ALGORITMA BACKPROPAGATION NEURAL NETWORK PADA SOFTWARE MATLAB UNTUK KLASIFIKASI DATASET IRIS

TUGAS - 01

SISTEM BERBASIS PENGETAHUAN

DOSEN PEMBIMBING

Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng.

Disusun Oleh :

Raihan Nagib (2006574654)

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

2023

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena atas segala rahmat dan hidayah-Nya, penulis berhasil menyusun laporan tugas pertama pada mata kuliah Pemodelan dan Pembelajaran Mesin yang berjudul "Algoritma *Back-Propagation Neural Network* (BPNN) Pada Software Matlab Untuk Klasifikasi Dataset Iris".

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku dosen pengampu mata kuliah Sistem Berbasis Pengetahuan yang telah memberikan pengajaran, bimbingan, dan saran kepada mahasiswa sehingga penulis berhasil menyelesaikan tugas pertama pada mata kuliah Sistem Berbasis Pengetahuan.

Walaupun penulis sudah membuat algoritma dan menyusun laporan dengan maksimal, penulis menyadari bahwa hasil simulasi dan laporan yang telah dibuat masih banyak kekurangan sehingga penulis sangat mengharapkan adanya kritik dan saran yang dapat membangun dari kepada Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku dosen pengampu mata kuliah Sistem Berbasis Pengetahuan supaya laporan ini dapat menjadi lebih baik lagi. Penulis juga berharap laporan ini dapat memberikan manfaat bagi pembaca dan pengembangan dunia pendidikan pada masa kini.

ABSTRAK

Back-propagation Neural Network (BPNN) merupakan suatu algoritma *neural network* untuk melakukan *tuning* pada bobot neural network berdasarkan *error rate* yang telah didapatkan dari *epoch* sebelumnya atau dapat dikatakan didapatkan dari iterasi. Dengan melakukan *tuning* pada bobot neural network, *error rate* dapat berkurang sehingga model yang dibuat akan lebih baik. Pada laporan penelitian yang penulis lakukan, penulis memanfaatkan algoritma BPNN untuk mengklasifikasi dataset *iris*. Dataset ini berisi 150 sampel dari tiga spesies berbeda dari bunga iris (iris sentosa, iris virginica, dan iris versicolor) dengan empat buah fitur pengukuran, yakni panjang sepal, lebar sepal, panjang petal, dan lebar petal.

Kata kunci: *Back-propogation in neural network, iris dataset, eksperimen BPNN*

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini, banyak peneliti yang ingin mengembangkan ilmu pengetahuan dan teknologi dengan cara mendalami ilmu pengetahuan dan teknologi yang sudah ada. Pendalaman ilmu pengetahuan dan teknologi khususnya keteknikan yang dilakukan oleh peneliti telah menghasilkan suatu teknologi yang dapat berfokus pada penggunaan data dan algoritma yang mirip dengan cara berpikir manusia yang berarti teknologi tersebut mempelajari data dengan sendirinya. Teknologi tersebut adalah *machine learning*. Ada banyak jenis dari *machine learning,* salah satunya adalah *artificial neural network* (ANN). *Artificial neural network* (ANN) memiliki prinsip seperti kerja otak pada manusia yang mana itu dapat digunakan untuk analisis yang kompleks di bidang-bidang seperti kesehatan ataupun keteknikan dan bisa juga diterapkan untuk *image processing*, *translation machine*, dan lain sebagainya. Supaya ada gambaran mengenai *artificial neural network,* penulis membuat suatu program dengan menggunakan software MATLAB untuk mengimplementasikan algoritma *back-propagation* dari *artificial neural network* menggunakan dataset yang telah diberikan.

1.2 Rumusan Masalah

1. Bagaimana teori dasar dari *artificial neural network*?
2. Bagaimana cara untuk mengimplementasikan *artificial neural network* dengan menggunakan algoritma *back-propagation*?
3. Bagaimana cara untuk melakukan klasifikasi suatu dataset pada *artificial neural network* menggunakan algoritma *back-propagation*?

1.3 Tujuan Penelitian

1. Memahami teori dasar dari *artificial neural network*
2. Memahami teori dasar dari *back-propagation method*
3. Memahami cara untuk mengimplementasikan *artificial neural network* dengan menggunakan algoritma *back-propagation*
4. Memahami cara untuk mengimplementasikan *back-propagation* pada *dataset*
5. Memahami cara untuk melakukan klasifikasi dataset pada *artificial neural network* dengan menggunakan algoritma *back-propagation*

1.4 Manfaat Penelitian

Penulisan laporan tentang *Back-propogation Neural Network* (BPNN) dilakukan untuk memberikan manfaat bagi masyarakat seperti pemberian informasi yang bisa dijadikan referensi tentang *machine learning method* menggunakan *back-propogation neural network* (BPNN) yang merupakan bagian dari *artificial neural network* (ANN).

1.5 Metode Penelitian

- Melakukan simulasi program menggunakan software MATLAB pada setiap pasangan data *training* dan *testing*
- Melakukan proses *feed-forward* pada setiap pasangan data *train*ing dengan menggunakan bobot awal secara random yang diperoleh dari metode Nguyen-Widrow
- Menggunakan metode *back-propagation* untuk melakukan pengoreksian pada nilai bobot awal di setiap pasangan data *training*
- Menghitung nilai akurasi total dari hasil klasifikasi data *training* setiap *epoch*
- Melakukan proses *feed-forward* pada setiap pasangan data *train*ing dengan menggunakan bobot yang sudah dikoreksi dengan menggunakan metode *back-propagation*
- Menghitung nilai akurasi total dari hasil klasifikasi data *testing*
- Melakukan *update* bobot dan bias

BAB II
DASAR TEORI

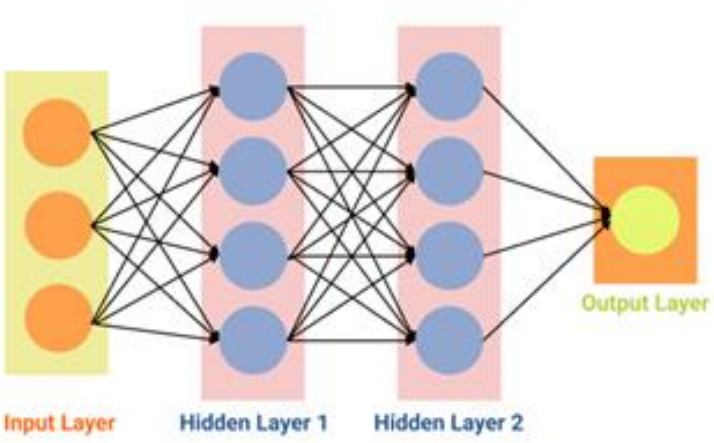
2.1 Artificial Neural Network (ANN)

Artificial neural network (ANN) adalah bentuk pemodelan yang dapat membuat suatu prediksi mengenai hubungan antara ekosistem merespon perubahan variabel lingkungan dengan terinspirasi oleh cara kerja sistem saraf biologis lebih lebih tepatnya pada sel otak manusia dalam melakukan proses informasi. Neural network terdiri atas sejumlah besar elemen pemrosesan informasi (neuron) yang saling terhubung dan bekerja secara bersama untuk menyelesaikan sebuah masalah tertentu seperti klasifikasi ataupun prediksi.

Algoritma dari ANN ada 2, yaitu *supervised learning* dan *unsupervised learning*. Pada supervised learning, sampel input dan output data yang cocok diberikan ke jaringan untuk training. Tujuan dari pendekatan ini adalah untuk mendapatkan output yang diinginkan untuk input tertentu. Jaringan unsupervised learning ANN adalah jaringan ANN yang lebih kompleks daripada supervised learning ANN karena unsupervised learning mencoba membuat ANN memahami struktur data yang diberikan sebagai input sendiri.

2.2 Lapisan yang Terdapat pada Artificial Neural Network (ANN)

Di dalam tubuh manusia, neuron terbagi menjadi 3 bagian, yaitu dendrites, cell body, dan axon. Dendrites diartikan sebagai sinyal masukan dan dipengaruhi oleh weight (bobot). Cell body diartikan sebagai tempat untuk proses komputasi sinyal. Axon diartikan sebagai bagian yang membawa sinyal output ke neuron lainnya. Dengan menganalogikan seperti sistem saraf biologis, ANN dapat direpresentasikan menjadi 3 bagian, yaitu *input layer*, *hidden layer*, dan *output layer*. *Input layer* adalah lapisan yang membawa data yang masuk ke dalam sistem yang kemudian data tersebut akan diproses di lapisan selanjutnya. *Hidden layer* adalah lapisan yang berada di antara *input layer* dan *output layer* yang berfungsi untuk memproses nilai bobot sehingga dihasilkan output yang sesuai. *Output layer* merupakan laposan yang menerima masukan dari lapisan sebelumnya dan berfungsi untuk memberikan informasi mengenai *output* dari sistem.



Gambar 1. Arsitektur Artificial Neural Network (ANN)

2.3 Prinsip Kerja Artificial Neural Network (ANN)

Artificial Neural Network (ANN) berfungsi untuk menyelesaikan suatu masalah dengan cara mempelajari data *testing* dan *training* yang telah diberikan dengan sendirinya dengan menggunakan satu perangkat yang sudah ada pola *training*-nya. Dengan mempelajari pola dari data *testing* dan *training* yang telah diberikan, ANN akan memproses nilai bobotnya hingga output yang dihasilkan sesuai dengan target yang telah ditentukan.

2.3.1 Faktor Bobot

Bobot adalah parameter penting dalam jaringan saraf yang mewakili kekuatan koneksi antara node yang satu dengan node yang lain. Semakin besar nilai bobot maka semakin kuat koneksi antar node. Bobot dapat disesuaikan dan perlu diatur supaya jaringan saraf berfungsi dengan baik. Jaringan saraf akan akan terus bertambah bobotnya karena jaringan saraf akan terus meningkatkan kemampuan belajarnya. ANN dapat belajar dari masalah baru ketika ada masalah baru muncul yang mana ANN akan langsung menyusun kembali nilai bobot supaya sesuai dengan nilai karakter yang ada.

2.3.2 Faktor Aktivasi

Tingkat aktivasi adalah keadaan internal yang dimiliki oleh setiap neuron yang mana neuron akan mengirimkan aktivitasnya sebagai sinyal ke neuron lainnya. Aktivasi mempunyai fungsi penjumlahan input dan bobot yang sampai ke neuron. Hasil penjumlahan tersebut akan diproses di masing-masing neuron oleh fungsi aktivasi yang mana hasilnya akan dibandingkan dengan nilai ambang. Jika hasil yang diperoleh di atas ambang batas maka aktivasi neuron akan dibatalkan. Sebaliknya, jika hasil yang diperoleh di bawah ambang batas maka neuron akan aktif. Setelah neuron diaktifkan,

neuron akan mengirimkan nilai *output* ke semua neuron yang ada kaitannya dengan bobot *output* dan *input* selanjutnya.

2.4 Parameter Artificial Neural Network (ANN)

Untuk mendapatkan model yang akurat dengan menggunakan perhitungan algoritma BPNN, dibutuhkan nilai parameter dan *hyperparameter* yang terbaik. Hyperparameter pada machine learning adalah parameter untuk mengendalikan proses pembelajaran.

Berikut adalah *hyperparameter* untuk algoritma BPNN :

1. **Learning Rate (α)**

   Learning rate adalah *hyperparameter* untuk mengatur besarnya perubahan bobot pada setiap epoch. Apabila nilai learning rate semakin besar maka perubahan bobot untuk setiap epoch akan semakin besar. Apabila nilai learning rate makin besar maka proses training akan mencapai fungsi minimum yang lebih cepat tetapi ada kelemahan, yaitu risiko ketidakakuratan pada tahap tersebut akan membesar.

2. **Momentum Coefficient (μ)**

   Momentum adalah suatu metode untuk mempertahankan arah dari *gradient descent*. Arah dari *gradient descent* didapatkan dari hasil kombinasi arah yang dikomputasi pada iterasi saat ini dengan iterasi sebelumnya. *Momentum coefficient* dapat digunakan untuk mengatur besar pengaruh arah iterasi sebelumnya.

3. **Layer Size (J)**

   Layer size adalah *hyperparameter* untuk menentukan keakuratan dan kualitas model. Apabila lapisan yang tersedia semakin banyak maka model yang terbentuk dari algoritma BPNN akan semakin akurat. Namun, jumlah lapisan tidak hanya berpengaruh pada keakuratan model tetapi juga bergantung pada kualitas model dan kualitas & kuantitas data *training*.

4. **Loss Function**

   Loss function adalah fungsi untuk menghitung perbedaan antara *output* algoritma dan *output* target. Hasil perhitungan *loss function* dapat digunakan untuk meng-*update* bobot.

5. **Epoch Count**

   Epoch count adalah *hyperparameter* untuk menentukan jumlah algoritma yang akan mengolah seluruh dataset *training*. Ketika jumlah epoch semakin banyak maka

bobot yang mengalami perubahan pada algoritma akan semakin banyak. Hal ini menyebabkan adanya peningkatan akurasi dari model tersebut.

Selain itu, ada 2 metode untuk inisialisasi awal dari tiap bobot antar neuron :

1. **Random**

   Metode inisialisasi secara *random* adalah metode untuk menentukan bobot awal secara acak yang mana penentuan bobot awal tersebut dapat dilakukan dengan menggunakan *random function* pada MATLAB. Tujuan dari inisialisasi awal secara random adalah untuk menghasilkan *symmetry breaking.* Konsep dari *symmetry breaking* adalah ketika besar bobot pada model sama maka fenomena yang terjadi akan tidak ideal. Dari konsep tersebut, bisa diterapkan untuk menentukan inisialisasi secara random dengan cara menentukan ambang batas atas dan bawah dengan tujuan agar bobot neuron yang dihasilkan tidak terlalu besar untuk proses pembelajaran.

2. **Nguyen-Widrow**

   Metode Nguyen-Widrow adalah suatu algoritma yang dapat memodifikasi bobot dengan tujuan untuk meningkatkan kecepatan proses pembelajaran.

   Berikut adalah cara -cara yang dilakukan dengan metode Nguyen-Widrow

   - Menentukan nilai dari faktor skala yang dilambangkan dengan β
   - Menentukan Inisialisasi bobot secara random pada rentang -0.5 sampai 0.5
   - Menghitung norm dari vektor bobot
   - Melakukan *update* bobot
   - Mengatur *set bias* mengunakan bilangan acak antara β sampai -β

2.5 Algoritma dari Pembelajaran Backpropagation

Backpropagation adalah metode pelatihan yang menggunakan metode ANN untuk menghitung turunan di dalam *deep feedforward neural networks*. *Backpropagation* membentuk bagian penting dari sejumlah algoritma supervised learning untuk training feedforward neural networks. Tujuan dari backpropogation adalah untuk meminimalkan error pada output yang dihasilkan oleh jaringan. Metode backpropagation biasanya menggunakan jaringan multilayer.

Ada 2 tahap perhitungan dari *backpropagation*, yaitu yang pertama melakukan perhitungan maju dan yang kedua menghitung error antara *output* dan target. Perhitungan maju digunakan untuk menghasilkan *output* dengan cara memberikan pola *input* ke dalam *artificial*

neural network (ANN). Perhitungan tersebut dapat memperbaiki bobot di semua neuron karena perhitungan tersebut melakukan *backpropagation* pada error. Pola input yang masuk ke dalam ANN akan diolah oleh setiap neuron yang menghasilkan tingkat aktivasi dengan tujuan untuk menghasilkan *output* ANN yang kemudian membandingkan output ANN tersebut dengan target yang telah ditentukan. Apabila terjadi perbedaan antara *output* ANN dan target maka akan menghasilkan error yang mana error tersebut akan dilakukan *backpropagation* sampai ke lapisan paling depan dengan tujuan untuk melakukan pengubahan bobot pada hubungan neuron.

Berikut adalah tahap dari *training* dan *testing* pada ANN :

1. **Training**

   Tahap *training* pada ANN adalah tahap untuk mendapatkan nilai bobot, bias, alpha dan μ. Berikut adalah tahap-tahap *training* pada ANN :

   a. Inisialisasi data

      Hal ini dilakukan pertama kali dengan menggunakan random value dan nguyen widrow. Pada inisialisasi data ini akan dibuat pembobotan yang dilakukan apakah data yang kita miliki dapat diproses selanjutnya atau tidak. Parameter yang menentukan kedua hal ini adalah α dan μ.

      1. Pembobotan awal (random value)

         Digunakan pembobotan awal yang ada dari rentang -0.5 hingga 0.5. Nilai tersebut secara acak ditempatkan pada pembobotan pada v dan w.

      2. Nguyen Widrow

         Metode ini digunakan untuk penentuan bobot pada *input layer* dan *hidden layer*. Bobot awal yang ditentukan sebelumnya akan mencari faktor skala

         $$\beta = 0.7h^{\frac{1}{x}}$$

         Selanjutnya, norma dari vektor bobot dicari dengan persamaan:

         $$||v_j|| = \sqrt{\sum_{i=1}^{P} {v_{ij}}^2}$$

b. Preprocessing

Preprocessing yang dilakukan adalah rangkaian normalisasi dengan mengubah rentang dari variabel yang digunakan. Hal ini dilakukan agar rentang yang kita miliki sama sehingga pengaruhnya akan sama terhadap data. Preprocessing data yang dilakukan menggunakan metode Z-score.

(masukin rumus z-score)

c. Training data

Pada training data yang dilakukan, feedforward, backpropagation, pembaruan bobot dan bias dilakukan terus menerus setelah adanya error value yang didapatkan pada hasil data hasil error. Nantinya proses ini akan berakhir apabila nilai dari error yang didapatkan sudah sesuai dengan nilai yang diinginkan.

- Feedforward

  Pada proses ini, ANN aka mengklasifikasi sampel data ke dalam kelas-kelas yang ditentukan sebelumnya. Feedforward memiliki algoritma sebagai berikut :

  1. Komputasi lapisan masukan. Untuk setiap masukan $x_i, i = 1,2,\dots,n$:
     a. Menerima input x_n
     b. Mengirimkan input ke *neuron-neuron* pada *hidden layer*.
  2. Komputasi lapisan tersembunyi. Untuk setiap unit tersembunyi $z_j, j = 1,2,\dots,p$:
     a. Menghitung sinyal masukan dari lapisan masukan dengan bobotnya dengan $zin_j = b_j + \sum \quad x_i v_{ij}$.
     b. Menghitung nilai aktivasi setiap *neuron* pada lapisan tersembunyi yang dinyatakan sebagai $z_j = \sigma(zin_j)$.
     c. Mengirimkan nilai aktivasi sebagai masukan pada *neuron-neuron* lapisan tersembunyi berikutnya atau lapisan keluaran.
  3. Komputasi lapisan keluaran. Untuk setiap keluaran $y_k, k = 1,2,\dots,m$:
     a. Menghitung sinyal masukan dengan bobotnya dengan$yin_k = b_k + \sum \quad z_j w_{jk}$.

b. Menghitung nilai aktivasi setiap *neuron* keluaran sebagai keluarannya dengan $y_k = \sigma(yin_k)$.

c. Kuantisasi nilai keluaran dengan ketentuan :

$$y_k = \begin{cases} 0 & ;0 \le y_k \le 0.3 \\ y_k & ;0.3 < y_k < 0.7 \\ 1 & ;0.7 \le y_k \le 1 \end{cases}$$

Setelah data selesai pada feedforward, data akan masuk kedaam backpropagation untuk keperluan optimasi dengan penentuan nilai kesalahan yang ada.

- Backpropagation

1. Komputasi error di lapisan keluaran. Untuk setiap keluaran$y_k, k = 1,2,\dots,m$:
   a. Menerima target yang bersesuaian dengan input
   b. Menghitung gradient descend dari fungsi nilai kesalahan dengan persamaan $\delta_k = (t_k - y_k).\sigma'(y_{in_k})$
   c. Menghitung besar koreksi bobot masukan dari *neuron* lapisan keluaran dengan $\Delta w_{jk} = \alpha\delta_k z$, di mana z adalah nilai aktivasi dari *neuron* lapisan tersembunyi sebelumnya
   d. Menghitung besar koreksi *bias neuron* keluaran dengan persamaan $\Delta b_k = \alpha\delta_k$
   e. Mengirimkan δ_k ke *neuron-neuron* pada lapisan sebelumnya
2. Komputasi kesalahan di lapisan tersembunyi. Untuk setiap unit tersembunyi $z_j, j = 1,2,\dots,p$:
   a. Menghitung semua koreksi error dengan $\delta_{inj} = \sum \delta_k w_{jk}$
   b. Menghitung nilai aktivasi koreksi kesalahan dengan $\delta_j = \delta_{in_j}\sigma'(z_{in_j})$
   c. Menghitung besar koreksi bobot pada lapisan tersembunyi dengan $\Delta v_{ij} = \alpha\delta_j x_i$

d. Menghitung besar koreksi bias lapisan tersembunyi dengan $\Delta b_j = \alpha \delta_j$

3. Pembaruan bobot dan bias

   Proses ini akan terus dilakukan, apabila data yang sebelumnya sudah ada, data tersebut akan diperbarui dengan fungsi berikut:

   Bobot output layer (w):

   $$w_{jk}\,(t+1) = w_{jk}\,(t) + \Delta w_{jk}$$

   $$w_{0k}\,(t+1) = w_{0k}\,(t) + \Delta w_{0k}$$

   Bobot hidden layer (v):

   $$v_{jk}\,(t+1) = v_{jk}\,(t) + \Delta v_{jk}$$

   $$v_{0k}\,(t+1) = v_{0k}\,(t) + \Delta v_{0k}$$

4. Menghitung error

   Proses penghitungan error yang dilakukan akan dilakukan terus menerus. Error yang digunakan adalah error kuadratik.

   $$error\ (E) = |t_{pk}m - y_{pk}|$$

2. Testing

Tahap *testing* adalah tahap untuk menguji data hasil *training* dengan data yang lain yang berbeda dengan menggunakan proses *feedforward* tanpa adanya perubahan bobot. *Output* yang dihasilkan dari tahap *testing* akan dibandingkan dengan data target pengujian. Hasil perbandingan tersebut akan didapatkan nilai *recognition rate* (RR) yang merupakan persentase dari keakuratan bobot dan bias terhadap data *input training*.

BAB III
PENJELASAN DATASET

Dataset yang digunakan adalah dataset *iris* yang merupakan salah satu dataset yang paling terkenal daalam ilmu data dan pembelajaran mesin. Dataset ini terdiri dari 150 sampel dari tiga spesies berbeda dari bungu iris, yakni Iris Sentosa, Iris Virginica, dan Iris Versicolor. Setiap sampel dalam dataset ini memiliki empat buah fitur pengukuran yang diambil dari bunga-bunga tersebut. Berikut ini adalah deskripsi detail dari dataset iris.

1. Jumlah data : 150 sampel
2. *Feature* :
    - Panjang Sepal (Sepal Length) adalah panjang sepal bunga iris dalam sentimeter.
    - Lebar Sepal (Sepal Width): Ini adalah lebar sepal bunga iris dalam sentimeter.
    - Panjang Petal (Petal Length): Ini adalah panjang petal bunga iris dalam sentimeter.
    - Lebar Petal (Petal Width): Ini adalah lebar petal bunga iris dalam sentimeter.
3. *Target* : tiga spesies dari bunga iris, yakni iris sentosa, iris virginica, dan iris versicolor.
4. Tujuan dataset : untuk melakukan klasifikasi atau pengenalan spesies bunga iris berdasarkan fitur-fitur pengukuran yang ada.

BAB IV
ALGORITMA

4.1 Data Preprocessing

4.1.1 Inisialisasi Data dan *Data Cleaning*

Model ANN yang akan dikembangkan adalah sebuah model yang berfungsi untuk memprediksi kelas dari spesies bunga iris, yakni Iris Sentosa, Iris Virginica, dan Iris Versicolor. Adapun pada proses inisialisasi ini, akan dihapus salah satu *feature*, yakni kolom "Id" karena kolom ini tidak diperlukan dalam proses pembelajaran model yang dibangun. Kolom "Id" tidak diperlukan dalam proses pembelajaran model karena kolom ini hanya menunjukkan jumlah sampel yang ada pada dataset dan tidak memiliki keterkaitan/korelasi dengan target yang ada pada dataset. Berikut ini adalah program untuk inisialisasi dataset yang akan digunakan dalam membuat model ANN khususnya BPNN.

```
% Import dataset
data_name = 'iris.csv';
data = readtable(data_name);

data = removevars(data, {'Id'});

% Mencari informasi umum dari dataset
column_names = data.Properties.VariableNames; % mengetahui
nama setiap kolom dari data
row_num = size(data, 1); % mengetahui jumlah baris dari data
column_num = size(data,2); % mengetahui jumlah kolom dari
data
```

4.1.2 One-Hot Encoding

Data target yang ada akan digambarkan dalam bentuk vektor biner. Proses ini dilakukan untuk mengatasi variabel kategori. Model ANN tidak akan dapat memproses data kategori dalam bentuk teks/string sehingga dengan menggunakan one-hot encoding dimungkinkan untuk mengubah data dalam bentuk teks/string yang dapat dipahami oleh algoritma.

```
% One Hot Encoding
i = 1;
while i <= column_num % mengecek setiap kolom
    if iscell(data.(column_names{i})) % mengecek apakah kolom
mengandung data teks

        keys = unique(data.(column_names{i}))'; % Mendapatkan
variabel unique dari satu kolom
        values = cell(1, size(keys, 2));
```

```
        % Mengubah variabel unique ke dalam bentuk angka
        for j = 1:size(keys, 2) % va
            values{j} = j;
        end

        % Membuat mapping dari angka ke teks unique
        map = containers.Map(keys, values);
        for j = 1:row_num
            data.(column_names{i}){j} =
map(data.(column_names{i}){j});
        end
        data.(column_names{i}) =
cell2mat(data.(column_names{i}));

        %one hot encoding
        tmp = double(data.(column_names{i}) ==
1:size(values,2));
        table_name = cell(1, size(tmp, 2));
        for j = 1:size(tmp, 2)
            table_name{j} = strcat((column_names{i}),
num2str(j));
        end

        tmp = array2table(tmp, 'VariableNames', table_name);

        % sisipkan one-hot encoding tersebut dan menghapus
yang lama
        data = [data(:, 1:i-1) tmp data(:, i+1:end)];
        i = size([data(:, 1:i-1) tmp], 2) + 1;
        column_names = data.Properties.VariableNames;
    else
        i = i + 1;
    end
end
```

4.2 Inisialisasi Model Artificial Neural Network (ANN)

4.2.1 Pemisahan Data Training dan Target

Proses ini dilakukan dengan menentukan target dari dataset.

```
% Pemisahan data
data_random = data(randperm(size(data,1)), :);

% Memisahkan feature dan target
feature = data_random(:, 1:end-3);
target = data_random(:, 6:end);

feature_num_row = size(feature, 1);
feature_num_column = size(feature, 2);

target_num_row = size(target, 1);
target_num_column = size(target, 2);
```

4.2.2 Pemisahan Data *Training* dengan Data *Testing*

Proses ini akan memisahkan data menjadi dua bagian, yaitu *training set* dan *test set*. Dalam pengembangan model *Artificial Neural Netowork* (ANN), dataset akan dipisahkan dengan rasio antara data *training* dan data *testing* yaitu sebesar 70:30.

```
% Pemisahan Data Training dengan Data Testing
train_size = 0.7 * feature_num_row;

X_train = table2array(feature(1:train_size, :));
y_train = table2array(target(1:train_size, :));

X_test = table2array(feature(train_size + 1:end, :));
y_test = table2array(target(train_size + 1:end, :));
```

4.2.3 Inisialisasi Variabel Input Layer, Hidden Layer, dan Output Layer

Variabel yang diinisialisasikan ini digunakan untuk menampung nilai yang ada pada neuron di *input layer*, *hidden layer*, dan *output layer*. Pada model yang dibuat, dilakukan inisialisasi variabel untuk dua buah *hidden layer* dengan lima buah neuron input dan empat buah neuron hidden, serta tiga buah neuron output.

```
% Melakukan Input Layer, Hidden Layer, dan Output Layer
n = feature_num_column; % input layer
p = 10;                 % hidden layer
m = target_num_column;  % output layer
```

4.2.4 Penentuan Nilai Bobot dan Bias Menggunakan Metode *Nguyen Widrow*

Setelah proses *data cleaning* selesai dilakukan, kemudian akan dijalankan proses penentuan nilai bobot yang ada di antara *input layer* dengan *hidden layer* pertama, *hidden layer* pertama dengan *hidden layer* kedua, dan *antara hidden layer kedua* dengan *output layer*. Adapun penentuan nilai bobot dan bias akan dilakukan dengan menggunakan metode Nguyen Widrow yang bertujuan untuk menentukan bobot yang digunakan pada model.

```
% Penentuan nilai bobot menggunakan metode nguyen widrow
a = -0.5;
b = 0.5;

V = rand(n, p) + a;
W = rand(p, m) - b;

beta_V = 0.7 * (p) .^ (1/n);
beta_W = 0.7 * (m) .^ (1/p);

V_0j = -beta_V + (beta_V - (-beta_V)) .* rand(1,p);
W_0k = -beta_W + (beta_W - (-beta_W) .* rand(1,m));
```

```
V_j = sqrt(sum(V.^2));
W_k = sqrt(sum(W.^2));

Vij_new = (beta_V .* V) / V_j;
Wjk_new = (beta_W .* W) / W_k;


% Inisialisasi nilai lama
W_jk_lama = 0;
W_0k_lama = 0;
V_ij_lama = 0;
V_0j_lama = 0;
```

4.2.5 Penentuan Parameter Iterasi

Proses ini dikenal dengan inisialisasi *hyperparameter*, dimana parameter-parameter iterasi akan diinisialisasikan, mulai dari jumlah epoch, error, flag, iterasi epoch, hingga *alfa* (laju pembelajaran/*learning rate*).

```
% Penentuan nilai parameter iterasi
iterasi = 1000;
iter = 0;
Ep_stop = 1;
alpha = 0.2;
miu = 0.6;
```

4.3 Training Model

4.3.1 Proses Iterasi pada Data Training

Algoritma ini dilakukan untuk men-*training* model dengan melakukan proses iterasi pada tiap *epoch* dan data *training*. Dengan demikian, total iterasi yang akan dilakukan oleh program yaitu *epoch* * jumlah data *training*.

4.3.2 Proses Feedforward

Pada proses *feedforward*, akan dilakukan kalkulasi untuk menentukan keluaran dari *hidden layer* dengan menggunakan data *testing* dan bobot yang telah diinisialisasikan sebelumnya. Setelah diperoleh keluaran dari *hidden layer*, selanjutnya data tersebut akan dijadikan masukan pada fungsi aktivasi. Adapun jenis fungsi aktivasi yang digunakan pada model ini adalah fungsi *sigmoid*.

4.3.3 Perhitungan Error

Setelah diperoleh diperoleh keluaran dari output layer, akan dilakukan proses perhitungan error pada epoch. Adapun metode perhitungan error yang digunakan pada model ini yaitu dengan menggunakan pendekatan *Mean Squared Error*.

4.3.4 Algoritma Backpropagation

Pada algoritma *backpropagation*, akan dilakukan proses kalkulasi untuk memperoleh hasil turunan dari fungsi aktivasi yang ada pada output layer. Setelah hasil turunan dari fungsi aktivasi diperoleh, maka proses kalkulasi untuk mendapatkan delta pada output layer akan dijalankan.

Setelah delta pada output layer diperoleh, selanjutnya akan dilakkan proses *update* terhadap nilai bobot dan bias yang digunakan pada model.

```
% Training data
while Ep_stop == 1 && iter < iterasi
    iter = iter + 1;
    for a = 1:length(X_train)
        % Proses feedforward
        z_inj = V_0j + X_train(a,:) * V;

        % Proses aktivasi menggunakan fungsi sigmoid
        for j = 1:p
            zj(1,j) = 1/(1+exp(-z_inj(1,j)));
        end

        y_ink = W_0k + zj * W;
        for r = 1:m
            yk(1,r) = 1/(1+exp(-y_ink(1,r))); % Aktivasi
sigmoid
        end

        % Menghitung nilai error
        E(1, a) = sum(abs(y_train(a,:) - yk));

        % Proses backpropagation
        do_k = (y_train(a,:) - yk) .* (yk .* (1-yk));
        W_jk = alpha * zj' * do_k + miu * W_jk_lama;
        W_0k = alpha * do_k + miu * W_0k_lama;

        W_jk_lama = W_jk;
        W_0k_lama = W_0k;

        do_inj = do_k * W';
        do_j = do_inj .* (zj .* (1-zj));
        V_ij = alpha * X_train(a,:)' * do_j + miu *
V_ij_lama;
        V_0j = alpha * do_j + miu * V_0j_lama;

        V_ij_lama = V_ij;
        V_0j_lama = V_0j;

        W = W + W_jk;
        W_0k = W_0k + W_0k;

        V = V + V_ij;
        V_0j = V_0j + V_0j;
    end
```

```
    % Menghitung nilai error pada tiap epoch
    Ep(1, iter) = sum(E) / length(X_train);

    if Ep(1,iter) < 0.01
        Ep_stop = 0;
    end
    acc_p(1,iter) = 1 - Ep(1, iter);
end
```

4.4 Data Testing

4.4.1 Proses Iterasi pada Data Testing

Algoritma ini dilakukan untuk melakukan proses *testing* pada model yang telah dibangun dengan melakukan proses iterasi pada tiap *epoch* dan data *testing*. Dengan demikian, total iterasi yang akan dilakukan oleh program yaitu *epoch* * jumlah data *training*.

4.4.2 Confusion Matrix

Proses ini dilakukan untuk menghitung *true value* yang pada nantinya nilai tersebut dapat digunakan untuk menghitung akurasi (*accuracy*) terhadap model yang telah dibangun.

4.4.3 Evaluasi Model Artificial Neural Network (ANN)

Pada tahap ini, akan ditampilkan nilai *true value* dan *false value* pada command window, yang dimana nilai tersebut diperoleh dari proses *testing* terhadap data. Proses kalkulasi terhadap akurasi dari model yang telah di-*train* juga akan dilakukan dengan memanfaatkan data *true value* yang diperoleh. Adapun nilai akurasi dari model juga akan ditampilkan pada command window.

```
% Melakukan testing
E_test = zeros(1,length(X_test));
right = 0;
wrong = 0;

for a = 1:length(X_test)
    z_inj_test = X_test(a,:)*V + V_0j;
    % Proses aktivasi menggunakan sigmoid

    for j=1:p
        zj_test(1,j) = 1/(1+exp(-z_inj_test(1,j))); %Aktivasi
sigmoid
    end

    y_ink_test = zj_test * W + W_0k;

    for k=1:m
```

```
        yk_test(1,k) = 1/(1+exp(-y_ink_test(1,k))); %Aktivasi
sigmoid
    end

    for j = 1:m
        predict(a,j) = yk_test(j);
    end

    %Menghitung nilai error
    E_test(1,a) = sum(abs(y_test(a,:) - yk_test));

    [value, index] = max(yk_test);
    Y_test = zeros(train_size, target_num_column);
    Y_test(a, index) = 1;

    if Y_test(a, :) == y_test(a,:) % Menghitung jumlah
prediksi benar
        right = right + 1;

    else % Menghitung jumlah prediksi salah
        wrong = wrong + 1;
    end
end



% Evaluasi hasil data testing
avgerrortest = sum(E_test)/length(X_test);
recog_rate = (right/length(X_test))*100;

accuracy = right/(right+wrong);
error_rate = wrong/(right+wrong);

figure;
plot(Ep);
ylabel('Error'); xlabel('Iterasi')

figure;
plot(acc_p);
ylabel('Accuracy'); xlabel('Iterasi')

disp("Didapat nilai error " + Ep(end) + " pada iterasi ke-" +
iter);
disp("Accuracy rate = " + (accuracy*100) + "%");
disp("Error rate    =  " + (error_rate*100) + "%");
disp("Accuracy rate pada training sebesar " + (acc_p(end)*100)
+ "%");
disp(" ");
disp("Pada test diperoleh: ");
disp("Error sebesar " + E_test(end) + " pada iterasi ke-" +
iter);
disp("Jumlah Benar  = " + right + "/" + (right + wrong));
```

BAB IV
ANALISIS HASIL PROGRAM

A. Analisis Kelompok

Dataset Iris merupakan salah satu dataset yang bagus digunakan untuk menguji suatu model *machine learning* dalam tugasnya untuk pengenalan pola dan klasifiaksi karena sifatnya yang sederhana dan memiliki keragaman yang baik. Dapat diketahui jika dataset memiliki 4 kolom feature dan 1 kolom target. Dataset ini menggambarkan tiga buah spesies bunga iris (Iris sentosa, Iris versicolor, dan Iris virginica) dengan empat buah fitur pengukuran (panjang dan lebar sepal, serta panjang dan lebar petal) untuk masing-masing bunga.

Keempat *feature* yang ada pada dataset *iris* sudah cukup bersih dan siap digunakan untuk algoritma BPNN. Dalam hal ini, tidak perlu terdapat *feature* yang diubah menjadi bentuk numerik, ataupun dinormalisasi. Adapun untuk *target* masih berupa string/teks sehingga harus diubah menjadi data numerik dengan metode one-hot encoding.

Kemudian dilakukan proses pembagian data training dan testing dengan perbandingan 70% dan 30%. Data yang dipisah menjadi data training dan testing sebelumnya diacak terlebih dahulu karena beberapa alasan, yakni untuk menghindari bias, mengurangi *overfitting*, serta memastikan keacakan dalam validasi silang.

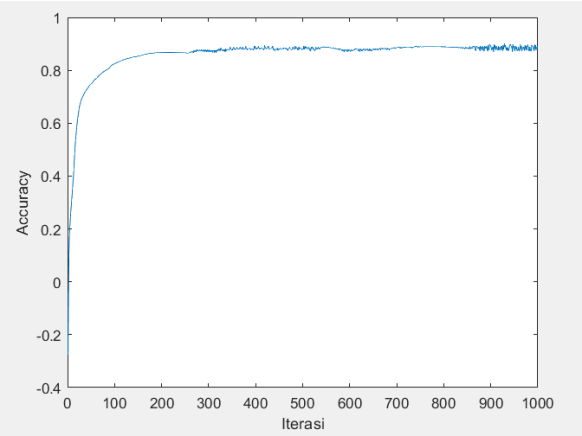
Tabel 1. Hasil Pembelajaran dengan 70% *data training* dan *30% data testing*

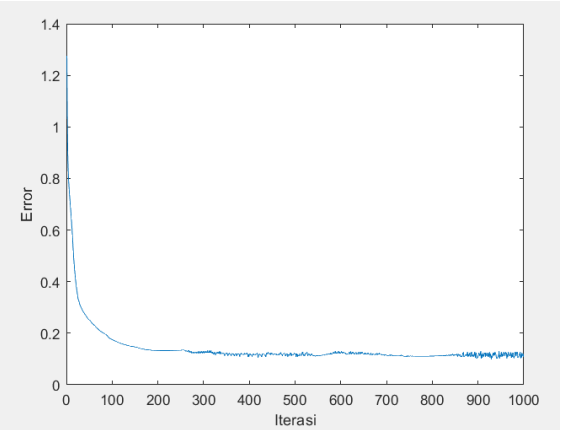
Variasi	Variasi Input					Hasil Pembelajaran	
	Jumlah Input	Jumlah Hidden Neuron	Laju Pembelajaran	Miu	Iterasi	Recognition Rate	Error iterasi terakhir
1	4	15	0.2	0.4	1000	97.7778%	0.0045548
	4	15	0.2	0.4	5000	95.5556%	0.0004589
	4	15	0.2	0.8	1000	93.3333%	0.014532
	4	15	0.2	0.8	5000	97.7778%	0.0016734
2	4	15	0.4	0.4	1000	97.7778%	0.029384
	4	15	0.4	0.4	5000	95.5556%	0.0050248
	4	15	0.4	0.8	1000	68.8889%	0.0040875
	4	15	0.4	0.8	5000	73.3333%	0.55011

3	4	15	0.8	0.4	1000	95.5556%	0.0016399
	4	15	0.8	0.4	5000	97.7778%	6.837e-05
	4	15	0.8	0.8	1000	52.1111%	0.0055164
	4	15	0.8	0.8	5000	64.4444%	1.9958
4	4	50	0.2	0.4	1000	100%	0.016598
	4	50	0.2	0.4	5000	97.7779%	0.0072203
	4	50	0.2	0.8	1000	97.7778%	0.18358
	4	50	0.2	0.8	5000	88.8880%	0.050312
5	4	50	0.4	0.4	1000	97.7778%	0.022341
	4	50	0.4	0.4	5000	97.7778%	0.0019751
	4	50	0.4	0.8	1000	62.2222%	1.9999
	4	50	0.4	0.8	5000	62.2222%	2.0002
6	4	50	0.8	0.4	1000	100%	0%
	4	50	0.8	0.4	5000	64.4444%	0.015706
	4	50	0.8	0.8	1000	26.6667%	1.0012
	4	50	0.8	0.8	5000	33.3333%	1

Berikut ini merupakan hasil percobaan yang telah dilakukan dari beberapa variasi nilai parameter model dengan jumlah iterasi sebanyak 1000.

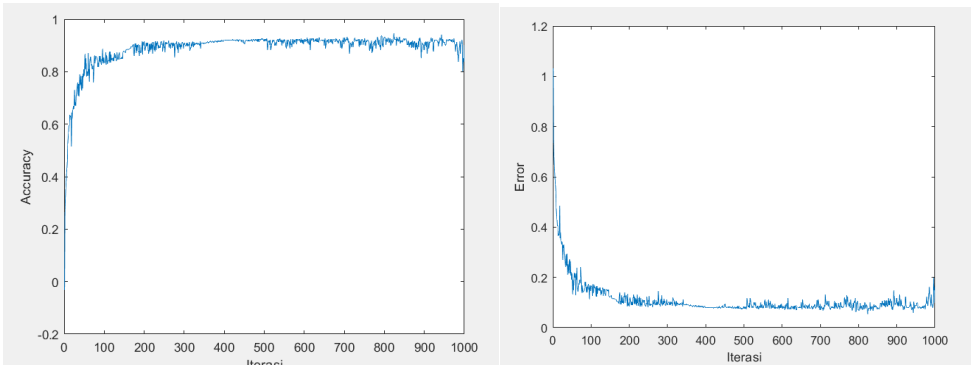
- Hidden neuron = 15, alpha = 0.2, miu = 0.4





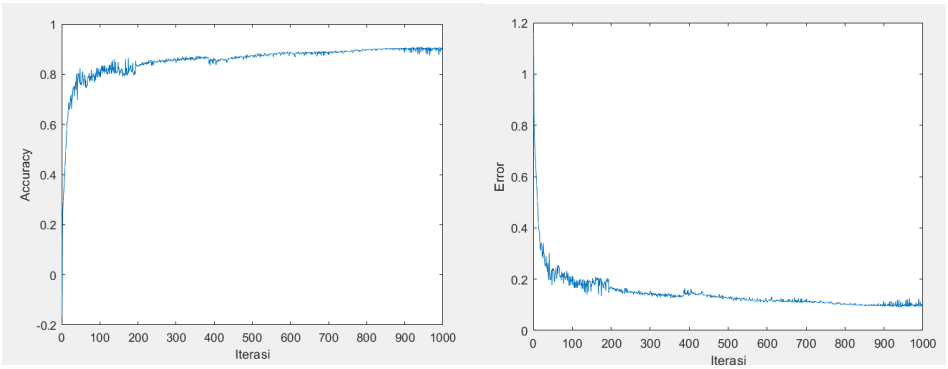

```
Didapat nilai error 0.12245 pada iterasi ke-1000
Accuracy rate = 97.7778%
Error rate    =  2.2222%
Accuracy rate pada training sebesar 87.755%

Pada test diperoleh:
Error sebesar 0.0045548 pada iterasi ke-1000
Jumlah Benar  = 44/45
```

- Hidden neuron = 15, alpha = 0.2, miu = 0.8



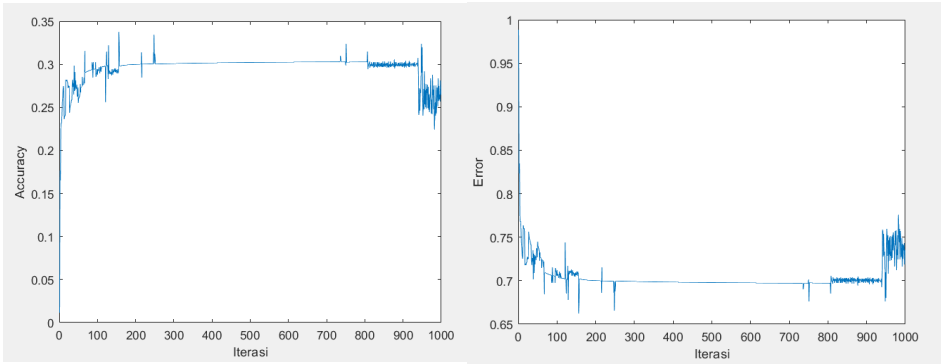

```
Didapat nilai error 0.097136 pada iterasi ke-1000
Accuracy rate = 93.3333%
Error rate    =  6.6667%
Accuracy rate pada training sebesar 90.2864%

Pada test diperoleh:
Error sebesar 0.014532 pada iterasi ke-1000
Jumlah Benar  = 42/45
```

- Hidden neuron = 15, alpha = 0.4, miu = 0.4



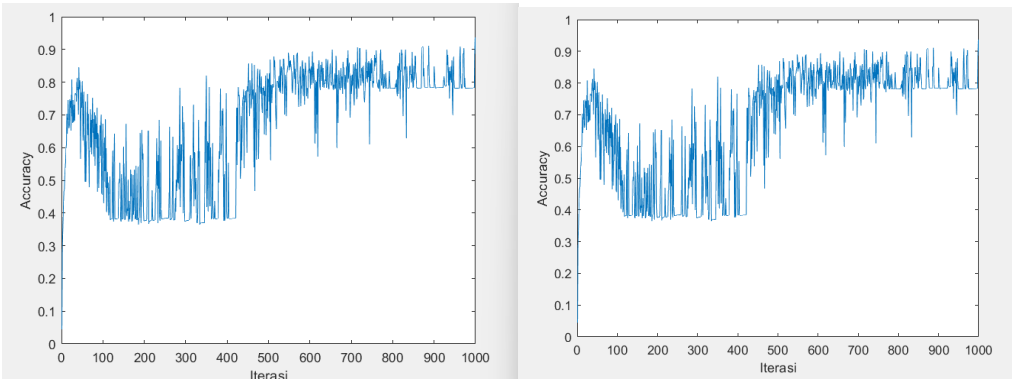

```
Didapat nilai error 0.096903 pada iterasi ke-1000
Accuracy rate = 97.7778%
Error rate    =  2.2222%
Accuracy rate pada training sebesar 90.3097%

Pada test diperoleh:
Error sebesar 0.029384 pada iterasi ke-1000
Jumlah Benar  = 44/45
```

- Hidden neuron = 15, alpha = 0.4, miu = 0.8



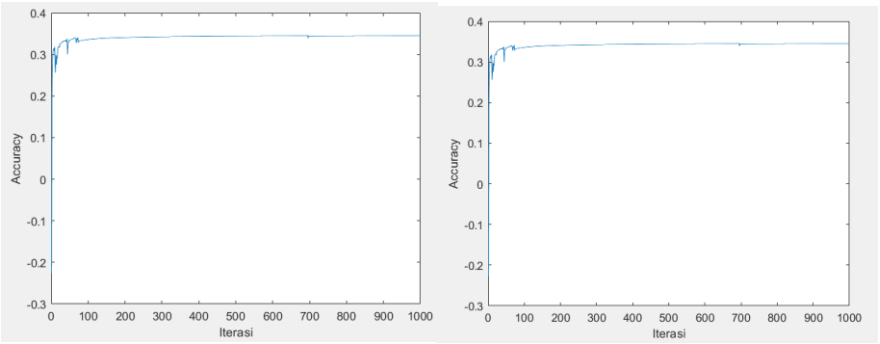

```
Didapat nilai error 0.73563 pada iterasi ke-1000
Accuracy rate = 68.8889%
Error rate    =  31.1111%
Accuracy rate pada training sebesar 26.4366%

Pada test diperoleh:
Error sebesar 0.0040875 pada iterasi ke-1000
Jumlah Benar  = 31/45
```

- Hidden neuron = 15, alpha = 0.8, miu = 0.4



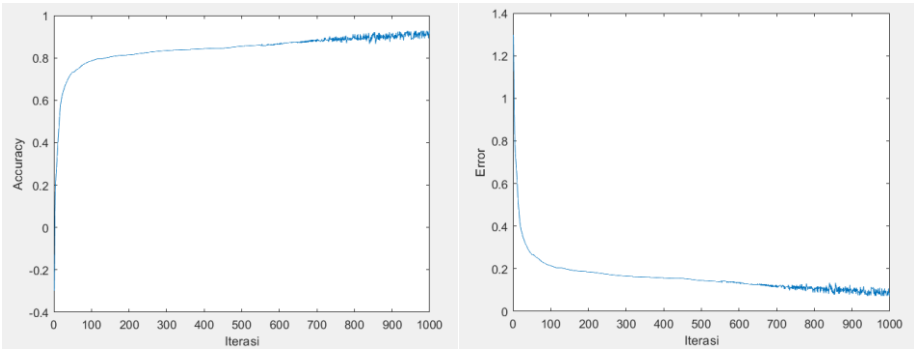

```
Didapat nilai error 0.10023 pada iterasi ke-1000
Accuracy rate = 95.5556%
Error rate    =  4.4444%
Accuracy rate pada training sebesar 89.9769%

Pada test diperoleh:
Error sebesar 0.0016399 pada iterasi ke-1000
Jumlah Benar  = 43/45
```

- Hidden neuron = 15, alpha = 0.8, miu = 0.8



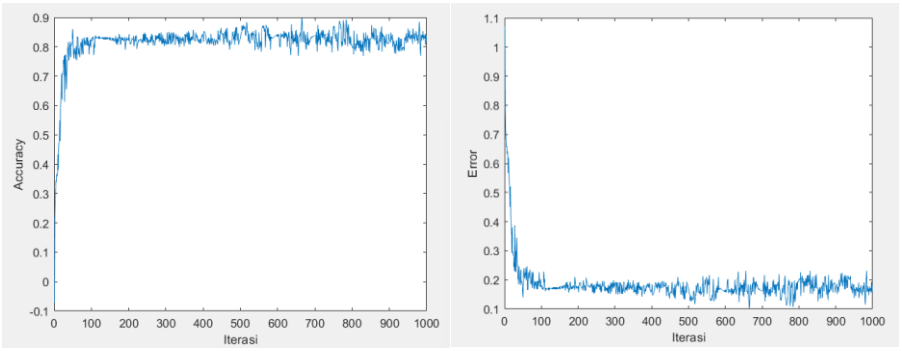

```
Didapat nilai error 0.65511 pada iterasi ke-1000
Accuracy rate = 51.1111%
Error rate    =  48.8889%
Accuracy rate pada training sebesar 34.4893%

Pada test diperoleh:
Error sebesar 0.0055164 pada iterasi ke-1000
Jumlah Benar  = 23/45
```

- Hidden neuron = 50, alpha = 0.2, miu = 0.4



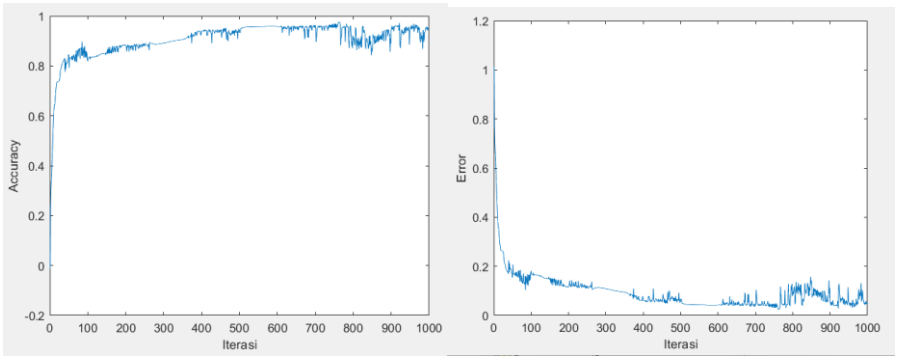

```
Didapat nilai error 0.069463 pada iterasi ke-1000
Accuracy rate = 100%
Error rate    =  0%
Accuracy rate pada training sebesar 93.0537%

Pada test diperoleh:
Error sebesar 0.016598 pada iterasi ke-1000
Jumlah Benar  = 45/45
```

- Hidden neuron = 50, alpha = 0.2, miu = 0.8



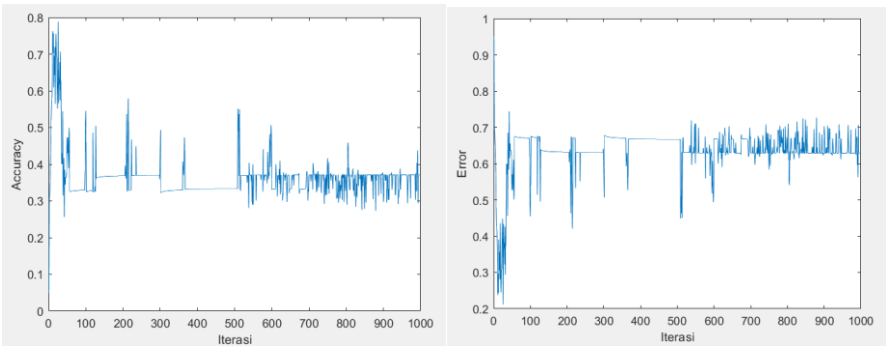

```
Didapat nilai error 0.16371 pada iterasi ke-1000
Accuracy rate = 97.7778%
Error rate    =  2.2222%
Accuracy rate pada training sebesar 83.6288%

Pada test diperoleh:
Error sebesar 0.18358 pada iterasi ke-1000
Jumlah Benar  = 44/45
```

- Hidden neuron = 50, alpha = 0.4, miu = 0.4



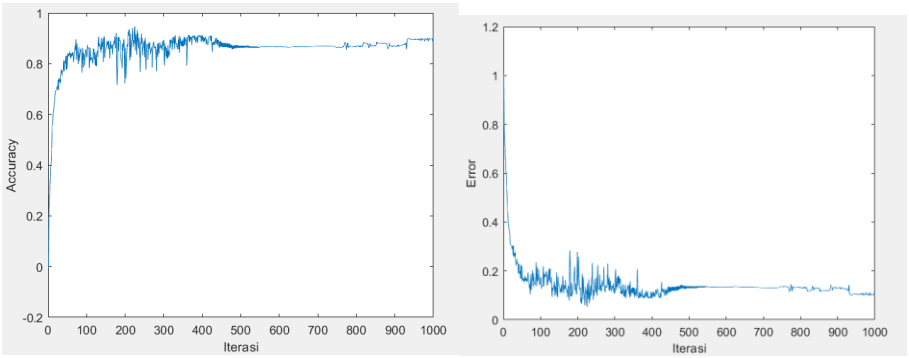

```
Didapat nilai error 0.047624 pada iterasi ke-1000
Accuracy rate = 95.5556%
Error rate    =  4.4444%
Accuracy rate pada training sebesar 95.2376%

Pada test diperoleh:
Error sebesar 0.00028061 pada iterasi ke-1000
Jumlah Benar  = 43/45
```

- Hidden neuron = 50, alpha = 0.4, miu = 0.8



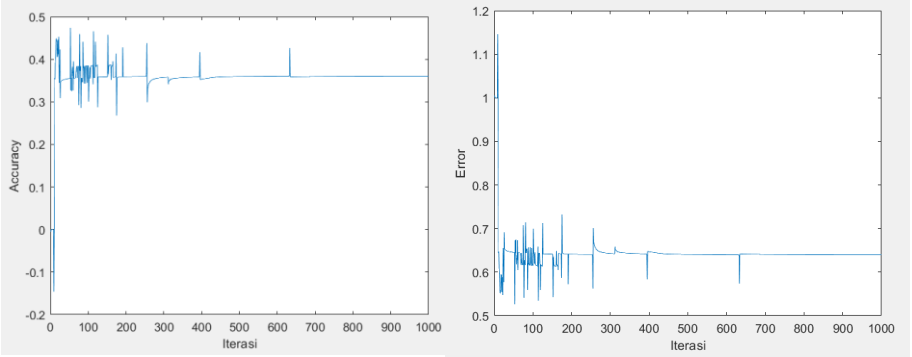

```
Didapat nilai error 0.62885 pada iterasi ke-1000
Accuracy rate = 62.2222%
Error rate    =  37.7778%
Accuracy rate pada training sebesar 37.1151%

Pada test diperoleh:
Error sebesar 1.9999 pada iterasi ke-1000
Jumlah Benar  = 28/45
```

- Hidden neuron = 50, alpha = 0.8, miu = 0.4




```
Didapat nilai error 0.10057 pada iterasi ke-1000
Accuracy rate = 100%
Error rate    =  0%
Accuracy rate pada training sebesar 89.9435%

Pada test diperoleh:
Error sebesar 0.0054954 pada iterasi ke-1000
Jumlah Benar  = 45/45
```

- Hidden neuron = 50, alpha = 0.8, miu = 0.8



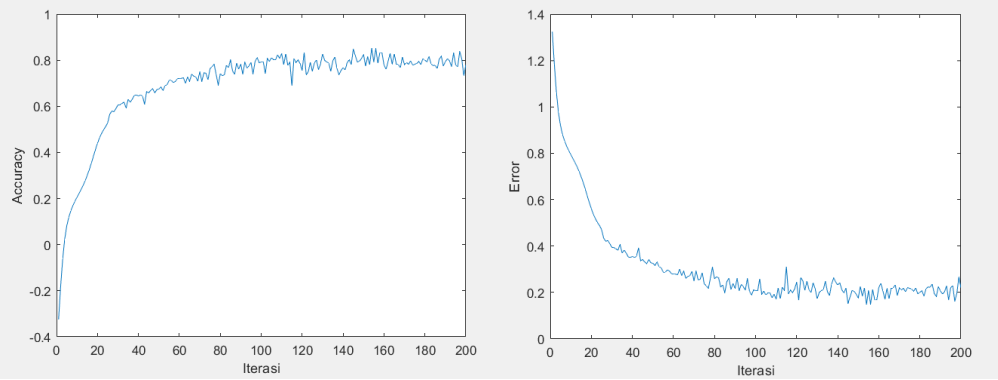

```
Didapat nilai error 0.63953 pada iterasi ke-1000
Accuracy rate = 26.6667%
Error rate    =  73.3333%
Accuracy rate pada training sebesar 36.0474%

Pada test diperoleh:
Error sebesar 1.0012 pada iterasi ke-1000
Jumlah Benar  = 12/45
```

Pada percobaan yang telah dilakukan, dapat diketahui jika perubahan nilai parameter model memberikan perubahan yang signifikan terhadap tingkat akurasi dari model BPNN yang telah dibangun. Diperoleh tingkat akurasi terkecil yakni sebesar 26.6667% dengan rincian parameter model yaitu jumlah hidden neuron sebanyak 50, alpha sebesar 0.8, miu sebesar 0.8, dan jumlah iterasi yang dilakukan sebanyak 1000. Sedangkan, tingkat akurasi terbesar yang didapat yakni 100% dengan rincian parameter model yaitu jumlah hidden neuron sebanyak 50, alpha sebesar 0.2, miu sebesar 0.4, dan jumlah iterasi yang dilakukan sebanyak 1000, serta percobaan dengan jumlah hidden neuron sebanyak 50, alpha sebesar 0.8, miu sebesar 0.4, dan jumlah iterasi yang dilakukan sebanyak 1000.

Dari percobaan yang dilakukan, dapat diketahui jika nilai alpha dan miu memiliki tingkat pengaruh yang tinggi terhadap tingkat akurasi model. Hal sebaliknya berlaku pada

jumlah hidden neuron dan jumlah iterasi yang dilakukan. Hal tersebut tentunya bukanlah sesuatu yang kebetulan terjadi.

Seperti yang telah disinggung sebelumnya, dataset *iris* merupakan dataset yang bersih, sederhana, dan memiliki keragaman yang baik. Hal ini menyebabkan pengujian terhadap dataset *iris* tidak memerlukan model BPNN yang kompleks dengan jumlah iterasi yang panjang. Berikut ini dilakukan percobaan pengujian dataset *iris* dengan jumlah hidden neuron sebanyak 5, alpha sebesar 0.2, miu sebesar 0.4, dan jumlah iterasi sebanyak 200.




```
Didapat nilai error 0.21915 pada iterasi ke-200
Accuracy rate = 97.7778%
Error rate    =  2.2222%
Accuracy rate pada training sebesar 78.0853%

Pada test diperoleh:
Error sebesar 0.079432 pada iterasi ke-200
Jumlah Benar  = 44/45
```

Dapat dilihat jika tingkat akurasi yang diperoleh tetaplah tinggi. Hal ini mengindikasikan bahwa jumlah hidden neuron dan iterasi yang banyak tidaklah memberikan dampak yang signifikan terhadap tingkat akurasi dari model yang telah dibangun.

Hal yang berbeda terjadi pada nilai parameter alpha dan miu yang diterapkan pada model. Alpha merupakan salah satu parameter penting pada algoritma ANN. Alpha atau disebut juga dengan laju pembelajaran digunakan untuk mengendalikan sejauh mana bobot (weights) dari koneksi antar neuron diperbarui selama proses pelatihan atau pembelajara. Dengan kata lain, alpha menentukan seberapa besar atau seberapa kecil perubahan bobot yang akan diterapkan pada setiap iterasi pelatihan berdasarkan kesalahan (error) yang dihasilkan oleh model.

Nilai alpha yang tepat merupakan hal yang penting. Laju pembelajaran yang terlalu besar dapat menyebabkan model divergen, yaitu kehilangan kemampuan untuk mencapai konvergen karena perubahan bobot yang terlalu besar dan tidak stabil. Di sisi lain, laju pembelajaran yang terlalu kecil dapat menyebabkan model konvergen terlalu lambat atau terjebak dalam minimum lokal.

Sedangkan laju pembelajaran momentum (miu) merupakan suatu teknik dalam pelatiham model ANN yang mengkombinasikan dua konsep, yakni laju pembelajaran dan momentum. Nilai laju pembelajaran momentum, digunakan untuk membantu percepatan konvergensi dan mengatasi masalah terjebak dalam minimum lokal dan plateau (stagnansi/lambat dalam hal perbaikan performa) saat melatih model ANN.

Nilai alpha dan miu yang tepat akan menghasilkan performa model yang maksimal. Hal ini dapat diketahui dari kecenderungan perubahan nilai performa model seiring dengan peningkatan/penurunan nilai alpha dan miu. Dengan nilai alpha dan miu yang optimal, akan dihasilkan tingkat akurasi yang tinggi, sekitar 90% - 100%, serta grafik peningkatan nilai akurasi dan penurunan nilai error dengan *ripple* yang kecil.

B. Analisis Personal

Dari hasil percobaan yang telah dilakukan, dapat diketahui jika model *Artificial Neural Network* (ANN) yang telah dibangun sudah memberikan performa yang sangat baik hampir disemua variasi nilai parameter model. Performa dari model sangat memuaskan dengan tingkat akurasi sebagian besar di rentang 90% - 100%. Performa yang optimal dari percobaan yang dilakukan dipengaruhi oleh dua buah faktor utama, yakni kualitas dari dataset dan arsitektur model yang digunakan pada model ANN. Dataset dan model yang baik, tentu akan memberikan tingkat keakuratan model yang maksimal.

Dalam suatu model *Artificial Neural Network* (ANN), kumpulan dataset yang berkualitas akan selalu memberikan hasil yang efektif (baik), meskipun arsitektur model dari sistem dapat dibilang cukup sederhana. Secara umum, terdapat beberapa parameter pentung yang harus ada dalam dataset untuk keefektifan dari model yang dibangun.

1. Kualitas yang bagus dari dataset. Ibarat suatu suara, tingkat kebisingan yang rendah tentu merupakan salah satu fitur kritis dalam masalah klasifikasi suara.
2. Kuantitas dari dataset. Pada dasarnya, *Artificial Neural Network* (ANN) membutuhkan data yang sangat besar untuk dilatih (*training*). Oleh sebab itu, tingkat akurasi dapat dipengaruhi oleh kuantitas dari dataset.
3. Variabilitas dari data. Variasi dari semua kemungkinan kasus yang ada pada dataset merupakan hal penting yang dapat meningkatkan kualitas dari dataset.

Dari sisi arsitektur model *Artificial Neural Network* (ANN) yang dibangun, terdapat beberapa hal yang dapat dipertimbangkan, di antaranya adalah sebagai berikut.

1. Arsitektur. Apakah model yang digunakan sesuai dengan masalah yang ada pada dataset? Terkadang penggunaan model *Artificial Neural Network* (ANN) untuk melakukan klasifikasi suatu masalah tertentu bukanlah merupakan ide yang bagus. Ini merupakan hal penting yang perlu dipertimbangkan sebelum memulai *machine learning* dengan menggunakan ANN.
2. *Overtraining*. Pelatihan model secara berlebihan. Oleh karena itu, proses penambahan beberapa lapisan dropout terkadang dapat membantu mengatur model menjadi lebih baik.
3. *Tuning hyperparameter*. Proses *tuning hyperparameter* yang digunakan pada model *Artificial Neural Network* (ANN) perlu dilakukan untuk mendapatkan kemungkinan parameter terbaik untuk jumlah *layer*, *learning rate* (alpha), *optimizer*, dan lain-lain.

Terkait dengan dataset yang digunakan, dapat diketahui jika dataset *iris* memanglah sebuah dataset berkualitas yang dapat memberikan hasil prediksi yang akurat. Tidak heran jika dataset *iris* merupakan salah satu faktor penting penyebab performa model yang sangat baik.

Terkait dengan arsitektur model ANN, berdasarkan percobaan yang telah dilakukan, dapat diketahui jika perubahan nilai parameter model akan memberikan dampak yang cukup signifikan terhadap tingkat akurasi/performa dari model. Nilai parameter dari model ini bisa berupa jumlah hidden neuron, nilai laju pembelajaran (alpha), laju pembelajaran momentum (miu), serta jumlah iterasi yang dilakukan selama proses pelatihan model.

Pada dasarnya, nilai alpha dan miu memiliki fungsi yang tidaklah jauh berbeda. Adapun kedua nilai parameter ini harus diatur sedemikian rupa sehingga dihasilkan performa model yang maksimal. Pada rujukan algoritma ANN yang digunakan, direkomendasikan nilai alpha yang digunakan yaitu 0.2 – 0.4 dan miu sebesar 0.7-0.8. Dengan digunakannya nilai alpha dan miu berdasarkan rujukan tersebut, akan diperoleh tingkat akurasi dari model yang sangat tinggi (seperti yang ditunjukkan pada tabel hasil percobaan).

Pada tabel hasil percobaan, dapat dilihat kecenderungan hasilnya yaitu tingkat akurasi dari model akan mengecil ketika nilai alpha dan miu yang diberikan pada model cukup besar nilainya. Hal ini dapat terjadi karena pada proses pelatihan model, akan terjadi peristiwa "overshooting". Ini merupakan peristiwa dimana model "melompat" terlalu jauh ke arah yang salah ketika mencoba menemukan minimum global dari *loss function*. Akibatnya, model akan melewatkan nilai minimum yang diinginkan dan akan kesulitan untuk mencapai konvergen.

Tidak hanya itu, nilai laju pembelajaran dan laju pembelajaran momentum yang terlalu besar dapat menyebabkan pelatihan model akan menjadi tidak stabil sehingga model akan mengalami divergensi. Divergensi terjadi ketika bobot dan bias model terus meningkat hingga mencapai nilai yang terlalu besar. Oleh karena itu, diperlukan nilai alpha dan miu yang tepat sehingga dihasilkan performa model yang optimal.

Tidak hanya alpha dan miu, jumlah hidden neuron dan jumlah iterasi juga perlu disesuaikan agar model memiliki efektivitas yang baik. Jumlah hidden neuron dan iterasi yang banyak tidak selalu memberikan hasil yang lebih baik. Sebaliknya, jumlah hidden neuron dan iterasi yang sedikit tidak selalu memberikan hasil yang lebih buruk.

Dataset *iris* merupakan dataset yang dapat dibilang cukup sederhana. Tidak heran jika dataset *iris* sudah dapat diproses dengan baik menggunakan model ANN yang sederhana. Jumlah hidden neuron yang banyak tidaklah terlalu dibutuhkan dalam membangun model ANN untuk memproses dataset *iris*. Model ANN yang kompleks akan terkesan "overkill" jika hanya digunakan untuk memproses dataset *iris*.

Penggunaan jumlah hidden neuron yang terlalu banyak pada dataset yang sederhana juga dapat menyebabkan terjadinya peristiwa "overfitting". Peristiwa ini terjadi ketika model mempelajari data pelatihan dengan sangat baik, bahkan sampai ke tingkat detail kecil yang mungkin hanyalah sebuah *noise* dalam data. Akibatnya, model menjadi terlalu spesifik untuk data pelatihan dan tidak dapat menggeneralisasi dengan baik untuk data yang tidak terlihat sebelumnya (tidak adaptif)

Hal yang sama berlaku juga pada jumlah iterasi. Jumlah iterasi yang banyak hanya diperlukan jika digunakan dataset yang besar, serta model ANN yang kompleks. Hal ini disebabkan karena dataset yang besar dapat membuat pelatihan model akan memerlukan waktu yang lebih lama. Model ANN yang kompleks juga memerlukan lebih banyak iterasi untuk mempelajari representasi yang baik dari data. Itulah sebabnya untuk dataset *iris*, jumlah iterasi yang dilakukan tidak perlu terlalu banyak. Hal ini terbukti dengan percobaan sebelumnya yang menghasilkan tingkat akurasi tinggi dengan hanya 200 iterasi.

Selain hal-hal yang telah disebutkan sebelumnya, jumlah hidden neuron dan iterasi yang terlalu banyak dapat menyebabkan komputasi yang berlebihan. Hal ini membuat pelatihan dan inferensi akan menjadi lebih lambat dan membutuhkan lebih banyak sumber daya komputasi sehingga waktu yang diperlukan untuk menjalankan model akan menjadi lebih lama.

Dari penjelasan-penjelasan tersebut dapat disimpulkan jika untuk memproses dataset *iris*, tidak diperlukan model ANN yang terlalu kompleks dengan jumlah hidden neuron yang sangat banyak. Jumlah iterasi yang dilakukan juga tidak perlu terlalu banyak untuk menghindari peristiwa *overfitting* dan lamanya waktu komputasi. Nilai laju pembelajaran dan laju pembelajaran momentum pada model ANN juga harus disesuaikan sedemikian rupa sehingga tidak terjadi peristiwa *overshooting*.

BAB V
KESIMPULAN

Berdasarkan percobaan yang dilakukan, penulis dapat menyimpulkan beberapa hal, di antaranya adalah:

- Bervariasinya laju pembelajaran pada algoritma BPNN akan mempengaruhi seberapa cepat program mempelajari sekumpulan data. Semakin kecil nilai laju pembelajaran, maka kecenderungan program untuk mencapai target error dengan jumlah iterasi yang terbatas itu semakin mengecil. Sebaliknya, jika nilai laju pembelajaran semakin besar, program akan lebih cepat mencapai target error bahkan sebelum batas iterasi yang ditentukan.
- Kegunaan dari Nguyen-Widrow adalah untuk mempercepat model mendapatkan titik konvergensi pada pelatihaan data (data training).
- Ada beberapa tahapan sebelum memodelkan *Artificial Neural Network* (ANN), seperti halnya membersihkan data (data cleaning) jika ada data yang bukan numerik, lalu, melakukan normalisasi data jika diperlukan, lalu melakukan pengacakan data dan pemisahan data menjadi data training dan data testing.
- Terdapat 2 metode dalam menenrukan bobot (*wight*) dan bias pada model BPNN, yaitu metode random dan metode Nguyen-Widrow. Yang mana metode yang kami gunakan adalah metode Nguyen-Widrow.
- Bervariasinya nilai koefisien momentum juga dapat berpengaruh pada model. Jika nilai tersebut semakin kecil, penurunan error akan menajdi lebih lembat. Namun, jika nilai koefisien momentum semakin besar, penurunan grafik akan semakin cepat yang menyababkan akan tercapainya titik target error.
- Jumlah variasi hidden layer yang terlalu besar dapat menghasilkan model yang lebih baik namun jika terlalu besar juga dapat menyebabkan waktu menjalankan model yang lebih lama karena nilai-nilai yang belum pernah diidentifikasi sebelumnya.
- Penentuan nilai parameter algoritma seperti nilai laju pembelajaran (alfa), laju pembelajaran momentum (miu), dan jumlah hidden neuron layer mempengaruhi diperolehnya nilai error, akurasi, dan jumlah iterasi untuk menentukan durasi ANN dalam mencapai konvergensi.
- Nilai laju pembelajaran (alfa), laju pembelajaran momentum (miu), dan jumlah hidden neuron layer berpengaruh terhadap durasi ANN untuk mencapai konvergensi dan potensi untuk mencapai konvergensi untuk dataset *iris* yang digunakan.

REFERENSI

[1] B. Kusumaputro, Artificial Neural Network, 2022