc_training;

    fprintf('\n\nEPOCH              :  %d \n',epoch_count);
    fprintf('TRAIN SCORE \n');
    fprintf('MSE Train           :  %.4f \n',
sum(error)/feature_row);
    fprintf('Accuracy Train   :  %.2f \n', acc_training);

    if error_per_epoch(1,epoch_count) < error_target
        stop = 1;
    end

    Backprop_testing_1Layer

    epoch_count = epoch_count+1;
end

figure;
plot(error_per_epoch)
hold on;
plot(error_test_per_epoch)

figure;
plot(acc_training_per_epoch)
```

D. Algoritma Pengujian

Setiap p = 1 sampai p = Pu :

Proses *feedforward*

3.1. Komputasi *Input Layer*. Setiap x_uji_{pi}, i = 1,2,...,I

- Menerima input x_uji_{pi}
- Mengirimkannya ke semua *Hidden Layer*.

3.2. Komputasi *Hidden Layer*. Setiap Z_j, j = 1,2,...,J

- $z\_in\_uji_j = v_{0j} + \Sigma x\_uji_{pi} v_{ij}$
- $z\_uji_j = \frac{1}{1+e^{-z\_in\_uji_j}}$

3.3. Komputasi *Output Layer*. Setiap y_k, k = 1,2,...,K

- $y\_in\_uji_k = w_{0k} + \Sigma z\_uji_j w_{jk}$
- $y\_uji_{pk} = \frac{1}{1+e^{-y\_in\_uji_k}}$
- Kuantisasi output
  - 0 <= y_uji_{pk} < 0.3 → y_rec_{pk} = 0
  - 0.7 < y_uji_{pk} <= 1 → y_rec_{pk} = 1
  - Else → y_rec_{pk} = y_uji_{pk}.

3.4. Menghitung *recognition rate*:

$$RR = \frac{banyaknya\ data\ output\ yang\ benar}{banyaknya\ data\ pengujian} \times 100\%$$

Output dikatakan benar jika nilai biner nya sama dengan binernya target.

Program MATLAB Proses Pengujian

```
%% Data Input
load('data.mat')
feature_test = dataset(7001:10000,1:12);
feature_test = feature_test{:,:};
class_test = dataset(7001:10000,13);
class_test = class_test{:,:};

[feature_row_test,feature_col_test] = size(feature_test);
[class_row_test,class_col_test] = size(class_test);

%% Normalisasi
% %Tanpa Normalisasi
% feature_norm_test = feature_test;

%Normalisasi Min-Max
feature_norm_test = zeros(size(feature_test));
for m = 1 : feature_row_test
    for n_test = 1 : feature_col_test
       feature_norm_test(m,n_test) = ((feature_test(m,n_test) -
min(feature_test(:,n_test)))/(max(feature_test(:,n_test)) -
min(feature_test(:,n_test))));
    end
end

% %Normalisasi Z-Score
% feature_norm_test = zscore(feature_test,[],1);
%% Hyperparameter & Declare Variable

%Variable
data_count_test = feature_row_test;
true_count_test = 0;

for n_test = 1:data_count_test
    %Forward Pass
    %Input -> Hidden
    x_test = feature_norm_test(n_test,:);
```

```
    t_test = class_test(n_test);

    z_in_test = bias_xz + x_test*weight_xz;

    for m=1:z_size
        z_test(1,m) = 1/(1+exp(-z_in_test(1,m)));
    end

    %Hidden -> Output
    y_in_test = bias_zy + z_test*weight_zy;

    for l=1:y_size
        y_test(n_test,l) = 1/(1+exp(-y_in_test(1,l)));
    end

    %Threshold
    for s=1:1
        if y_test(n_test,s) >= 0.5
            y_test(n_test,s) = 1;
        end
        if y_test(n_test,s) < 0.5
            y_test(n_test,s) = 0;
        end
    end

    %Cost function
    error_test(1,n_test) = 0.5*((t_test-y_test(n_test,:)).^2);

    %Recognition Rate
    if y_test(n_test,:) == t_test
       true_count_test = true_count_test + 1;
    end

end

error_test_per_epoch(1,epoch_count) =
sum(error_test)/feature_row_test;

fprintf('TEST SCORE \n');
mse_testing = sum(error_test)/feature_row_test;
fprintf('MSE Testing        :  %.4f \n', mse_testing);
acc_testing = true_count_test/data_count*100;
fprintf('Accuracy Testing   :  %.2f \n', acc_testing);
```

BAB III
ANALISIS HASIL PERCOBAAN

3.1 Hasil Percobaan

Parameter dari jaringan akan divariasikan yaitu metode inisialisasi , nilai learning rate (α), jumlah neuron pada hidden layer (J), dan menggunakan momentum atau tidak. Variasi dari parameter ini diharapkan dapat memberikan hasil yang dapat menggambarkan pengaruh parameter terhadap kondisi dan hasil dari ANN.

3.2.1 Hasil Variasi Metode Inisialisasi

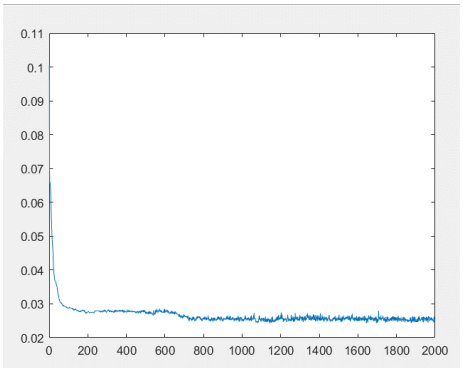
Pada percobaan ini akan mencoba membandingkan dari hasil menggunakan metode inisialisasi random dan nguyen-widrow. Variabel tetap yang digunakan adalah

- Learning Rate = 0.1
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan

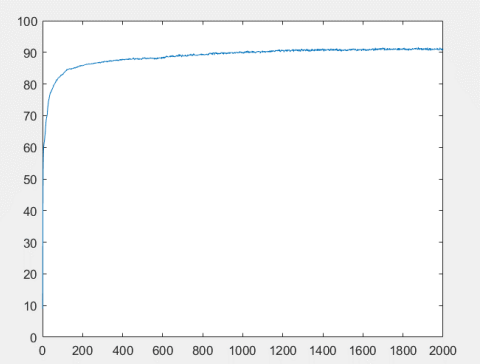
Hasil yang diperoleh adalah sebagai berikut:

a) Random Initialization

Error Per Epoch



Accuracy



Final Loss Training = *0.0364*

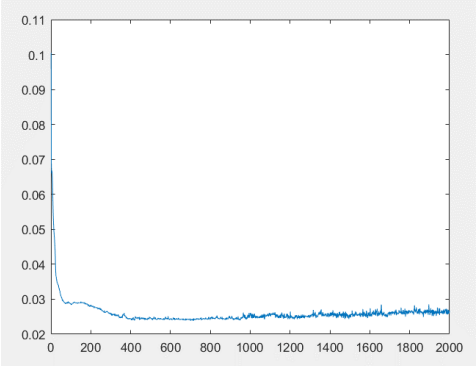
Final Accuracy Training = *87.14*

Final Loss Testing = *0.1009*

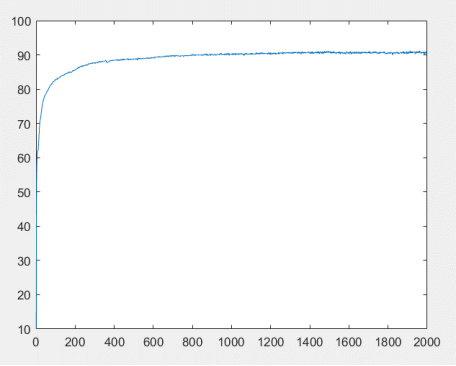
Final Accuracy Testing = *89.10*

b) Nguyen-widrow Initialization

Error Per Epoch



Accuracy



Final Loss Training = *0.0266*

Final Accuracy Training = *90.60*

Final Loss Testing = *0.0754*

Final Accuracy Testing = *92.03*

3.2.2 Hasil Variasi Metode Normalisasi

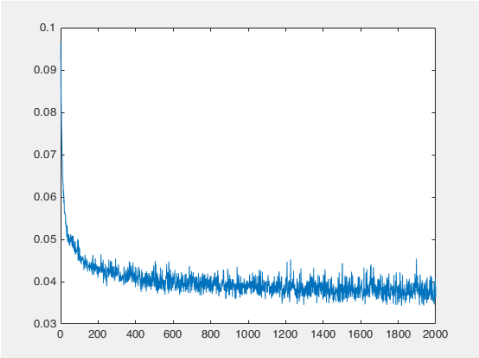
Pada percobaan ini akan mencoba membandingkan dari hasil tanpa metode normalisasi serta dengan metode normalisasi Min-Max dan Z-Score. Variabel tetap yang digunakan adalah

- Learning Rate = 0.1
- Ukuran hidden layer = 7 neuron
- Target error maksimal = 0.0001
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan
- Metode Inisialisasi = Nguyen-Widrow

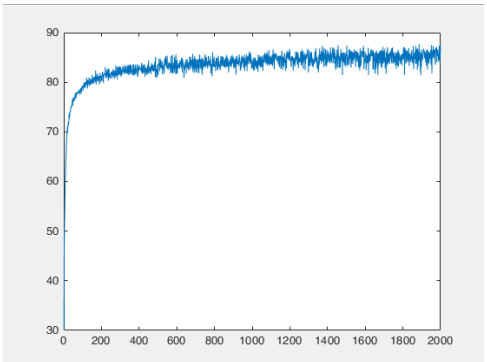
Hasil yang diperoleh adalah sebagai berikut:

a) Tanpa Normalisasi

Error Per Epoch



Accuracy



Final Loss Training = *0.04*

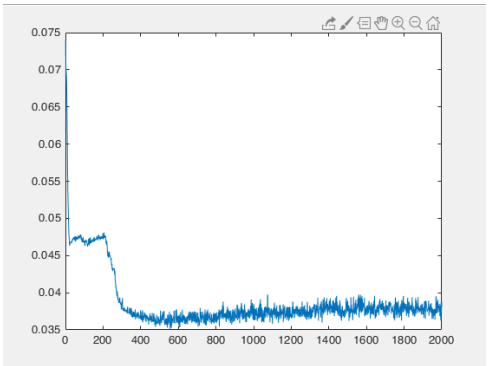
Final Accuracy Training = *87.14*

Final Loss Testing = *0.10*

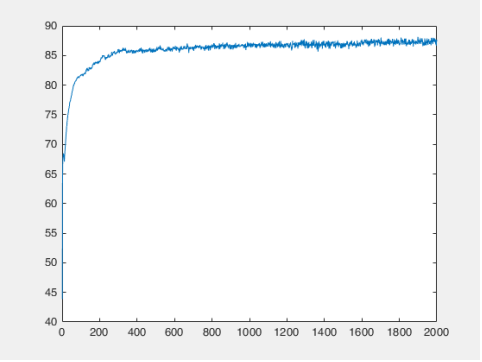
Final Accuracy Testing = *89.10*

b) Metode Normalisasi Z-Score

Error Per Epoch



Accuracy



Final Loss Training = *0.04*

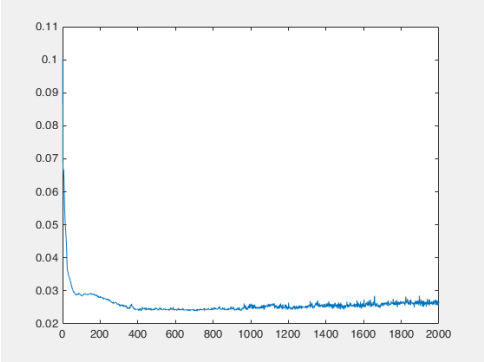
Final Accuracy Training = *87.64*

Final Loss Testing = *0.10*

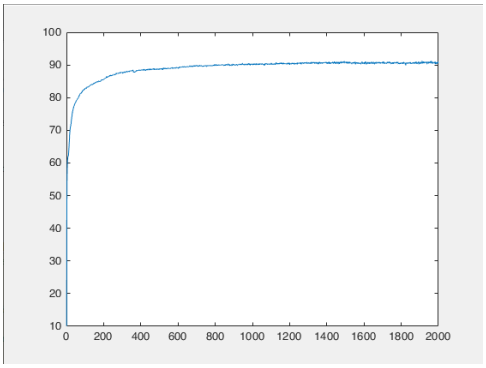
Final Accuracy Testing = *89.13*

c) Metode Normalisasi Min-Max

Error Per Epoch



Accuracy



Final Loss Training = *0.03*

Final Accuracy Training = *90.60*

Final Loss Testing = *0.08*

Final Accuracy Testing = *92.03*

3.2.3 Hasil Variasi Jumlah Epoch

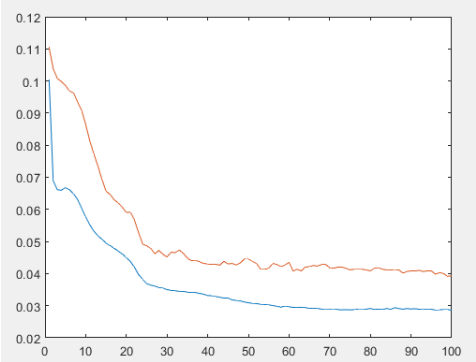
Pada percobaan ini akan mencoba membandingkan pengaruh dari perbedaan jumlah epoch. Variabel tetap yang digunakan adalah

- Learning Rate = 0.1
- Target error maksimal = 0.0001
- Ukuran hidden layer = 7 neuron
- Momentum = Tidak Digunakan
- Metode Inisialisasi = Nguyen-Widrow
- Metode Normalisasi = Min-Max

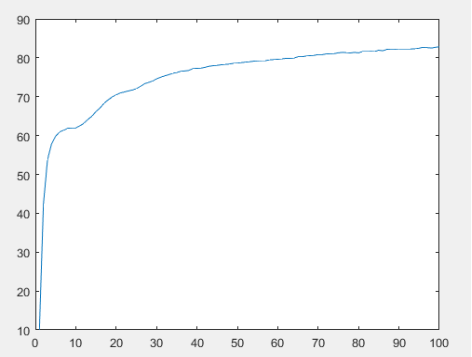
Hasil yang diperoleh adalah sebagai berikut:

a) **100 Epoch**

Error Per Epoch



Accuracy



Final Loss Training = *0.0286*

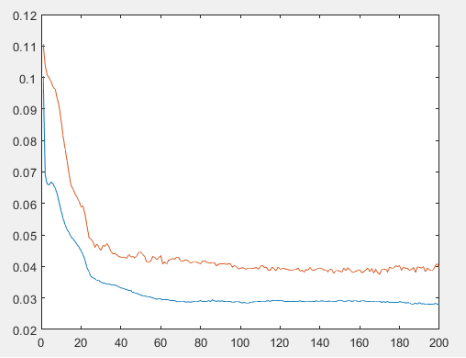
Final Accuracy Training = *82.81*

Final Loss Testing = *0.0393*

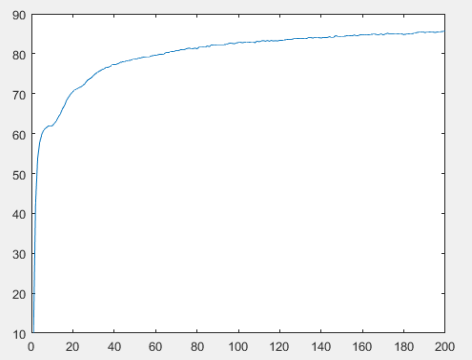
Final Accuracy Testing = *92.13*

b) **200 Epoch**

Error Per Epoch



Accuracy



Final Loss Training = *0.028*

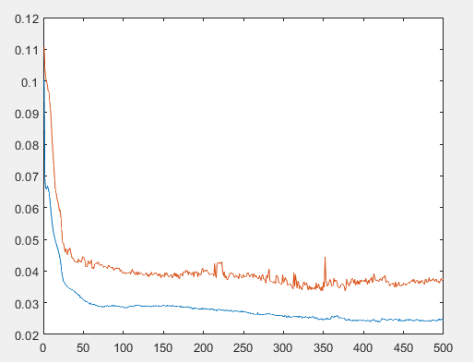
Final Accuracy Training = *85.61*

Final Loss Testing = *0.0397*

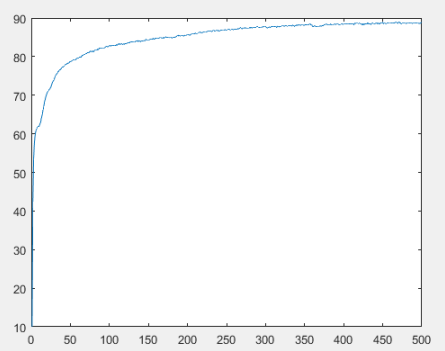
Final Accuracy Testing = *92.07*

c) 500 Epoch

Error Per Epoch



Accuracy



Final Loss Training = *0.0247*

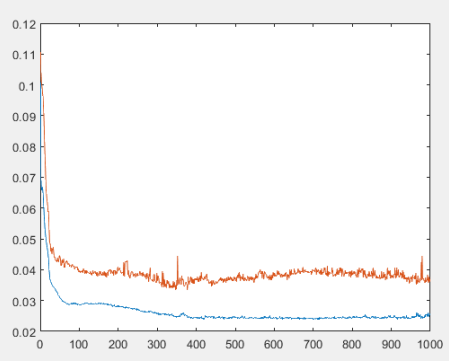
Final Accuracy Training = *88.86*

Final Loss Testing = *0.0375*

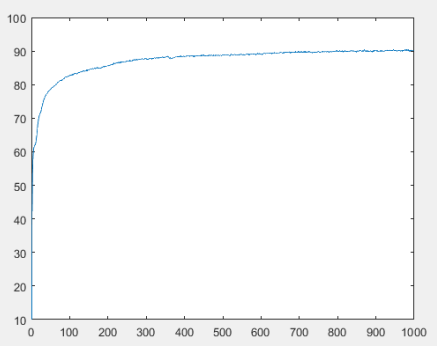
Final Accuracy Testing = *92.50*

d) 1000 Epoch

Error Per Epoch



Accuracy



Final Loss Training = *0.0257*

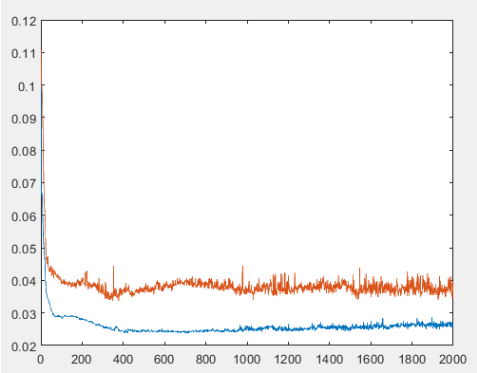
Final Accuracy Training = *90.17*

Final Loss Testing = *0.0368*

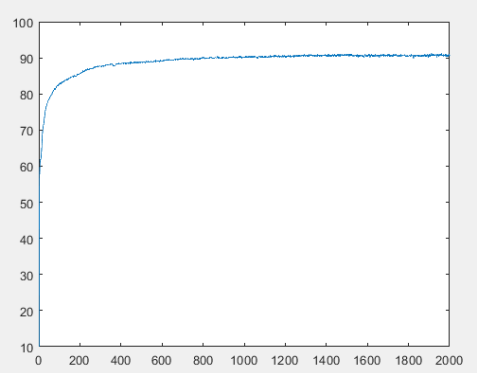
Final Accuracy Testing = *92.63*

e) 2000 Epoch

Error Per Epoch



Accuracy



Final Loss Training = *0.0266*

Final Accuracy Training = *90.60*

Final Loss Testing = *0.0398*

Final Accuracy Testing = *92.03*

3.2.4 Hasil Variasi Jumlah Neuron Hidden Layer

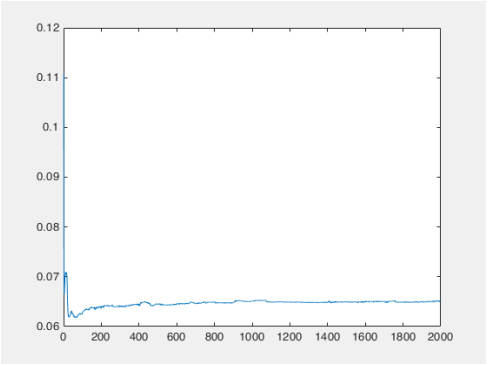
Pada percobaan ini akan mencoba membandingkan pengaruh dari perbedaan jumlah neuron hidden layer. Variabel tetap yang digunakan adalah

- Learning Rate = 0.1
- Target error maksimal = 0.0001
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan
- Metode Inisialisasi = Nguyen-Widrow
- Metode Normalisasi = Min-Max

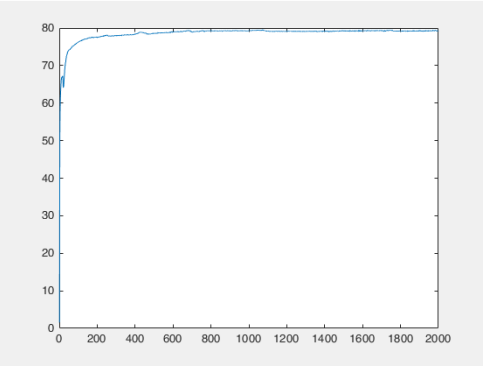
Hasil yang diperoleh adalah sebagai berikut:

a) 2 Neuron Hidden Layer

Error Per Epoch



Accuracy



Final Loss Training = 0.07

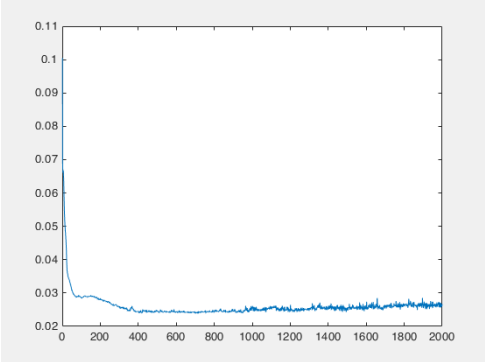
Final Accuracy Training = 79.36

Final Loss Testing = 0.17

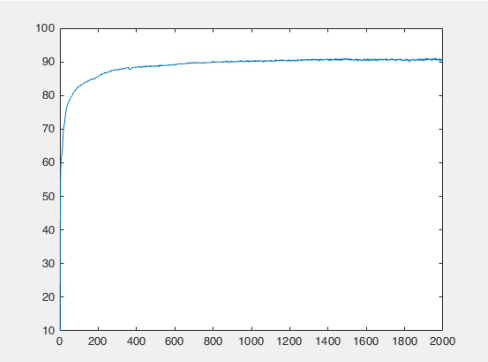
Final Accuracy Testing = 82.80

b) 7 Neuron Hidden Layer

Error Per Epoch



Accuracy



Final Loss Training = 0.03

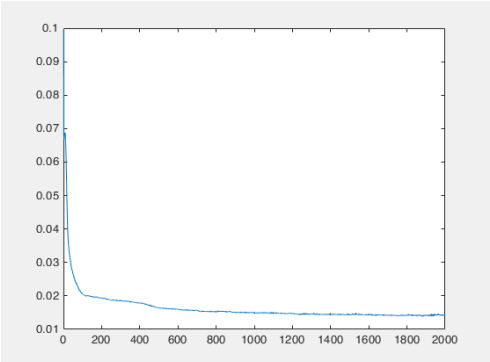
Final Accuracy Training = 90.60

Final Loss Testing = 0.08

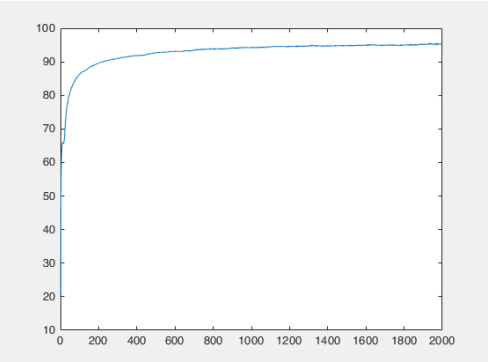
Final Accuracy Testing = 92.03

c) 17 Neuron Hidden Layer

Error Per Epoch



Accuracy



Final Loss Training = 0.01

Final Accuracy Training = 95.37

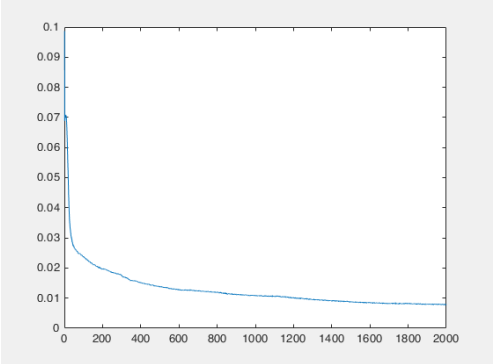
Final Loss Testing = 0.05

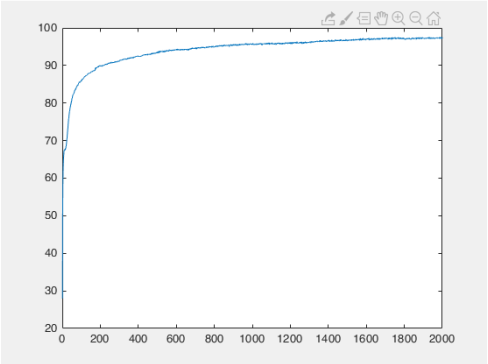
Final Accuracy Testing = 92.90

d) 31 Neuron Hidden Layer

Error Per Epoch

Accuracy





Final Loss Training = 0.01

Final Accuracy Training = 97.19

Final Loss Testing = 0.04

Final Accuracy Testing = 94.33

3.2.5 Hasil Variasi Nilai Learning Rate (α)

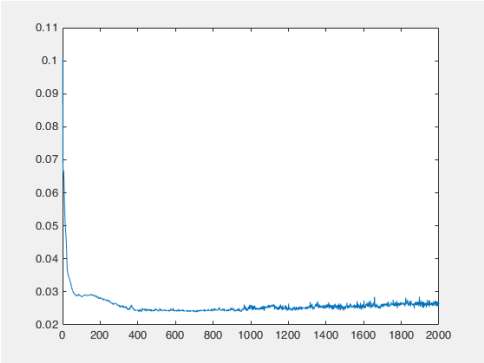
Pada percobaan ini akan mencoba membandingkan nilai learning rate yang berbeda. Nilai learning rate divariasikan dengan nilai 0.1,0.2,0.15,0.05. Variabel tetap yang digunakan adalah:

- Metode Inisialisasi = Nguyen-widrow
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.1
- Maximum epoch = 2000 epoch
- Momentum = Tidak Digunakan

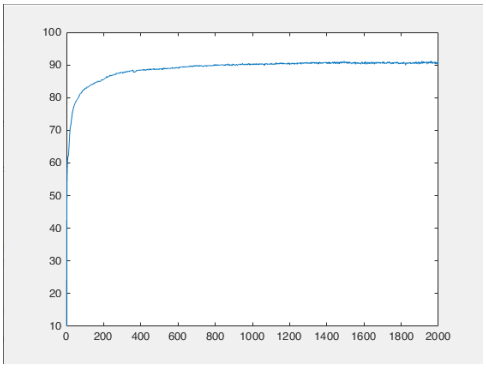
Hasil yang diperoleh adalah sebagai berikut:

a) α = 0.1

Error Per Epoch



Accuracy



Final Loss Training = 0.03

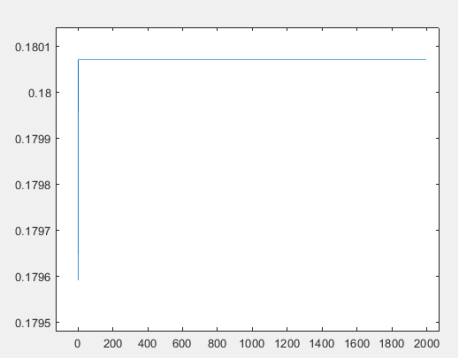
Final Accuracy Training = 90.6

Final Loss Testing = 0.08

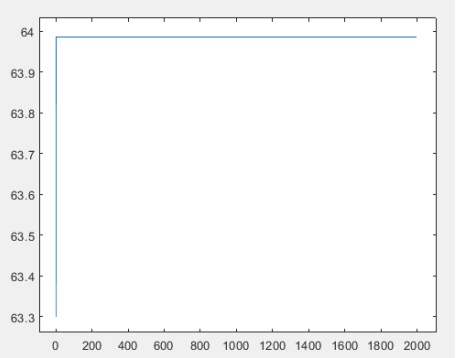
Final Accuracy Testing = 92.03

b) α = 0.2

Error Per Epoch



Accuracy



Final Loss Training = 0.18

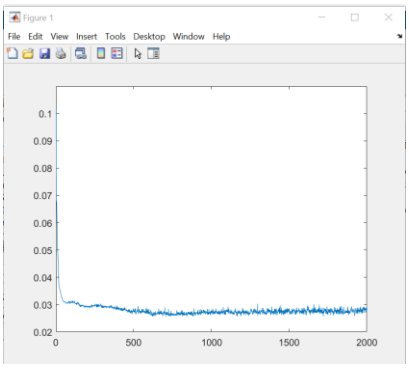
Final Accuracy Training = 63.99

Final Loss Testing = 0.42

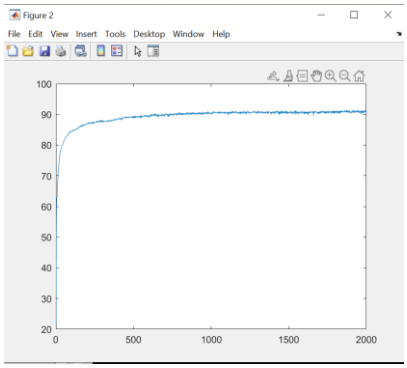
Final Accuracy Testing = 63.37

c) α = 0.15

Error Per Epoch



Accuracy



Final Loss Training = 0.03

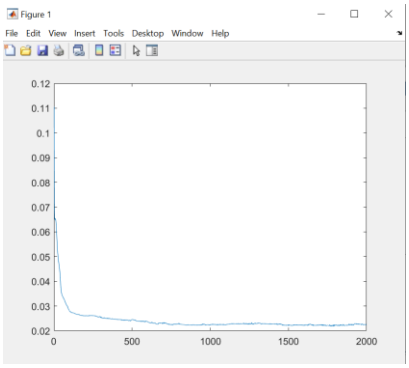
Final Accuracy Training = 90.56

Final Loss Testing = 0.08

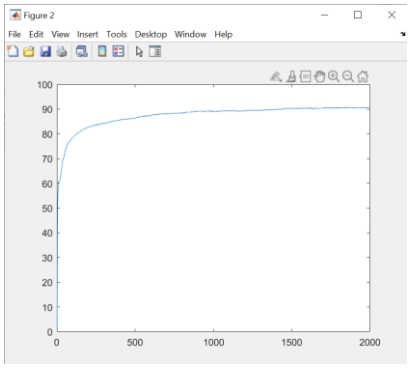
Final Accuracy Testing = 91.43

d) α = 0.05

Error Per Epoch



Accuracy



Final Loss Training = 0.02

Final Accuracy Training = 90.67

Final Loss Testing = 0.07

Final Accuracy Testing = 92.93

3.2.6 Hasil Variasi Koefisien Momentum (μ)

Pada percobaan ini akan mencoba menganalisa penggunaan momentum serta mencoba beberapa nilai koefisien momentum (μ) yang berbeda. Koefisien momentum (μ) divariasikan senilai 0, 0.2, 0.4, 0.6 . Variabel tetap yang digunakan adalah :

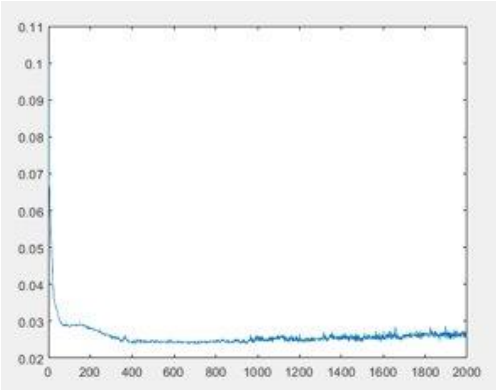
- Metode Inisialisasi = Nguyen-widrow

- Learning rate = 0.1
- Ukuran hidden layer = 7 node
- Target error maksimal = 0.0001
- Maximum epoch = 2000 epoch

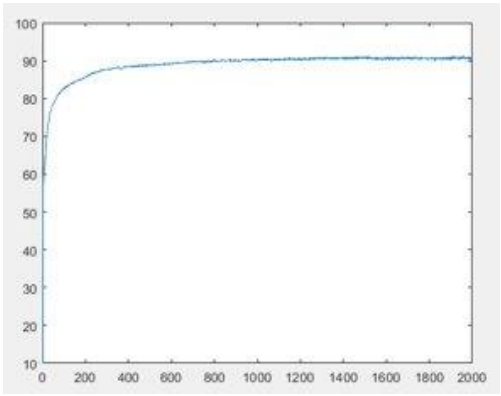
Hasil yang diperoleh adalah sebagai berikut:

a) **μ** = 0

Error Per Epoch



Accuracy



Final Loss Training = 0. 03

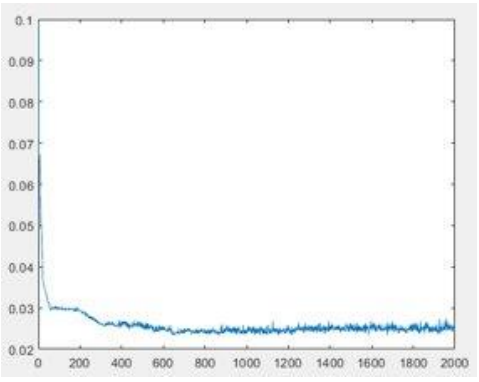
Final Accuracy Training = 90.6

Final Loss Testing = 0.08

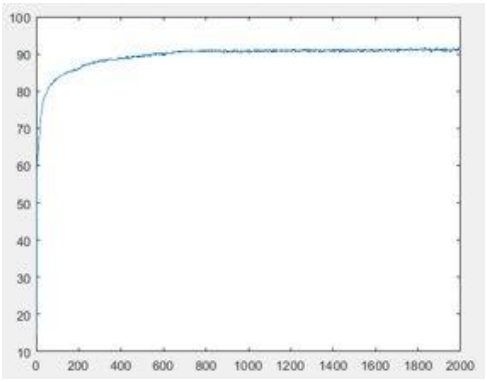
Final Accuracy Testing = 92.03

b) μ **= 0.2**

Error Per Epoch



Accuracy



Final Loss Training = 0. 03

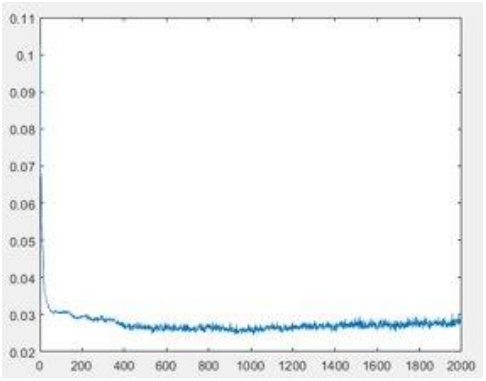
Final Accuracy Training = 90.71

Final Loss Testing = 0.08

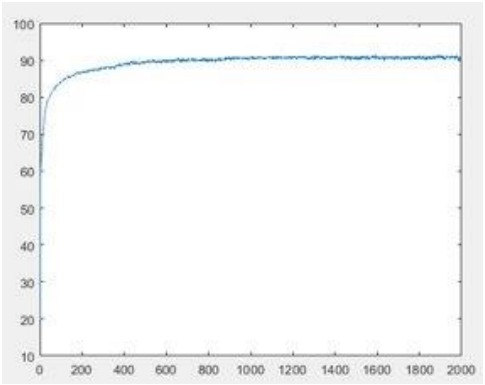
Final Accuracy Testing = 91.57

c) μ = **0.4**

Error Per Epoch



Accuracy



Final Loss Training = 0. 03

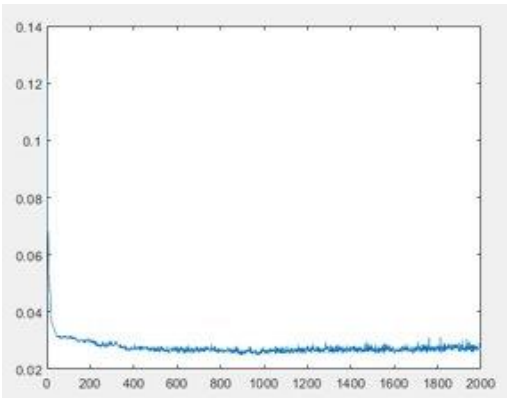
Final Accuracy Training = 90.79

Final Loss Testing = 0.08

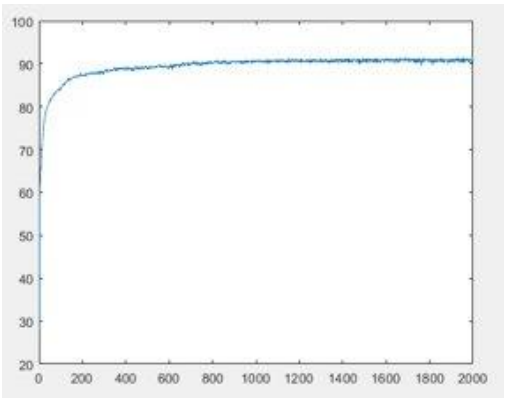
Final Accuracy Testing = 92.13

d) μ = **0.6**

Error Per Epoch



Accuracy



Final Loss Training = 0. 03

Final Accuracy Training = 91.03

Final Loss Testing = 0.08

Final Accuracy Testing = 91.13

3.2 Analisis Hasil Data

3.2.1 Analisis Variasi Metode Inisialisasi

Pada hasil percobaan dengan variasi metode inisialisasi, dengan mengubah metode yang digunakan, maka dapat dianalisa pengaruh metode tersebut dalam proses dan hasil *backpropagation*. Hasil yang diperoleh bahwa metode inisialiasi nguyen-widrow dapat mempercepat ANN mencapai konvergensi. Namun secara hasil akhir tidak memberikan perbedaan hasil akhir yang signifikan dibandingkan metode inisialisasi acak. Sehingga dapat dikatakan bahwa metode nguyen-widrow cocok untuk diterapkan pada dataset atau sistem yang kompleks untuk mengurangi jumlah iterasi yang digunakan

3.2.2 Analisis Variasi Metode Normalisasi

Pada hasil percobaan dengan variasi metode normalisasi, dengan memvariasikan metode normalisasi menjadi tiga buah, diperoleh hasil bahwa metode normalisasi sangat berpengaruh dengan fluktuasi dari nilai error per epoch. Metode normalisasi min-max memberikan hasil fluktuasi yang paling kecil dan metode tanpa normalisasi memberikan hasil yang paling fluktuatif. Hal ini disebabkan oleh adanya parameter yang mendominasi parameter yang lainnya. Parameter yang memiliki nilai atau rentang yang lebih besar akan mendominasi parameter yang memiliki nilai atau rentang yang lebih kecil.

3.2.3 Analisis Variasi Jumlah Epoch

Pada hasil percobaan dengan variasi jumlah epoch, dengan memvariasikan jumlah epoch yang digunakan menjadi 100,200,500,1000 dan 2000, diperoleh hasil bahwa jumlah epoch yang lebih banyak akan menghasilkan jaringan yang lebih baik. Hal ini dilakukan karena nilai bobot – bobot pada jaringan sudah mencapai titik optimum. Pengunaan epoch yang terlalu sedikit dapat membuat bobot jaringan belum mencapai titik optimum sehingga memberikan prediksi yang kurang baik. Dapat dilihat juga bahwa pergerakan error per epoch baik dari training dan testing memberikan pola garis yang mirip sehingga dapat dikatakan bahwa jaringan kita tidak mengalami overfitting.

3.2.4 Analisis Variasi Jumlah Neuron Hidden Layer (J)

Pada hasil percobaan dengan variasi jumlah neuron hidden layer, dengan mengubah jumlah neuron, maka dapat dianalisa pengaruh dari jumlah neuron terhadap proses dan hasil *backpropagation*. Hasil yang diperoleh adalah idealnya semakin banyak neuron yang digunakan maka akan memberikan hasil yang lebih baik. Hal ini disebabkan karena semakin banyak neuron yang digunakan pada hidden layer maka akan semakin banyak bobot dan hubungan yang dapat di tune untuk memberikan hasil yang lebih baik. Namun terdapat

beberapa nilai neuron hidden layer yang tidak cocok dengan jaringan, nilai neuron tersebut perlu diwaspadai.

Hal lain yang perlu diperhatikan fenomena underfitting dan overfitting, dimana pengunaan jumlah neuron yang terlalu sedikit dapat menyebabkan underfitting sedangkan penggunaan jumlah neuron yang terlalu banyak dapat menyebabkan overfitting.

Terlihat pada jumlah neuron 5 dan 7 bahwa jaringan mengalami underfitting, hal ini ditunjukkan oleh nilai akurasi testing yang lebih tinggi dibandingkan nilai akurasi training[4]. Dengan menambah jumlah neuron menjadi 17 dan 31, terlihat bahwa nilai akurasi training sudah lebih baik daripada akurasi testing sehingga jaringan sudah tidak mengalami underfitting lagi.

3.2.5 Analisis Variasi Nilai Learning Rate (α)

Pada hasil percobaan dengan variasi nilai learning rate, dengan mengubah nilai alpha, maka dapat dianalisa pengaruh alpha terdapat proses dan hasil *backpropagation*. Hasil yang diperoleh adalah semakin besar alpha, maka akan mempercepat jaringan mencapai konvergensi. Namun, hal ini menimbulkan kekurangan yaitu semakin besar alpha maka akan semakin besar potensi sistem tidak mencapai konvergensi. Sebaliknya adalah semakin kecil alpha, maka semakin lama waktu yang dibutuhkan untuk mencapai konvergensi namun semakin besar potensi sistem untuk mencapai konvergensi. Sehingga perlu dicari nilai learning rate yang paling optimal sehingga iterasi yang dilakukan menjadi efektif tanpa menyebabkan jaringan gagal untuk mencapai konvergensi.

3.2.6 Analisis Variasi Koefisien Momentum (μ)

Pada hasil percobaan dengan variasi koefisien momentum, dengan mengubah jumlah koefisien tersebut, kita dapat melihat pengaruh dari momentum terhadap proses dan hasil *backpropagation*. Momentum akan membuat sistem kita menjadi dinamis dengan mempertimbangkan data sebelumnya[5]. Hasil yang diperoleh adalah penggunaan momentum dapat mempercepat ANN untuk mencapai konvergensi karena mengurangi fluktuasi yang tidak diperlukan pada pengupdatean bobot. Namun penggunakaan koefisien momentum yang terlalu besar dapat memperlambat proses pelatihan karena semakin besar penalti yang diberikan terhadap nilai update bobot.

BAB IV
KESIMPULAN

Setelah melakukan percobaan dengan berbagai variasi parameter dan metode, diperoleh kesimpulan yaitu:

1. *Backpropagation* merupakan metode ANN yang merupakan metode pembelajaran sistem yang mengubah nilai bobot dari neuron relatif terhadap error yang diperoleh
2. Metode Normalisasi Min-Max memberikan hasil yang lebih baik dibandingkan tidak menggunakan normalisasi dan menggunakan normalisasi Z-Score untuk dataset Electrical Grid Stability Simulated Data Data Set
3. Metode Inisialisasi nguyen-widrow mempercepat ANN untuk mencapai konvergensi
4. Epoch yang terlalu sedikit membuat jaringan menghasilkan hasil yang kurang baik karena jaringan belum mendapatkan nilai bobot yang optimum.
5. Nilai learning rate (α) berpengaruh terhadap durasi ANN untuk mencapai konvergensi dan potensi untuk mencapai konvergensi tersebut.
6. Jumlah neuron yang terlalu sedikit dapat mengakibatkan underfitting sedangkan jumlah neuron yang terlalu banyak dapat mengakibatkan terjadinya overfitting
7. Penggunaan momentum akan mengurangi noise pergerakan ANN dan mempercepat ANN untuk mencapai konvergensi
8. Pada percobaan ini model terbaik didapatkan ketika melakukan percobaan variasi jumlah neuron hidden layer dengan hasil sebagai berikut:

   Final Loss Training = 0.01
   Final Accuracy Training = 97.19
   Final Loss Testing =0.04
   Final Accuracy Testing = 94.33

DAFTAR REFERENSI

[1] D. Wahyuningsih, I. Zuhroh, and - Zainuri, "Prediksi Inflasi Indonesia Dengan Model Artificial Neural Network," *J. Indones. Appl. Econ.*, vol. 2, no. 2, pp. 2–2008, 2008, doi: 10.21776/ub.jiae.2008.002.02.7.

[2] D. B. Kusumoputro, "BACKPROPAGATION," 2021.

[3] "Artificial_Neural_Network @ Www.Saedsayad.Com." [Online]. Available: http://www.saedsayad.com/artificial_neural_network.htm.

[4] "45854380 @ stackoverflow.com." [Online]. Available: https://stackoverflow.com/a/45854380.

[5] "stochastic-gradient-descent-with-momentum-a84097641a5d @ towardsdatascience.com." [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d#:~:text=Momentum %5B1%5D or SGD with,models are trained using it.

Pembagian Tugas

Nama	Deskripsi Tugas
Ahmad Akbar Habibilah	• Mencari dataset • Membuat kodingan metode normalisasi • Mensimulasikan, mengambil hasil serta analisis variasi learning rate • Membuat slide presentasi cara kerja backpropagation
George	• Membuat kodingan metode inisialisasi • Mensimulasikan mengambil hasil serta analisis variasi momentum dan jumlah epoch • Membuat slide presentasi variasi hyperparameter
Hansel Matthew	• Membuat kodingan forward pass dan backward pass • Integrasi keseluruhan bagian kodingan • Mensimulasikan, mengambil hasil serta analisis variasi metode inisialisasi • Membuat slide presentasi dasar teori
Kemas Muhammad Rizki Fadhila	• Bertanggung jawab membuat kodingan testing • Mensimulasikan, mengambil hasil serta analisis variasi metode normalisasi dan jumlah hidden neuron • Membuat slide presentasi cara kerja backpropagation