IMPLEMENTASI MODEL ARTIFICIAL NEURAL NETWORK (ANN) DENGAN METODE PEMBELAJARAN BACKPROPAGATION UNTUK SISTEM KENDALI MENGGUNAKAN MATLAB

TUGAS - 02

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 kedua pada mata kuliah Sistem Berbasis Pengetahuan yang berjudul "Implementasi Model *Artificial Neural Network* (ANN) dengan Metode Pembelajaran *Backpropagation* untuk Sistem Kendali Menggunakan MATLAB".

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. Dengan melakukan proses *training* pada bobot neural network, *error rate* dapat berkurang sehingga model yang dibuat akan memiliki performa yang lebih baik. Pada laporan penelitian ini, penulis akan memanfaatkan algoritma BPNN untuk membuat pengendali dari suatu sistem kendali dengan menggunakan model ANN. Dalam prosesnya, akan dilakukan proses *training*, validasi, dan *testing* menggunakan 12 ribu sampel data yang diperoleh dari *plant* yang ada. Dengan menggunakan algoritma BPNN, akan dicari nilai bobot terbaik dari model ANN yang akan diterapkan sebagai pengendali.

Kata kunci: *Artificial Neural Network, backpropagation, kendali, plant, loop tertutup*

DAFTAR ISI

BAB 1
PENDAHULUAN

2.1 Latar Belakang

Artificial Neural Network (ANN) atau dalam bahasa Indonesia disebut dengan jaringan saraf tiruan merupakan suatu konsep pengolahan informasi dengan menggunakan model komputasi yang terinspirasi dari struktur dan fungsi jaringan saraf biologis manusia. ANN merupakan bagian dari pembelajaran mesin dan merupakan inti dari algoritma *deep learning*.

Nama dan struktur dari ANN terinsiprasi dari otak manusia. ANN akan bekerja dengan cara meniru proses neuron pada otak manusia ketika mengirimkan sinyal antara satu dengan yang lainnya. Model ANN terdiri dari jaringan neuron buatan (disebut juga dengan unit pemrosesan) yang saling terhubung satu sama lain untuk membentuk jaringan saraf buatan. Neuron buatan dalam model ANN ini akan bekerja sama untuk memproses suatu informasi masukan, mempelajari pola-pola yang ada dalam data, dan melakukan tugas-tugas tertentu seperti klasifikasi, *pattern recognition*, hingga kendali.

Pada laporan ini, akan dibahas mengenai implementasi model ANN dengan metode pembelajaran *backpropagation* untuk suatu sistem kendali. Jadi, model ANN yang dibangun akan digunakan sebagai pengendali dari *plant* yang ada. Dalam hal ini, percobaan akan dilakukan pada dua tahapan yang berbeda. Pada masing-masing tahap, akan dilakukan tiga buah proses utama, yakni *training*, validasi, dan *testing*.

Pada tahap pertama, akan disimulasikan sistem kendali lup terbuka dengan menggunakan model ANN. Berdasarkan data masukan berupa bilangan acak pada rentang tertentu, serta data keluaran *plant* pada satu dan dua waktu sebelumnya, akan dilakukan proses *training* untuk memperoleh nilai bobot terbaik, validasi untuk mengetahui kapan model mulai mengalami *overfitting*, serta *testing* untuk mengetahui performa dari model yang telah dibangun.

Pada tahap kedua, akan disimulasikan sistem kendali lup tertutup menggunakan model ANN dengan bobot yang diperoleh dari proses *training* dari tahap sebelumnya. Pada tahap ini, akan digunakan masukan berupa bilangan acak pada rentang tertentu, serta data keluaran model ANN (hasil prediksi) pada satu dan dua waktu sebelumnya (data *feedback*). Sama seperti tahap satu, akan dilakukan juga *training*, validasi, dan *testing*. Pada akhirnya, data keluaran dari model ANN akan dibandingkan dengan data keluaran dari *plant* untuk mengetahui bagaimana performa dari model ANN sebagai pengendali pada suatu sistem kendali.

Pada laporan ini akan dijelaskan pemodelan ANN untuk diimplementasikan sebagai pengendali dari sistem kontrol. Bilangan random antara range tertentu akan dimanfaatkan sebagai input utama ke dalam plant. Data output didapatkan dari data masukan utama, data masukan pada waktu berikutnya, dan data keluaran pada waktu sebelumnya yang diproses oleh plant. Keseluruhan data keluaran yang didapatkan secara langsung dari plant akan dikomparasi dengan data keluaran model ANN dengan memanfaatkan data masukan yang sudah didapatkan sebelumnya.

2.2 Rumusan Masalah

- Bagaimana proses yang perlu dilakukan untuk mendapatkan model ANN yang memiliki performa bagus sebagai pengendali pada suatu sistem kendali?

- Bagaimana pengaruh dari proses validasi terhadap performa dari model ANN sebagai pengendali pada suatu sisem kendali?
- Bagaimana pengaruh dari parameter-parameter model ANN, seperti laju pembelajaran, laju pembelajaran momentum, hingga jumlah *hidden neuron* terhadap performa model ANN sebagai pengendali dari suatu sistem kendali?
- Bagaimana performa dari model ANN sebagai pengendali secara keseluruhan?

2.3 Tujuan Penelitian

1. Mengembangkan model ANN yang memiliki performa bagus sebagai pengendali pada suatu sistem kendali melalui proses *training*, validasi, dan *testing* menggunakan algoritma *backpropagation*.
2. Mengetahui pengaruh dari proses validasi terhadap pemilihan bobot yang tepat dari model ANN untuk menghindari terjadinya *overfitting*.
3. Menganalisis pengaruh dari parameter-parameter model ANN, seperti laju pembelajara, laju pembelajaran momentum, hingga jumlah *hidden neuron* terhadap performa model ANN sebagai pengendali suatu sistem kendali.
4. Mengetahui performa model ANN sebagai pengendali.

2.4 Manfaat Penelitian

Penulisan laporan tentang "Implementasi Model *Artificial Neural Network* (ANN) dengan Metode Pembelajaran *Backpropagation* untuk Sistem Kendali Menggunakan MATLAB" dilakukan untuk memahami bagaimana tingkat keefektifan penggunaan model ANN sebagai pengendali di sistem kendali. Diharapkan laporan ini dapat memberikan manfaat bagi penulis, maupun masyarakat mengenai performa dari model ANN sebagai pengendali di suatu sistem kendali.

2.5 Metode Penelitian

Pada penelitian ini, akan dilakukan simulasi penerapan model ANN sebagai pengendali di suatu sistem kendali dengan membuat simulasi program MATLAB pada proses generasi data *plant*, training, validasi, hingga *testing*. Berikut adalah rincian tahapan yang dilakukan selama proses simulasi model ANN untuk sistem kendali pada tahap 1.

- Membuat suatu *database* dengan menggunakan data masukan berupa bilangan acak dari -1 hingga 1 di waktu sekarang dan satu waktu sebelumnya, dan data keluaran dari *plant* di satu waktu sebelumnya (nilai awal untuk data masukan dan keluaran di satu waktu sebelumnya pada awal proses generasi data bernilai nol). Dari proses ini, akan diperoleh data keluaran dari *plant* di waktu tersebut.
- Menentukan data *feature*, sebagai input model ANN, dan *target* sebagai data sebenarnya (keluaran dari *plant*) dari *dataset* yang ada. *Feature* dan *target* ini akan digunakan pada proses *training*, validasi, dan *testing* dari model ANN.
- Melakukan proses *feedforward* pada setiap pasangan data pelatihan menggunakan bobot awal acak yang diperoleh dengan menggunakan metode inisialisasi Nguyen-Widrow.

- Melakukan proses *backpropagation* dengan menghitung gradien *descent* dari *loss function*/fungsi kesalahan terhadap nilai bobot dalam jaringan. Pada proses ini, nilai bobot dari model ANN akan dikoreksi selama proses *training* berlangsung.
- Menghitung nilai total error kuadratik pada keluaran data pelatihan (*train*) terhadap data keluaran dari *plant* di setiap epoch.
- Melakukan proses validasi di setiap 50 epoch dengan melakukan proses *feedforward* pada setiap pasangan data validasi dengan menggunakan bobot terbaru.
- Mengambil nilai bobot terbaik dari model berdasarkan proses *training* dan validasi untuk menghindari *overfitting* dengan mengamati nilai total error kuadratik yang diperoleh melalui proses *training* dan validasi. Jika nilai total error kuadratik yang diperoleh melalui proses validasi meningkat pada epoch tertentu, maka bobot pada epoch tersebut akan digunakan pada proses *testing*.
- Melakukan proses *feedforward* pada setiap pasangan data pengujain dengan menggunakan bobot terbaik yang sudah dikoreksi.
- Menghitung nilai total error kuadratik pada keluaran data pengujian terhadap data keluaran dari *plant* (data sebenarnya).

Pada tahap kedua, dilakukan proses yang sama. Perbedaannya yaitu untuk bobot awal yang digunakan pada tahap 2 akan diperoleh dari bobot terbaik pada tahap 1. Adapun nilai input yang digunakan pada proses *training*, validasi, dan *testing* dari model ANN akan menggunakan data masukan saat ini, satu waktu, dan dua waktu sebelumnya (dari *plant* sama seperti tahap 1), serta data keluaran dari model ANN pada satu waktu dan dua waktu sebelumnya (proses *feedback*, bukan dari *plant* seperti tahap 1).

2.6 Sistematika Penulisan

- Bab I merupakan bagian pendahuluan yang meliputi latar belakang, rumusam masalah, tujuan penelitian, manfaat penelitian, metodologi, dan sistematika penulisan.
- Bab I merupakan bagian dasar teori dari ANN, sistem kendali, dan model ANN untuk sistem kendali.
- Bab III merupakan bagian deskripsi *plant* dan dataset, program ANN pada sistem kendali, pengujian model dan pembahasan, perbandingan keluaran *plant* dengan keluaran model ANN, pengujian parameter-parameter model terhadap total error kuadratik sebagai indikator performa dari model ANN.
- Bab IV merupakan bagian kesimpulan akhir dari keseluruhan percobaan yang telah dilakukan.

BAB II
DASAR TEORI

2.1 Artificial Neural Network (ANN)

Artificial neural network (ANN) atau disebut juga dengan istilah jaringan saraf tiruan merupakan bentuk pemodelan yang memiliki cara kerja yang mirip dengan otak manusia. ANN akan mempelajari sebuah data untuk memberikan nilai-nilai yang sesuai sehingga melalui data masukan yang diberikan, akan dihasilkan data keluaran yang sesuai dengan data sebenarnya. Dalam hal ini, data keluaran yang dihasilkan dapat berupa klasifikasi ataupun estimasi.

Artificial Neural Network (ANN) terdiri dari lapisan input, satu atau lebih lapisan *hidden*, dan lapisan output. Setiap *node* atau neuron buatan pada model ANN akan terhubung satu sama lain dan memiliki bobot. Bobot inilah yang akan menentukan seberapa baik performa dari model ANN yang telah dibangun.

Model ANN akan mengandalkan data pelatihan untuk mempelajari dan meningkatkan akurasinya seiring dengan berjalannya waktu dengan menggunakan metode seperti *backpropagation*. Selama pelatihan, bobot-bobot dari model akan diatur sedemikian rupa sehingga dapat dihasilkan suatu model ANN dengan performa yang bagus. Model ANN dengan performa yang bagus akan menjadi sebuah alat yang ampuh dalam ilmu komputer dan kecerdasan buatan yang memungkinkan digunakan untuk melakukan klasifikasi, mengelompokkan data, ataupun memprediksi suatu nilai tertentu berdasarkan data masukan yang tidak pernah dilihat sebelumnya.

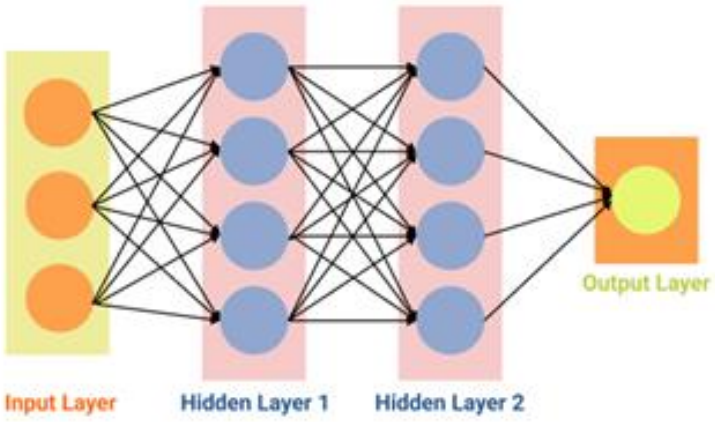
Suatu model ANN akan memiliki tiga buah komponen utama, yakni neuron, koneksi, dan struktur jaringan. Neuron merupakan unit pemrosesan pada model ANN. Neuron akan memiliki beberapa input, mengalirkan sinyal melalui fungsi matematika tertentu, dan menghasilka output. Input dari neuron merupakan perkalian antara nilai bobot dengan sinyal keluaran dari neuron sebelumnya. Adapun output dari neuron akan berupa nilai aktivasi dari akumulasi semua masukan yang ada. Output dari neuron ini akan diteruskan ke neuron berikutnya hingga sampai ke neuron pada lapisan paling akhir.

Neuron pada model ANN akan saling terhubung melalui koneksi yang memiliki bobot. Bobot ini merupakan nilai parameter yang harus disesuaikan selama proses pelatihan jaringan. Bobot yang optimal akan memberikan model ANN yang memiliki performa yang tinggi. Terakhir adalah struktur jaringan dari model ANN yang memiliki berbagai jenis, termasuk jaringan *feedforward*, rekursif, dan berbagai arsitektur lainnya. Struktur jaringan dari ANN akan mempengaruhi performa model ANN untuk menangani tugas-tugas tertentu.

2.2 Lapisan pada Model Artificial Neural Network (ANN)

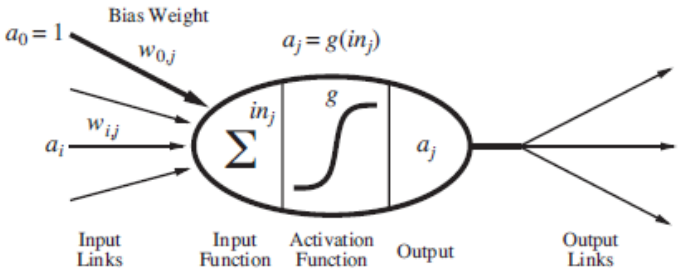
Di dalam tubuh manusia, neuron terbagi menjadi 3 bagian, yaitu *dendrites*, *cell body*, dan *axon*. *Dendrites* diartikan sebagai sinyal masukan dan dipengaruhi oleh *weight* (bobot). *Cell body* diartikan sebagai tempat untuk proses komputasi sinyal. *Axon* diartikan sebagai bagian yang membawa sinyal 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* terdiri dari beberapa neuron yang berfungsi untuk menerima data masukan. Setiap neuron yang ada pada *input layer* akan mewakili satu *feature* atau variabel pada data yang dimasukkan ke dalam jaringan. Setiap *neuron* pada *input layer* juga akan terhubung dengan setiap neuron yang ada di dalam *hidden layer*.
- *Hidden layer* adalah lapisan yang berada di antara *input layer* dan *output layer* yang berfungsi untuk meproses dan mempelajari data masukan sehingga dihasilkan nilai bobot yang optimal. Setiap neuron pada *hidden layer* akan mengambil masukan dari laposan sebelumnya, melakukan komputasi dengan bobot dan fungsi aktivasi, dan mengirimkan sinyal keluarannya ke lapisan berikutnya.
- *Output layer* merupakan lapisan terakhir dalam jaringan yang menerima masukan dari lapisan sebelumnya dan berfungsi untuk memberikan informasi mengenai *output* yang sesuai dari sistem berdasarkan data masukan (dan bobot) yang ada. *Output layer* mewakili hasil akhir dari pemrosesan jaringan, seperti klasifikasi, regresi, *pattern recognition*, dan lain sebagainya.



Gambar 2.1 Arsitektur Artificial Neural Network (ANN)

2.3 Prinsip Kerja Artificial Neural Network (ANN)



Gambar 2.2 Model Matematis Neuron

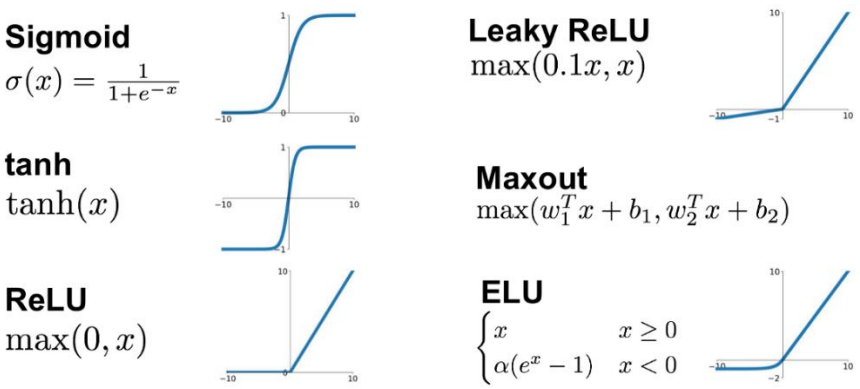
Artificial Neural Network (ANN) berfungsi untuk menyelesaikan suatu masalah dengan cara mempelajari data *testing* dan *training* yang telah diberikan dengan sendirinya dengan menggunakan satu perangkat yang sudah ada pola *training*-nya. Dengan mempelajari pola dari data *testing* dan *training* yang 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.



Gambar 2.3 Jenis-Jenis Fungsi Aktivasi

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 $zin_j = b_j + \sum x_i v_{ij}$.
   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$yin_k = b_k + \sum z_j w_{jk}$.
   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

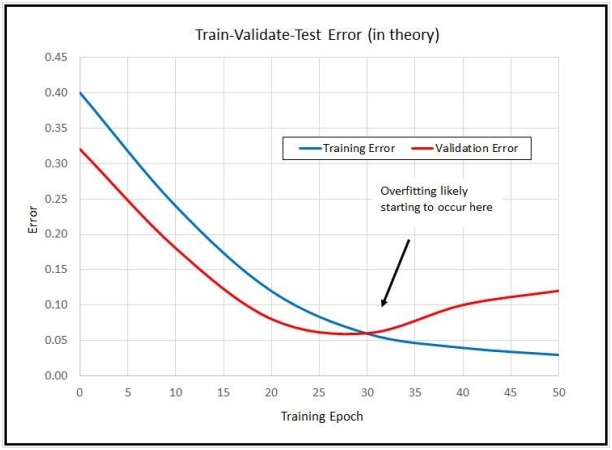
d. Menghitung besar koreksi *bias neuron* keluaran dengan persamaan $\Delta b_k = \alpha \delta_k$
e. Mengirimkan δ_k ke *neuron-neuron* pada lapisan sebelumnya

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. Validasi

Tahap validasi merupakan tahap untuk mengukur sejauh mana model ANN yang telah dilatih mampu menggeneralisasi dari data pelatihan ke data yang tidak pernah dilihat sebelumnya. Melalui tahap validasi, dimungkinkan untuk mengevaluasi sejauh mana model ANN mampu beradaptasi terhadap berbagai macam jenis data masukan. Sederhananya, tahapan validasi pada model ANN merupakan proses *testing* terhadap bobot yang sudah dikoreksi yang dilakukan selama proses *training* masih berlangsung. Jadi, proses validasi ini akan dilakukan di tengah-tengah proses *training* sedang berlangsung. Proses validasi ini akan dilakukan setiap proses *training* sudah dilakukan sebanyak 50 epoch. Dengan validasi, akan diamati bagaimana perkembangan nilai bobot dari model untuk menghindari terjadinya peristiwa *overfitting* selama proses *testing*.



Gambar 2.4 Grafik Error Proses *Training* dan Validasi

Dari grafik tersebut dapat diketahui jika nilai error dari validasi akan meningkat di titik tertentu. Adapun nilai error dari proses *training* dalam hal ini tetap menurun nilainya. Peningkatan nilai error validasi di tengah-tengah proses *training* ini mengindikasikan bahwa model kehilangan kemampuan untuk menggeneralisasi dan akan tampil buruk saaat diterapkan pada data yang tidak pernah dilihat sebelumnya (*overfitting*). Oleh karena itu, ketika gejala *overfitting* ini mulai tampak terlihat, proses *training* harus segera dihentikan atau nilai bobot terakhir sebelum nilai error validasi meningkat harus segera dicatat untuk digunakan pada proses *testing*.

3. 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*.

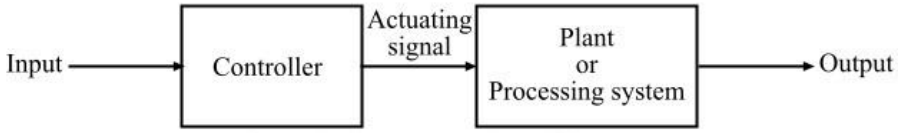
2.6 Sistem Kendali

Sistem kendali merupakan suatu sistem yang dapat digunakan untuk mengatur atau mengendalikan perilaku atau operasi dari suatu sistem ataupun proses. Tujuan utama dari sistem kendali yaitu untuk mencapai hasil yang diinginkan atau menjaga suatu variabel dalam batas yang ditentukan. Sistem kendali digunakan hampir di semua bidang, seperti di antaranya adalah otomotif, indutrsi, elektronika, dan lain sebagainya.

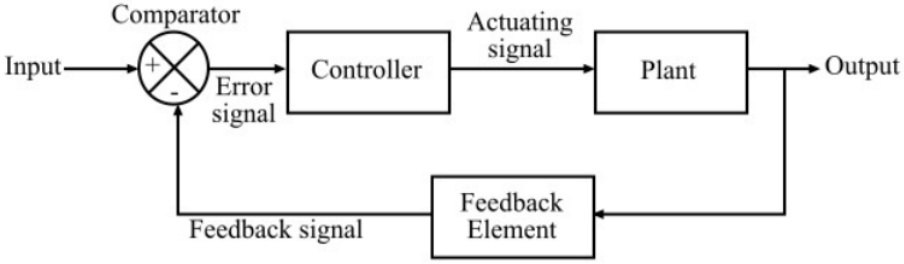
Sistem kendali terdiri dari beberapa komponen utama, di antaranya adalah input, pengendali, *plant*, dan output. Input merupakan masukan dari sistem pengendali. Input dapat meliputi nilai referensi agar nilai output bisa sesuai dengan nilai referensi yang diberikan. Selanjutnya, pengendali merupakan otak dari sistem kendali. Pengendali berguna untuk memproses nilai informasi masukan dengan algoritma tertentu sehingga akan dihasilkan sinyal kendali untuk mengendalikan *plant* agar dihasilkan respons keluaran sesuai dengan apa yang diinginkan. *Plant* merupakan perangkat yang akan dikendalikan oleh sistem kendali. Terakhir, output merupakan respons keluaran dari *plant* ketika diberikan sinyal kendali keluaran dari pengendali.

Pada praktiknya, keluaran dari *plant* akan bernilai konvergern pada keadaan tunak, memiliki nilai *steady state error* yang kecil, dan memiliki karakteristik respons transien yang

bagus jika *plant* dikendalikan dengan pengendali yang bagus. Sistem kendali dibagi menjadi dua jenis utama, yakni sistem kendali terbuka dan sistem kendali tertutup. Sistem kendali terbuka tidak menggunakan umpan balik (*feedback*) dari sistem untuk membuat keputusan. Pengendali akan mengirimkan sinyal kendali tanpa memperhatikan respons aktual dari sistem. Adapun sistem kendali tertutup akan menggunakan umpan balik (*feedback*) untuk memantau kondisi aktual dari sistem sehingga akan diperoleh nilai error. Pada prosesnya, nilai error ini akan diperkecil dengan menggunakan pengendali sehingga respons keluaran dari sistem bisa sesuai dengan *set-point* yang diberikan. Dengan proses inilah, sistem kendali tertutup biasanya akan menghasilkan respons keluara yang lebih baik dan akurat jika dibandingkan dengan sistem kendali terbuka.



Gambar 2.5 Sistem Open Loop



Gambar 2.6 Sistem Closed Loop

2.7 Perbedaan ANN untuk *Pattern Recognition* dengan ANN untuk Sistem Kendali

Dalam prosesnya untuk melakukan percobaan ini, perlu diketahui terlebih dahulu perbedaan penggunaan ANN dalam *pattern recognition* (klasifikasi, klusterisasi, dan lain sebagainya) dengan penggunaan ANN dalam sistem kendali. Pada dasarnya, ukuran sukses pada *pattern recognition* dapat diukur dengan menggunakan *recognition rate*, yakni berapa persen nilai benar/salah dari semua data yang di-*test* dengan menggunakan ANN. Adapun pada ANN dalam sistem kendali, ukuran sukses dapat dilihat dengan menghitung nilai MSE (*mean squared error*), yaitu perbedaan antara hasil keluaran ANN dengan data seharusnya (error kuadratik).

Pada ANN untuk *patterrecognition*, input yang diberikan dapat memiliki nilai dimensi yang sangat besar, hingga mencapai ribuan. Berbeda dengan ANN untuk sistem kendali, dimana dimensi inputnya umumnya tidak terlalu banyak karena hanya berganting pada jumlah input dan jumlah output dari *plant* yang digunakan.

Pada ANN untuk *pattern recognition*, proses normalisasi dilakukan dengan menggunakan *range* angka mulai dari 0 sampai 1. Hal yang berbeda terjadi pada ANN untuk sistem kendali dimana normalisasi harus dilakukan dengan menggunakan range dari -1 sampai dengan 1. Hal ini disebabkan karena daerah kerja dari sistem kendali yang bisa dimulai dari angka negatif.

Pada ANN untuk *pattern recognition*, dimensi data masukan akan mempunyai rentang nilai yang sama antar dimensi inputnya. Sedangkan data masukan pada ANN sistem kendali akan mempunyai rentang nilai yang dapat berbeda antar dimensi inputnya.

BAB III
ALGORITMA

3.1 Deskripsi Sistem

Pada pemodelan ANN untuk sistem kendali, model dari data masukan dan data keluaran dapat dirumuskan dengan menggunakan persamaan sebagai berikut.

$$\boldsymbol{y(k) = f[y(k-1), y(k-2), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)]}$$

dimana,

$\boldsymbol{y(k)}$ = keluaran sistem pada waktu sekarang (k)

$\boldsymbol{u(k)}$ = masukan sistem pada waktu sekarang (k)

$\boldsymbol{f(k)}$ = fungsi nonlinear yang tidak diketahui

Nilai n dam m merupakan variabel yang menujukkan orde dari sistem. Kedua variabel ini akan mempengaruhi seberapa jauh data lampau akan digunakan untuk menghitung data keluaran dari sistem kendali ANN.

Sistem yang digunakan pada percobaan ini merupakan sistem orde dua dengan nilai n dan m sama dengan 2. Artinya, *plant* akan memiliki fungsi keluaran yakni sebagai berikut.

$$\boldsymbol{y(k) = f[y(k-1), u(k), u(k-1)]}$$

dimana,

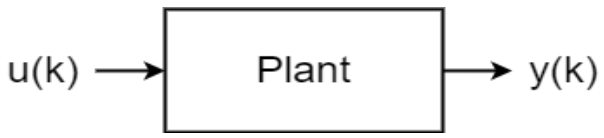
$\boldsymbol{y(k-1)}$ = keluaran sistem satu langkah sebelumnya $(\boldsymbol{k-1})$

$\boldsymbol{u(k)}$ = masukan sistem pada waktu sekarang $(\boldsymbol{k})$

$\boldsymbol{u(k-1)}$ = masukan sistem satu langkah sebelumnya $(\boldsymbol{k-1})$

Berikut ini merupakan persamaan dari *plant* sistem yang akan digunakan pada percobaan.

$$\boldsymbol{y(k) = \frac{1}{1 + y(k-1)^2} + 0.25u(k) - 0.3u(k-1)}$$

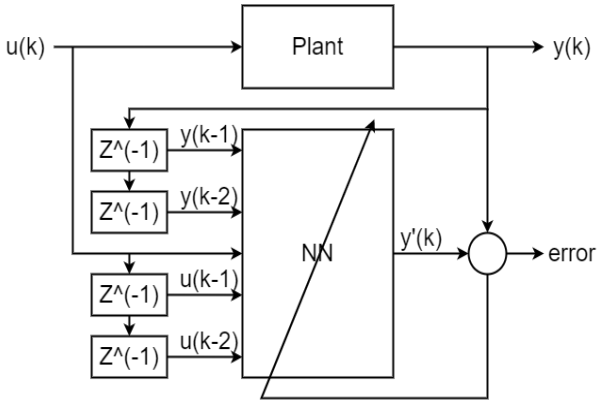


Gambar 3.1 Blok Diagram dari Sistem

Pada percobaan yang dilakukan, nilai input yang dimasukkan ke dalam *plant* ini merupakan data random dengan nilai yang berkisar di antara -1 hingga +1.

Percobaan akan memiliki dua buah tahapan. Pada prosesnya, akan dibandingkan nilai keluaran dari *plant* dengan nilai keluaran dari sistem kendali dengan menggunakan model ANN metode *back propagation*.

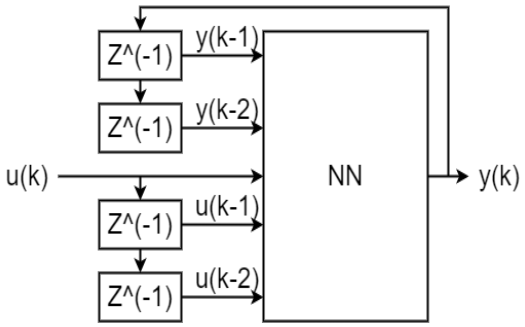
Pada tahap pertama, nilai u(k) dan y(k) yang dikeluarkan oleh *plant* akan diberikan kepada model ANN sebagai input. Adapun nilai u(k) dan y(k) ini akan digunakan untuk men-*training* model ANN dengan menggunakan metode *backpropagation* sehingga akan diperoleh nilai bobot terbaik dari model. Adapun di tengah-tengah proses *training* ini akan dilakukan proses validasi untuk menghindari terjadinya *overfitting*. Model ANN dengan nilai bobot terbaik ini pada nantinya akan digunakan untuk menghasilkan nilai y'(k). Nilai y'(k) akan dibandingkan dengan nilai y(k) keluaran *plant* untuk mengetahui tingkat keakuratan dari model ANN. Adapun cara membandingkan kedua nilai tersebut, yakni dengan menggunakan perhitungan nilai *Mean Squared Error* (MSE).



Gambar 3.2 Tahap 1 Sistem Kendali dengan ANN

Pada tahap kedua, akan digunakan nilai bobot terbaik dari tahap 1 yang memberikan hasil prediksi yang akurat dan nilai MSE yang kecil. Nilai bobot terbaik dari tahap 1 ini akan dijadikan bobot inisial di tahap 2. Pada tahap 2, nilai input yang digunakan akan sama seperti tahap 1. Akan tetapi, untuk input y(k) yang digunakan bukanlah dari *plant* seperti yang dilakukan pada tahap 1. Nilai input y(k) pada tahap 2 akan didapatkan melalui keluaran/hasil prediksi dari model ANN. Ini merupakan prinsip sistem kendali loop tertutup. Dengan proses ini, pada nantinya akan dibandingkan hasil prediksi dari model ANN, dengan input *feedback* dari output, dengan keluaran dari *plant* itu sendiri. Semakin mirip nilai keduanya, maka semakin bagus performa dari model ANN yang dibangun. Adapun nilai *Mean Squared Error* (MSE) dalam hal ini akan digunakan sebagai acuan indeks performa model.

Nilai bobot terbaik yang dapat memberikan prediksi yang akurat dan nilai MSE yang kecil selanjutnya akan digunakan pada tahap 2 untuk mendapatkan nilai y(k). Diinginkan dengan menggunakan bobot terbaik tersebut, akan diperoleh nilai y(k) yang akurat (mirip dengan nilai y(k) keluaran dari *plant*) dengan nilai MSE yang kecil.



Gambar 3.3 Tahap 2 Sistem Kendali dengan ANN

3.2 Generate Dataset

Generate dataset pada percobaan ini dapat dilakukan dengan memberikan nilai input acak dari rentang -1 sampai dengan 1 pada *plant* yang ada. Pada nantinya, akan diperoleh dataset (berupa *feature* dan *target*) yang terdiri dari 12 ribu sampel data yang akan digunakan untuk melakukan proses *training*, validasi, dan *testing* terhadap model ANN.

```
%% filename : generateDataset.m

%% PLANT
%                                1
% y(k) = ---------------------------------------------
```

```
%         (1 + y(k-1)^2)) - (0.25 * u(k)) - (0.3 * u(k-1))

%% Generate Dataset

% Data Random
y = zeros(12000, 1);
u = (1 + 1) * rand(12000, 1) - 1;

[u_row, u_column] = size(u);
[y_row, y_column] = size(y);

for k =1:y_row
    if k == 1
        y(k) = (1 / (1 + 0)) - (0.25 * u(k)) - (0.3 * 0);
    else
        y(k) = (1 / (1 + y(k-1)^2)) - (0.25 * u(k)) - (0.3 * u(k-1));
    end
end

% INPUT
uk = zeros(u_row, 1);
uk_min1 = zeros(u_row, 1);
uk_min2 = zeros(u_row, 1);
yk = zeros(y_row, 1);
yk_min1 = zeros(y_row, 1);
yk_min2 = zeros(y_row, 1);


for i = 1:u_row
    uk(i) = u(i);
    yk(i) = y(i);
end

for i = 1:(u_row - 1)
    uk_min1(i + 1) = u(i);
    yk_min1(i + 1) = y(i);
end

for i = 1:(u_row - 2)
    uk_min2(i + 2) = u(i);
    yk_min2(i + 2) = y(i);
end

dataset = cat(2, uk, uk_min1, uk_min2, yk_min1, yk_min2, yk);

dataTable = array2table(dataset);
dataTable = renamevars(dataTable, ["dataset1", "dataset2", "dataset3",
"dataset4", "dataset5", "dataset6"], ["u(k)", "u(k-1)", "u(k-2)", "y(k-1)",
"y(k-2)", "y(k)/output"]);

writetable(dataTable,'dataset.xlsx','Sheet',1);
```

Berikut adalah cuplikan dataset yang berhasil di-*generate*.

1	u(k)	u(k-1)	u(k-2)	y(k-1)	y(k-2)	y(k)/output
2	-0.6662	0	0	0	0	1.1666
3	-0.0672	-0.6662	0	1.1666	0	0.6402
4	-0.0873	-0.0672	-0.6662	0.6402	1.1666	0.7512
5	-0.7444	-0.0873	-0.0672	0.7512	0.6402	0.8515
6	0.2799	-0.7444	-0.0873	0.8515	0.7512	0.7330
7	0.7930	0.2799	-0.7444	0.7330	0.8515	0.3683
8	-0.1451	0.7930	0.2799	0.3683	0.7330	0.6790
9	-0.6483	-0.1451	0.7930	0.6790	0.3683	0.8901
10	-0.4102	-0.6483	-0.1451	0.8901	0.6790	0.8550
11	-0.8825	-0.4102	-0.6483	0.8550	0.8901	0.9214
12	-0.8797	-0.8825	-0.4102	0.9214	0.8550	1.0255
13	-0.9467	-0.8797	-0.8825	1.0255	0.9214	0.9880
14	0.1009	-0.9467	-0.8797	0.9880	1.0255	0.7648
15	0.1261	0.1009	-0.9467	0.7648	0.9880	0.5691

Gambar 3.4 Cuplikan Dataset

3.3 Tahap 1

3.3.1 Inisialisasi Data

Pertama-tama, perlu dilakukan pegimporan dataset yang telah di-*generate* pada tahap sebelumnya. Melalui sampel data yang ada di dalam dataset yang telah di-*generate* pada tahap sebelumnya, akan dilakukan proses *training*, validasi, dan *testing* terhadap model ANN. Model ANN yang telah dilatih diharapkan akan memiliki kemampuan untuk mengestimasi nilai keluaran dari *plant* berdasarkan informasi data masukan yang diberikan. Berikut adalah program untuk inisialisasi dataset yang akan digunakan dalam melatih model ANN yang dibangun dengan menggunakan algoritma *backpropagation*.

```
%% filename : tahap1.m

% Impor data
dataTable = readtable('dataset.xlsx', 'sheet', 'Sheet1');
```

3.3.2 Pemisahan Data Feature dan Target

Proses ini dilakukan dengan menentukan *feature* dan *target* dari dataset yang telah diimpor. *Feature* dari dataset meliputi kolom u(k), u(k-1), u(k-2), y(k-1), dan y(k-2). Adapun *target* meliputi data y(k).

```
% Penentuan feature dan target dari dataTable
feature = dataTable(:, 1:end-1);
target = dataTable(:, end);
```

3.3.3 Normalisasi

Proses selanjutnya, data *feature* dan *target* akan dinormalisasi dengan mengubah rentang nilai dalam dataset menjadi dari -1 hingga 1. Normalisasi ini sangat penting untuk dilakukan untuk meningkatkan kinerja dari jaringan dalam hal percepatan konvergensi ataupun peningkatan kemampuan dari model untuk menangkap pola dalam data.

```
% Normalisasi data feature
```

```
feature = normalize(feature, 'range', [-1, 1]);
target = normalize(target, 'range', [-1 1]);

% Mencari tahu jumlah baris dan kolom dari feature
[feature_num_row, feature_num_column] = size(feature);

% Mencari tahu jumlah baris dan kolom dari target
[target_num_row, target_num_column] = size(target);
```

3.3.4 Pemisahan Data Training, Validasi, dan Testing

Proses ini dilakukan dengan memisahkan data menjadi tiga bagian, yaitu data *training*, data validasi, dan data *testing*. Dalam hal ini, 5000 data pertama akan digunakan untuk data *training*, 2000 data selanjutnya akan digunakan untuk data validasi, dan 5000 data terakhir akan digunakan untuk data *testing*.

```
% Pemisahan Data Training, Data Validasi, dan Data Testing
X_train = table2array(feature(1:5000, :));
y_train = table2array(target(1:5000, :));

X_val = table2array(feature(5000 + 1:7000, :));
y_val = table2array(target(5000 + 1:7000, :));

X_test = table2array(feature(7000 + 1:end, :));
y_test = table2array(target(7000 + 1:end, :));
```

3.3.5 Inisialisasi Model BPNN

Ini merupakan tahap inisialisasi variabel yang menggambarkan jumlah *input* neuron, *hidden* neuron, dan *output* neuron. Pada model yang dibuat, dilakukan inisialisasi variabel untuk 10 buah *hidden neuron* dengan lima buah input *neuron* dan satu buah neuron *output*.

```
% Model BPNN
% Inisialisasi jumlah input layer, hidden layer, dan output
layer
n = feature_num_column;              % input neuron
p = 10;                              % hidden neuron
m = target_num_column;               % output neuron
```

3.3.6 Penentuan Nilai Bobot dan Bias Menggunakan Metode Nguyen Widrow

Proses selanjutnya yaitu menentukan nilai bobot yang ada di antara *input layer* dan *hidden layer* dan bobot yang ada di antara *hidden layer* dan *output layer*. Adapun penentuan nilai bobot dan bias ini akan dilakukan dengan menggunakan metode Nguyen Widrow yang bertujuan untuk meredam perbedaan yang terlalu jauh dalam besaran bobot dan bias antar lapisan jaringan yang dimana ini dapat mencegah masalah gradien yang meledak atau menciut selama pelatihan.

```
% Penentuan nilai bobot secara random dalam skala -0.5 sampai
0.5
a = -0.5;
b = 0.5;

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

% Optimasi nilai bobot menggunakan metode Nguyen-Widrow
```

```
beta_V = 0.7 * (p) .^ (1/n);     % Nilai beta V
beta_W = 0.7 * (m) .^ (1/p);     % Nilai beta W

V_0j = -beta_V + (beta_V - (-beta_V)) .* rand(1,p);  %
Inisialisasi nilai bobot V0j
W_0k = -beta_W + (beta_W - (-beta_W) .* rand(1,m));  %
Inisialisasi nilai bobot W0k

V_j = sqrt(sum(V.^2));                               %
Inisialisasi nilai bobot V_j
W_k = sqrt(sum(W.^2));                               %
Inisialisasi nilai bobot W_k

Vij_new = (beta_V .* V) ./ V_j;                      %
Inisialisasi nilai bobot Vij_new
Wjk_new = (beta_W .* W) ./ W_k;                      %
Inisialisasi nilai bobot Wjk_new

V = Vij_new;                                    % Bobot Nguyen-
Widrow
W = Wjk_new;                                    % Bobot Nguyen-
Widrow

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

3.3.7 Penentuan Parameter Iterasi

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

```
% Algoritma BPNN
% Penentuan nilai parameter iterasi
iterasi = 1000;              % JUMLAH EPOCH
iter = 0;
iter_val = 0;
Ep_stop = 1;
alpha = 0.01;                % LAJU PEMBELAJARAN
miu = 0.5;                   % LAJU PEMBELAJARAN MOMENTUM
```

3.3.8 Training

Algoritma ini dilakukan untuk men-*training* model dengan melakukan proses iterasi pada tiap *epoch* dan data *training*.

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 *bipolar sigmoid*.

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

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*.

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 ==============
        % Menghitung semua sinyal input dengan bobotnya
        z_inj = V_0j + X_train(a,☺) * V;

        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for j = 1:p
            zj(1,j) = -1 + 2 / (1 + exp(-z_inj(1, j)));
        end

        % Menghitung semua sinyal input dengan bobotnya
        y_ink = W_0k + zj * W;
        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for r = 1:m
            y_k(a,r) = -1 + 2 / (1 + exp(-y_ink(1, r)));
        end

        % Menghitung nilai error
        E(1, a) = abs(y_train(a,☺ - y_k(a));

        % Menghitung nilai total error kuadratik (MSE) data
validasi
        E_mse(1, a) = (y_train(a, ☺ - y_k(a)).^2;

        % ============== PROSES BACKPROPAGATION ==============
        % Menghitung informasi error
        do_k = (y_train(a,☺ - y_k(a)) .* ((1 + y_k(a)) * (1 -
y_k(a)) / 2);
        % Menghitung besarnya koreksi bobot unit output
        w_jk = alpha * zj' * do_k + miu * W_jk_lama;
        % Menghitung besarnya koreksi bias output
        w_0k = alpha * do_k + miu * W_0k_lama;
```

```
        W_jk_lama = w_jk;
        W_0k_lama = w_0k;

        % Menghitung semua koreksi error
        do_inj   = do_k * W';
        % Menghitung nilai aktivasi koreksi error
        do_j     = do_inj .* ((1 + zj) .* (1 - zj) / 2);
        % Menghtiung koreksi bobot unit hidden
        v_ij     = alpha * X_train(a,:)' * do_j + miu *
V_ij_lama;
        % Menghitung koreksi error bias unit hidden
        v_0j     = alpha * do_j + miu * V_0j_lama;

        V_ij_lama = v_ij;
        V_0j_lama = v_0j;

        % Menng-update bobot dan bias untuk setiap unit output
        W = W + w_jk;
        W_0k = W_0k + w_0k;

        % Mengupdate bobot dan bias untuk setiap unit hidden
        V = V + v_ij;
        V_0j = V_0j + v_0j;
    end

MSE_train_total = sum(MSE_train) / length(MSE_train);
MSE_val_total = sum(MSE_val) / length(MSE_val);
```

3.3.9 Validasi

Proses validasi dilakukan selama proses *training* berlangsung, lebih tepatnya setiap proses *training* sudah dilakukan sebanyak 50 *epoch*. Algoritmanya hanya meliputi proses *feedforward* untuk menghitung nilai *mean squared error*.

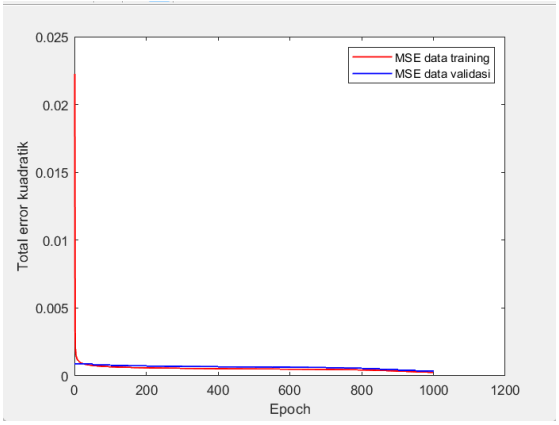
```
% Melakukan validasi data dengan bobot yang diperoleh dari
training
    % setiap 10 epoch
    if mod(iter, 50) == 0
        while Ep_stop == 1 && iter_val ≤ iter
            iter_val = iter_val + 1;
            for a = 1:length(X_val)
                % ============ PROSES FEEDFORWARD ============
                % Menghitung semua sinyal input dengan bobotnya
                z_inj_val = V_0j + X_val(a,☺) * V;

                % Proses aktivasi menggunakan fungsi bipolar
sigmoid
                for j = 1:p
                    zj_val(1,j) = -1 + 2 / (1 + exp(-
z_inj_val(1, j)));
                end

                % Menghitung semua sinyal input dengan bobotnya
                y_ink_val = W_0k + zj_val * W;
```

```
                % Proses aktivasi menggunakan fungsi bipolar
sigmoid
                for r = 1:m
                    y_k_val(a,r) = -1 + 2 / (1 + exp(-
y_ink_val(1, r)));
                end

                % Menghitung nilai error tiap epoch
                E_val(1, a) = abs(y_val(a,☺ - y_k_val(a));

                % Menghitung nilai total error kuadratik (MSE)
                % data validasi
                E_mse_val(1, a) = (y_val(a, ☺ - y_k_val(a)).^2;
            end
            % Menghitung nilai error validasi pada tiap epoch
            Ep_val(1, iter) = sum(E_val) / length(X_val);

            % Menghitung nilai total error kuadratik (MSE) pada
tiap epoch
            MSE_val(iter_val, 1) = sum(E_mse_val) /
length(X_val);
        end
    end

    % Ini masuk ke dalam while loop proses training karena
proses      training dan validasi dilakukan secara bebarengan
    % Menghitung nilai error training pada tiap epoch
    Ep(1, iter) = sum€ / length(X_train);

    % Menghitung nilai total error kuadratik (MSE) pada tiap
epoch
    MSE_train(iter, 1) = sum(E_mse) / length(X_train);

    acc_p(iter, 1) = 1 - MSE_train(iter, 1);
end
```

3.3.10 Generate File Excel dengan Bobot Terbaik dari Tahap 1

Proses ini dilakukan untuk meng-*generate* file *excel* yang berisi bobot terbaik dari tahap 1. Pada nantinya, bobot terbaik dari tahap 1 ini akan digunakan sebagai bobot awal pada tahap 2.

Bobot terbaik dapat diambil sebelum grafik MSE dari proses validasi meningkat. Melalui proses simulasi yang dilakukan, diperoleh grafik MSE dari data *training* dan validasi, yakni sebagai berikut.



Gambar 3.5 Grafik MSE dari Proses *Training* dan Validasi

Melalui grafik tersebut dapat diketahui jika nilai MSE dari proses validasi tidak meningkat sampai epoch 1000. Oleh karena itu, dapat diketahui jika bobot dan bias terbaik dari proses *training* di tahap 1 merupakan bobot dan bias di akhir proses *training*.

```
writematrix(V, 'bobot_V.xlsx','Sheet', 1);
writematrix(W, 'bobot_W.xlsx','Sheet', 1);
writematrix(V_0j, 'bias_V.xlsx','Sheet', 1);
writematrix(W_0k, 'bias_W.xlsx','Sheet', 1);
```

Berikut ini data nilai bobot dan bias terbaik yang diperoleh melalui proses *training* di tahap 1.

	VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	-0.7816	1.3562	0.4707	0.0843	0.6007	-0.0651	0.4303	-1.6423	-0.0694	-0.0619
2	0.5188	0.8784	0.3139	-0.4493	-1.1401	0.3512	0.4440	-0.4728	0.2829	0.5839
3	-0.0135	-0.0220	0.0749	-0.2149	-0.0850	-0.1803	0.4415	0.7470	0.5885	0.0504
4	-1.0797	0.3309	1.0650	-0.8367	-0.1514	-0.1705	0.2630	0.5555	0.4775	-0.3244
5	0.1922	0.6651	0.1764	0.2059	-0.4150	-0.5194	-0.3472	0.1791	0.6827	-0.2517

Gambar 3.6 Bobot V

	VarName1
	Number
1	0.9923
2	-1.4592
3	-1.4084
4	1.5664
5	0.9271
6	-0.7934
7	-0.0211
8	1.4400
9	-0.0669
10	-0.1451

Gambar 3.7 Bobot W

	VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	-0.0357	1.8803	0.9462	0.5315	0.2946	0.6740	-0.6806	2.3571	0.5638	0.6979

Gambar 3.8 Bias V

	VarName1
	Number
1	0.5338

Gambar 3.9 Bias W

3.3.11 Testing

Algoritma ini dilakukan untuk melakukan proses *testing* pada model yang telah dibangun dengan melakukan proses *feedforward* pada setiap sampel data *testing*. Dengan dilakukannya proses *testing* ini, akan diketahui nilai *mean squared error* antara data sebenarnya (keluaran *plant*) dengan data prediksi keluaran model ANN.

```
% Melakukan testing
E_test = zeros(length(X_test), 1);
for a = 1:length(X_test)
    % ==================== PROSES FEEDFORWARD
====================
    z_inj_test = X_test(a,:) * V + V_0j;
    % Proses aktivasi menggunakan sigmoid
    for j=1:p
        zj_test(1,j) = -1 + 2 / (1 + exp(-z_inj_test(1,j)));
%Aktivasi sigmoid
    end

    y_ink_test = zj_test * W + W_0k;

    for k=1:m
        y_k_test(a,k) = -1 + 2 / (1 + exp(-y_ink_test(1,k)));
%Aktivasi sigmoid
    end

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

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

    % MSE
    E_mse_test(a, 1) = (y_test(a, :) - y_k_test(a)).^2;

end

Ep_test = sum(E_test) / length(X_test);

MSE_test = sum(E_mse_test) / length(X_test);
```

3.3.12 Evaluasi

Pada tahap ini, akan ditampilkan grafik *mean squared error* pada proses *data training* dan validasi, grafik error yang didapat selama proses data *training*, visualisasi *scatter* antara data keluaran *plant* dengan data prediksi keluaran model ANN selama proses *training*, validasi, dan *testing*.

```
figure;
plot(MSE_train, 'r-', 'Linewidth', 1);
hold on
plot(MSE_val, 'b-', 'LineWidth', 1);
hold off
ylabel('Total error kuadratik'); xlabel('Epoch');
legend("MSE data training", "MSE data validasi");
```

```
figure;
plot(Ep, 'r-', 'Linewidth', 1);
ylabel('Total Error'); xlabel('Epoch');
legend("Error data training");

x = 1:5000;
y = 1:2000;
z = 1:5000;

figure;
scatter(x, y_train);
hold on
scatter(x, y_k);
legend("Output Plant Training", "Output ANN Training");
hold off

figure;
scatter(y, y_val);
hold on
scatter(y, y_k_val);
legend("Output Plant Validasi", "Output ANN Validasi");
hold off

figure;
scatter(z, y_test);
hold on
scatter(z, y_k_test);
legend("Output Plant Testing", "Output ANN Testing");
hold off

disp("Error final Data Training = " + (Ep(end)*100) + "%");
disp("Error final Data Validasi = " + (Ep_val(end)*100) + "%");
disp("Error final Data Testing = " + (Ep_test*100) + "%");

disp("MSE final Data Training = " + (MSE_train(end)*100) +
"%");
disp("MSE final Data Validasi = " + (MSE_val(end)*100) + "%");

disp("MSE total Data Training = " + (MSE_train_total*100) +
"%");
disp("MSE total Data Validasi = " + (MSE_val_total*100) + "%");
disp("MSE totak Data Testing = " + (MSE_test*100) + "%");
```

3.4 Tahap 2

3.4.1 Inisialisasi Data

Pertama-tama, perlu dilakukan pegimporan dataset yang telah di-*generate* pada tahap sebelumnya. Melalui sampel data yang ada di dalam dataset yang telah di-*generate* pada tahap sebelumnya, akan dilakukan proses *training*, validasi, dan *testing* terhadap model ANN. Model ANN yang telah dilatih diharapkan akan memiliki kemampuan untuk mengestimasi nilai keluaran dari *plant* berdasarkan

informasi data masukan yang diberikan. Berikut adalah program untuk inisialisasi dataset yang akan digunakan dalam melatih model ANN yang dibangun dengan menggunakan algoritma *backpropagation*.

```
%% filename : tahap2.m

% Impor data
dataTable = readtable('dataset.xlsx', 'sheet', 'Sheet1');
```

3.4.2 Pemisahan Data Feature dan Target

Proses ini dilakukan dengan menentukan *feature* dan *target* dari dataset yang telah diimpor. *Feature* dari dataset meliputi kolom u(k), u(k-1), u(k-2), y(k-1), dan y(k-2). Adapun *target* meliputi data y(k).

```
% Penentuan feature dan target dari dataTable
feature = dataTable(:, 1:end-1);
target = dataTable(:, end);
```

3.4.3 Normalisasi

Proses selanjutnya, data *feature* dan *target* akan dinormalisasi dengan mengubah rentang nilai dalam dataset menjadi dari -1 hingga 1. Normalisasi ini sangat penting untuk dilakukan untuk meningkatkan kinerja dari jaringan dalam hal percepatan konvergensi ataupun peningkatan kemampuan dari model untuk menangkap pola dalam data.

```
% Normalisasi data feature
feature = normalize(feature, 'range', [-1, 1]);
target = normalize(target, 'range', [-1 1]);

% Mencari tahu jumlah baris dan kolom dari feature
[feature_num_row, feature_num_column] = size(feature);

% Mencari tahu jumlah baris dan kolom dari target
[target_num_row, target_num_column] = size(target);
```

3.4.4 Pemisahan Data Training, Validasi, dan Testing

Proses ini dilakukan dengan memisahkan data menjadi tiga bagian, yaitu data *training*, data validasi, dan data *testing*. Dalam hal ini, 5000 data pertama akan digunakan untuk data *training*, 2000 data selanjutnya akan digunakan untuk data validasi, dan 5000 data terakhir akan digunakan untuk data *testing*.

```
% Pemisahan Data Training, Data Validasi, dan Data Testing
X_train = table2array(feature(1:5000, :));
y_train = table2array(target(1:5000, :));
vektor_y_plant_train = y_train;

X_val = table2array(feature(5000 + 1:7000, :));
y_val = table2array(target(5000 + 1:7000, :));
vektor_y_plant_val = y_val;

X_test = table2array(feature(7000 + 1:end, :));
y_test = table2array(target(7000 + 1:end, :));
vektor_y_plant_test = y_test;
```

3.4.5 Inisialisasi Model BPNN

Ini merupakan tahap inisialisasi variabel yang menggambarkan jumlah *input* neuron, *hidden* neuron, dan *output* neuron. Pada model yang dibuat, dilakukan inisialisasi variabel untuk 10 buah *hidden neuron* dengan lima buah input *neuron* dan satu buah neuron *output*.

```
% Model BPNN
% Inisialisasi jumlah input layer, hidden layer, dan output layer
n = feature_num_column;          % input layer
p = 10;                          % hidden layer
m = target_num_column;           % output layer
```

3.4.6 Impor Nilai Bobot dari Tahap 1

Tahap ini dilakukan dengan mengimpor bobot dan bias terbaik yang diperoleh melalui tahap 1 sebagai bobot dan bias awal pada tahap 2.

```
% Impor bobot dan bias yang sudah dilatih dari tahap 1
V = readmatrix("bobot_V.xlsx");
W = readmatrix("bobot_W.xlsx");
V_0j = readmatrix("bias_V.xlsx");
W_0k = readmatrix("bias_V.xlsx");

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

3.4.7 Penentuan Parameter Iterasi

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

```
% Algoritma BPNN
% Penentuan nilai parameter iterasi
iterasi = 10000;
iter = 0;
iter_val = 0;
Ep_stop = 1;
alpha = 0.01;
miu = 0.1;
```

3.4.8 Training

Algoritma ini dilakukan untuk men-*training* model dengan melakukan proses iterasi pada tiap *epoch* dan data *training*. Adapun perbedaan *training* tahap 1 dan 2 yaitu, hasil estimasi dari model ANN akan di-*feedback* sebagai masukan pada iterasi berikutnya di tahap 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 *bipolar sigmoid*.

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

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*.

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 = 3:length(X_train)
        % ============== PROSES FEEDFORWARD ==============
        % Menghitung semua sinyal input dengan bobotnya
        feature_train = [X_train(a, 1) X_train(a, 2) X_train(a, 3)
X_train(a, 4) X_train(a, 5)];
        z_inj = V_0j + feature_train * V;

        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for j = 1:p
            zj(1,j) = -1 + 2 / (1 + exp(-z_inj(1, j)));
        end

        % Menghitung semua sinyal input dengan bobotnya
        y_ink = W_0k + zj * W;
        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for r = 1:m
            y_k(a,r) = -1 + 2 / (1 + exp(-y_ink(1, r)));
        end

        for s = 1:m
            train_pred(a, s) = y_k(a);
        end

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

        % Menghitung nilai total error kuadratik (MSE) data validasi
        E_mse(1, a) = (y_train(a, :) - y_k(a)).^2;


        if a < length(X_train)
            X_train(a + 1, 4) = train_pred(a, 1);
        end
```

```
        if a < length(X_train)
            X_train(a + 1, 5) = train_pred(a - 1, 1);
        end

        % ============== PROSES BACKPROPAGATION ==============
        % Menghitung informasi error
        do_k = (y_train(a,:) - y_k(a)) .* ((1 + y_k(a)) * (1 -
y_k(a)) / 2);
        % Menghitung besarnya koreksi bobot unit output
        w_jk = alpha * zj' * do_k + miu * W_jk_lama;
        % Menghitung besarnya koreksi bias output
        w_0k = alpha * do_k + miu * W_0k_lama;

        W_jk_lama = w_jk;
        W_0k_lama = w_0k;

        % Menghitung semua koreksi error
        do_inj  = do_k * W';
        % Menghitung nilai aktivasi koreksi error
        do_j    = do_inj .* ((1 + zj) .* (1 - zj) / 2);
        % Menghtiung koreksi bobot unit hidden
        v_ij    = alpha * X_train(a,:)' * do_j + miu * V_ij_lama;
        % Menghitung koreksi error bias unit hidden
        v_0j    = alpha * do_j + miu * V_0j_lama;

        V_ij_lama = v_ij;
        V_0j_lama = v_0j;

        % Menng-update bobot dan bias untuk setiap unit output
        W = W + w_jk;
        W_0k = W_0k + w_0k;

        % Mengupdate bobot dan bias untuk setiap unit hidden
        V = V + v_ij;
        V_0j = V_0j + v_0j;
    end
```

3.4.9 Validasi

Proses validasi dilakukan selama proses *training* berlangsung, lebih tepatnya setiap proses *training* sudah dilakukan sebanyak 50 *epoch*. Algoritmanya hanya meliputi proses *feedforward* untuk menghitung nilai *mean squared error*. Adapun perbedaan validasi tahap 1 dan 2 yaitu, hasil estimasi dari proses *feedforward* akan di-*feedback* sebagai masukan pada iterasi berikutnya di tahap 2.

```
% Melakukan validasi data dengan bobot yang diperoleh dari training
    % setiap 10 epoch
    if mod(iter, 50) == 0
        while Ep_stop == 1 && iter_val <= iter
            iter_val = iter_val + 1;
            for a = 3:length(X_val)
                % ============== PROSES FEEDFORWARD ==============
                % Menghitung semua sinyal input dengan bobotnya
```

```
                feature_val = [X_val(a, 1) X_val(a, 2) X_val(a, 3)
X_val(a, 4) X_val(a, 5)];
                z_inj_val = V_0j + feature_val * V;

                % Proses aktivasi menggunakan fungsi bipolar sigmoid
                for j = 1:p
                    zj_val(1,j) = -1 + 2 / (1 + exp(-z_inj_val(1,
j)));
                end

                % Menghitung semua sinyal input dengan bobotnya
                y_ink_val = W_0k + zj_val * W;
                % Proses aktivasi menggunakan fungsi bipolar sigmoid
                for r = 1:m
                    y_k_val(a,r) = -1 + 2 / (1 + exp(-y_ink_val(1,
r)));
                end

                for s = 1:m
                    val_pred(a, s) = y_k_val(a);
                end

                % Menghitung nilai error
                E_val(1, a) = abs(y_val(a,:) - y_k_val(a));

                % Menghitung nilai total error kuadratik (MSE) data
validasi
                E_mse_val(1, a) = (y_val(a, :) - y_k_val(a)).^2;

                if a < length(X_val)
                    X_val(a + 1, 4) = val_pred(a, 1);
                end

                if a < length(X_val)
                    X_val(a + 1, 5) = val_pred(a - 1, 1);
                end
            end
            % Menghitung nilai error validasi pada tiap epoch
            Ep_val(1, iter) = sum(E_val) / length(X_val);

            % Menghitung nilai total error kuadratik (MSE) pada tiap
epoch
            MSE_val(iter_val, 1) = sum(E_mse_val) / length(X_val);
        end
    end

    % Ini masuk ke dalam while loop proses training karena proses
      training dan validasi dilakukan secara bebarengan

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

    % Menghitung nilai total error kuadratik (MSE) pada tiap epoch
```

```
    MSE_train(iter, 1) = sum(E_mse) / length(X_train);

    acc_p(iter, 1) = 1 - MSE_train(iter, 1);
end

MSE_train_total = sum(MSE_train) / length(MSE_train);
MSE_val_total = sum(MSE_val) / length(MSE_val);
```

3.4.10 Testing

Algoritma ini dilakukan untuk melakukan proses *testing* pada model yang telah dibangun dengan melakukan proses *feedforward* pada setiap sampel data *testing*. Dengan dilakukannya proses *testing* ini, akan diketahui nilai *mean squared error* antara data sebenarnya (keluaran *plant*) dengan data prediksi keluaran model ANN. Adapun perbedaan *testing* tahap 1 dan 2 yaitu, hasil estimasi dari model ANN akan di-*feedback* sebagai masukan pada iterasi berikutnya di tahap 2.

```
% Melakukan testing
E_test = zeros(length(X_test), 1);
for a = 3:length(X_test)
    %       ====================       PROSES       FEEDFORWARD
====================
    % Menghitung semua sinyal input dengan bobotnya
    feature_test  =  [X_test(a,  1)  X_test(a,  2)  X_test(a,  3)
X_test(a, 4) X_test(a, 5)];
    z_inj_test = V_0j + feature_test * V;

    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for j = 1:p
        zj_test(1,j) = -1 + 2 / (1 + exp(-z_inj_test(1, j)));
    end

    % Menghitung semua sinyal input dengan bobotnya
    y_ink_test = W_0k + zj_test * W;
    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for r = 1:m
        y_k_test(a,r) = -1 + 2 / (1 + exp(-y_ink_test(1, r)));
    end

    for s = 1:m
        test_pred(a, s) = y_k_test(a);
    end

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

    % Menghitung nilai total error kuadratik (MSE) data validasi
    E_mse_test(a, 1) = (y_test(a, :) - y_k_test(a)).^2;

    if a < length(X_test)
        X_test(a + 1, 4) = test_pred(a, 1);
    end

    if a < length(X_test)
```

```
        X_test(a + 1, 5) = test_pred(a - 1, 1);
    end
end

Ep_test = sum(E_test) / length(X_test);

MSE_test = sum(E_mse_test) / length(X_test);
```

3.4.11 Evaluasi

Pada tahap ini, akan ditampilkan grafik *mean squared error* pada proses *data training* dan validasi, grafik error yang didapat selama proses data *training*, visualisasi *scatter* antara data keluaran *plant* dengan data prediksi keluaran model ANN selama proses *training*, validasi, dan *testing*.

```
figure;
plot(MSE_train, 'm-', 'Linewidth', 1);
hold on
plot(MSE_val, 'b-', 'LineWidth', 1);
hold off
ylabel('Total error kuadratik'); xlabel('Epoch');
legend("MSE data training", "MSE data validasi");


figure;
plot(Ep, 'r-', 'Linewidth', 1);
ylabel('Total Error'); xlabel('Epoch');
legend("Error data training", "Error data validasi");

x = 1:5000; y = 1:2000; z = 1:5000;

figure;
scatter(x, vektor_y_plant_train);
hold on
scatter(x, train_pred);
legend("Output Plant Training", "Output ANN Training");
hold off

figure;
scatter(y, vektor_y_plant_val);
hold on
scatter(y, val_pred);
legend("Output Plant Validasi", "Output ANN Validasi");
hold off

figure;
scatter(z, vektor_y_plant_test);
hold on
scatter(z, test_pred);
legend("Output Plant Testing", "Output ANN Testing");
hold off

disp("Error final Data Training = " + (Ep(end)*100) + "%");
disp("Error final Data Validasi = " + (Ep_val(end)*100) + "%");
disp("Error final Data Testing = " + (Ep_test*100) + "%");
```

```
disp("MSE final Data Training = " + (MSE_train(end)*100) + "%");
disp("MSE final Data Validasi = " + (MSE_val(end)*100) + "%");

disp("MSE total Data Training = " + (MSE_train_total*100) +
"%");
disp("MSE total Data Validasi = " + (MSE_val_total*100) + "%");
disp("MSE totak Data Testing = " + (MSE_test*100) + "%");
```

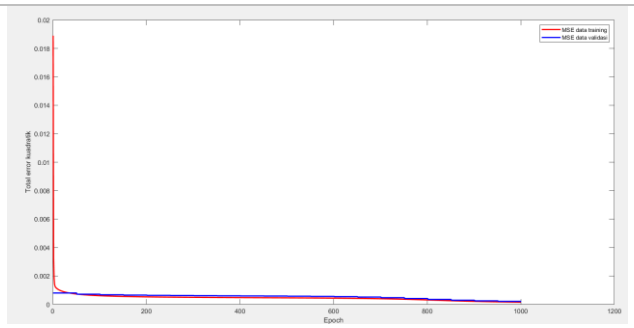
BAB IV
SIMULASI DAN ANALISIS HASIL

4.1 Hasil Pengujian Model dan Pembahasan

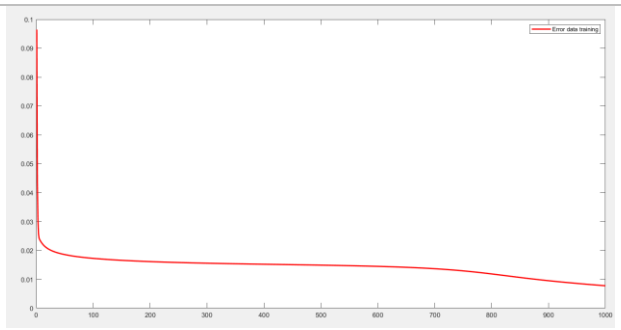
Proses pengujian model akan dilakukan pada sistem kendali ANN tahap 1 dan tahap 2. Pada proses pengujian, digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 1000
- Laju pembelajaran = 0.01
- Laju pembelajaran momentum = 0.5

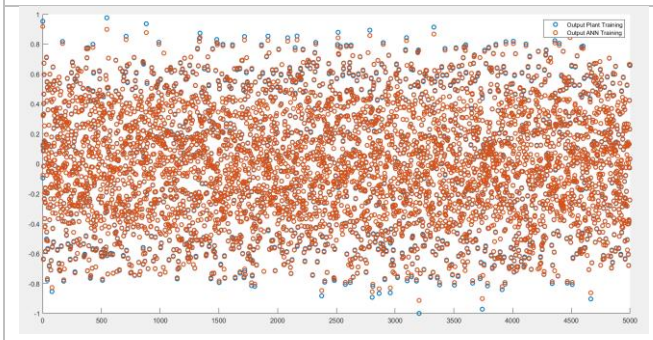
4.1.1 Tahap 1



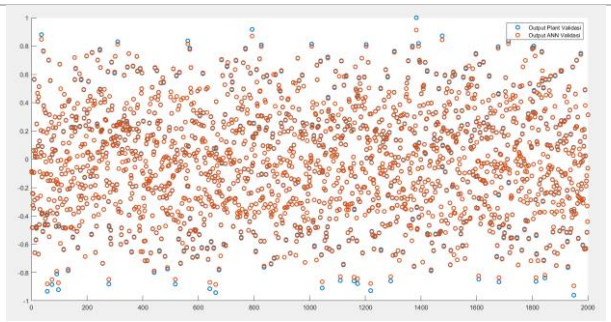
Gambar 4.1 Grafik Total Error Kuadratik Data *Training* dan Validasi



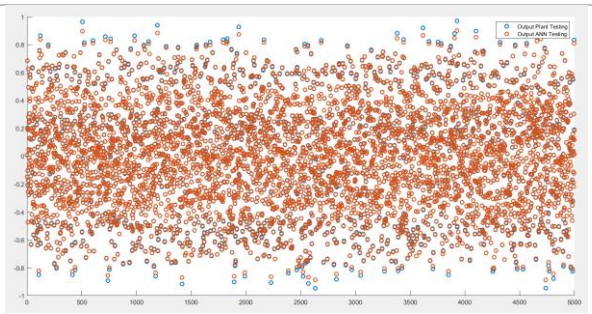
Gambar 4.2 Grafik Error Data Training



Gambar 4.3 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



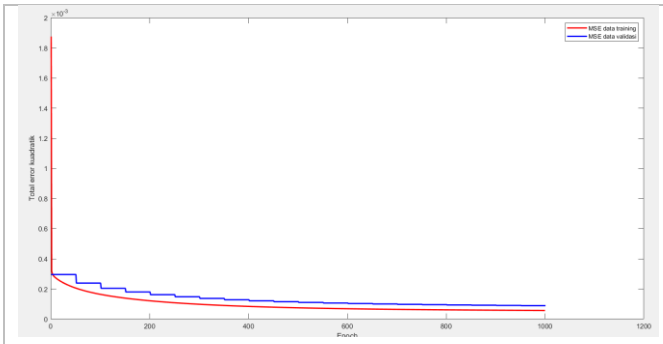
Gambar 4.4 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data Validasi)



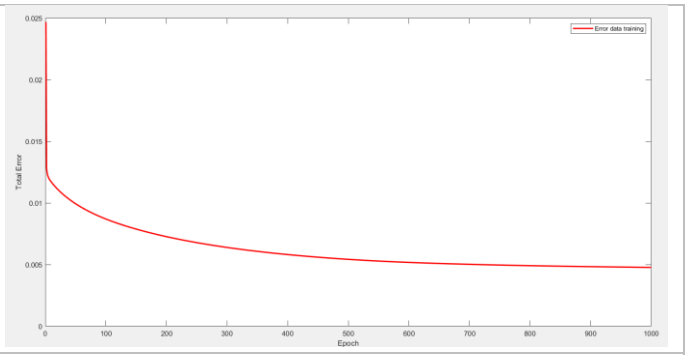
Gambar 4.5 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 1.472%
Error final Data Validasi = 1.4962%
Error final Data Testing = 1.4581%
MSE final Data Training = 0.045224%
MSE final Data Validasi = 0.058829%
MSE total Data Training = 0.054427%
MSE total Data Validasi = 0.063495%
MSE totak Data Testing = 0.051558%
```

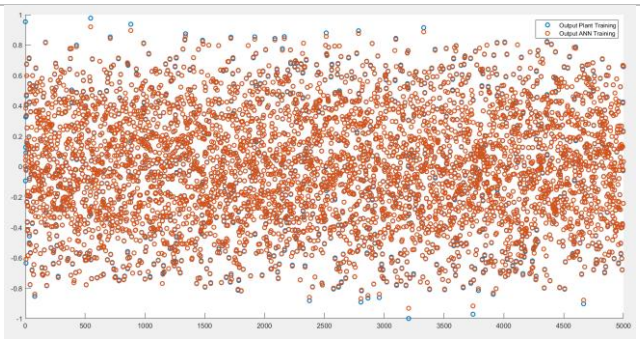
4.1.2 Tahap 2



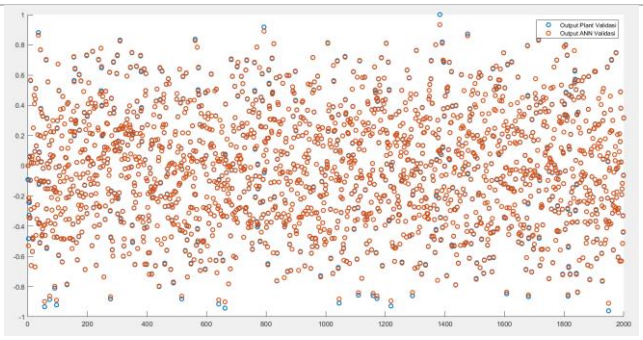
Gambar 4.6 Grafik Total Error Kuadratik Data *Training* dan Validasi



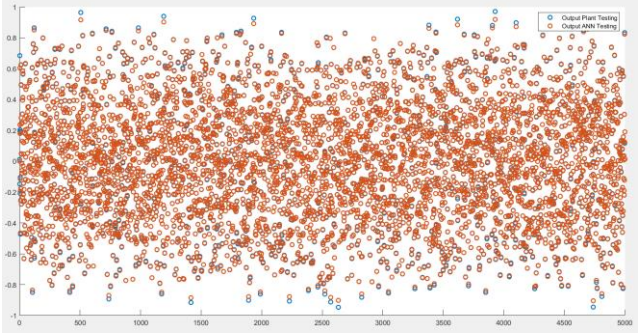
Gambar 4.7 Grafik Error Data Training



Gambar 4.8 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 4.9 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data Validasi)



Gambar 4.10 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.47662%
Error final Data Validasi = 0.53333%
Error final Data Testing = 0.49462%
MSE final Data Training = 0.0056803%
MSE final Data Validasi = 0.0089001%
MSE total Data Training = 0.0096597%
MSE total Data Validasi = 0.013547%
MSE total Data Testing = 0.0066021%
```

Dapat dilihat melalui grafik yang diperoleh dari tahap 1 dan 2 jika nilai *mean squared error* total yang diperoleh melalui proses *training*, validasi, dan *testing* sangat kecil nilainya. Hal ini mengindikasikan jika model ANN yang dibuat sudah memiliki performa cukup bagus untuk mengestimasi keluaran dari *plant* berdasarkan data masukan yang ada.

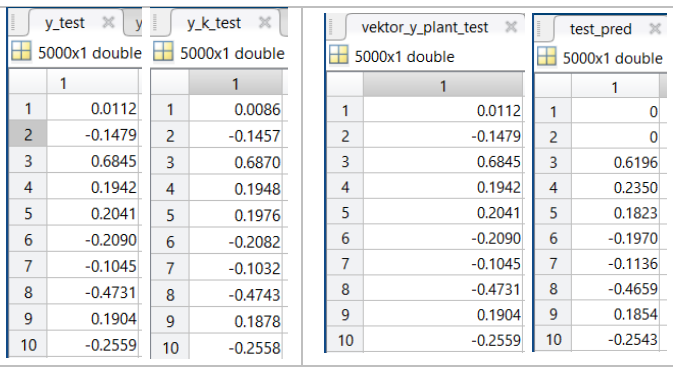
Dari grafik MSE data *training* dan validasi yang diperoleh dari tahap 1, dapat diketahui jika nilai MSE dari proses validasi tidak meningkat selama proses *training* berlangsung. Artinya, model ANN tidak mengalami *overfitting* saat melakukan estimasi terhadap nilai keluaran *plant*. Itulah sebabnya, bobot dan bias yang diperoleh setelah 1000 *epoch* akan digunakan sebagai bobot dan bias awal pada tahap 2.

Grafik MSE data *training* dan validasi yang diperoleh di tahap 2 pun nilainya cenderung menurun terus. Artinya, model ANN yang dibangun tidak mengalami *overfitting* ketika di-

training sebanyak 1000 epoch, serta model ANN juga memiliki kualitas yang bagus jika diterapkan sebagai pengendali di sistem kendali.

Jika diperhatikan lebih detail lagi pada grafik MSE data *training* dan validasi, dapat dilihat jika nilai MSE, baik itu di tahap 1 dan 2, cenderung turun drastis di proses iterasi yang bisa dibilang cenderung masih sedikit. Hal ini tentu bukan merupakan sesuatu yang kebetulan terjadi. Nilai MSE yang turun drastis di awal-awal iterasi proses *training* disebabkan karena model ANN sudah mulai "peka" terhadap pola yang ada pada data. Cepatnya kepekaan model ANN terhadap pola sampel data ini bisa terjadi karena digunakan dataset yang diperoleh melalui *plant* orde dua saja. Dalam hal ini, berarti *plant* orde dua bisa dibilang masih dianggap cukup sederhana bagi model ANN yang dibangun. *Plant* orde dua akan dianggap lebih kompleks jika dilakukan perubahan arsitektur dari model ANN, seperti pengurangan jumlah *hidden neuron*. Cepatnya proses belajar dari model ANN juga disebabkan karena dilakukannya proses normalisasi sebelum proses *training* untuk mengkoreksi nilai bobot dan bias dilakukan. Jika dibalik, model ANN dengan arsitektur yang sama seperti yang dilakukan dalam percobaan bisa menjadi lebih lambat dalam mempelajari dataset jika proses normalisasi tidak dilakukan, ataupun orde dari *plant* ditingkatkan. Peningkatan orde dari *plant* tentu akan meningkatkan lama waktu bagi model ANN untuk mengenali pola yang ada dalam dataset. Itulah sebabnya, penurunan nilai MSE akan membutuhkan lebih banyak iterasi yang perlu dilakukan.

Dalam proses algoritmanya, penurunan nilai MSE data *training* pada setiap *epoch*-nya disebabkan karena adanya proses *backpropagation* yang akan memperbaiki nilai bobot dan bias berdasarkan informasi error antara estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya. Perbandingan hasil estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya pun dapat dilihat dengan menggunakan *scatter plot*. Titik berwarna biru menggambarkan keluaran dari *plant* sedangkan yang berwarna oren menggambarkan hasil estimasi keluaran *plant* oleh model ANN. Dari hasil yang diperoleh, dapat diketahui jika rentang nilai keluaran *plant* dan model ANN berada di rentang -1 hingga 1. Adapun perbandingan antara keluaran *plant* dengan data estimasi keluaran model ANN tidaklah jauh berbeda yang dimana ini menyebabkan nilai MSE yang diperoleh dari proses simulasi sangatlah kecil nilainya. Hal ini bisa terjadi karena model ANN yang dibangun (baik itu dari segi dataset, *hyperparameter*, arsitektur, ataupun algoritma) sudah cukup bagus. Selain itu, dilakukan juga proses normalisasi terlebih dahulu sebelum proses *training* dilakukan, seperti yang sudah disinggung sebelumnya. Adapun dengan proses normalisasi ini, akan menjadi lebih mudah bagi model untuk mempelajari "pola" yang ada pada keseluruhan sampel data pada data *training*.



y_test — 5000x1 double

	1
1	0.0112
2	-0.1479
3	0.6845
4	0.1942
5	0.2041
6	-0.2090
7	-0.1045
8	-0.4731
9	0.1904
10	-0.2559

y_k_test — 5000x1 double

	1
1	0.0086
2	-0.1457
3	0.6870
4	0.1948
5	0.1976
6	-0.2082
7	-0.1032
8	-0.4743
9	0.1878
10	-0.2558

vektor_y_plant_test — 5000x1 double

	1
1	0.0112
2	-0.1479
3	0.6845
4	0.1942
5	0.2041
6	-0.2090
7	-0.1045
8	-0.4731
9	0.1904
10	-0.2559

test_pred — 5000x1 double

	1
1	0
2	0
3	0.6196
4	0.2350
5	0.1823
6	-0.1970
7	-0.1136
8	-0.4659
9	0.1854
10	-0.2543

Gambar 4.11 Cuplikan Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*) pada Tahap 1 dan 2

4.2 Pengaruh Jumlah Hidden Neuron Terhadap Mean Squared Error

- Jumlah iterasi = 1000
- Error minimum = 0.01
- Laju pembelajaran momentum = 0.5

4.2.1 Tahap 1

```
Error final Data Training = 1.472%
Error final Data Validasi = 1.4962%
Error final Data Testing = 1.4581%
MSE final Data Training = 0.045224%
MSE final Data Validasi = 0.058829%
MSE total Data Training = 0.054427%
MSE total Data Validasi = 0.063495%
MSE totak Data Testing = 0.051558%
```

Gambar 4.12 Hidden Neuron 10

```
Error final Data Training = 0.49879%
Error final Data Validasi = 0.57091%
Error final Data Testing = 0.53012%
MSE final Data Training = 0.0066778%
MSE final Data Validasi = 0.010734%
MSE total Data Training = 0.020274%
MSE total Data Validasi = 0.023143%
MSE total Data Testing = 0.0080127%
```

Gambar 4.13 Hidden Neuron 30

```
Error final Data Training = 0.45268%
Error final Data Validasi = 0.515%
Error final Data Testing = 0.48047%
MSE final Data Training = 0.0055046%
MSE final Data Validasi = 0.0093099%
MSE total Data Training = 0.016706%
MSE total Data Validasi = 0.020911%
MSE total Data Testing = 0.0067804%
```

Gambar 4.14 Hidden Neuron 50

```
Error final Data Training = 0.46955%
Error final Data Validasi = 0.52866%
Error final Data Testing = 0.50123%
MSE final Data Training = 0.0059087%
MSE final Data Validasi = 0.0096546%
MSE total Data Training = 0.014951%
MSE total Data Validasi = 0.018206%
```

Gambar 4.15 Hidden Neuron 70

4.2.1 Tahap 2

```
Error final Data Training = 0.47662%
Error final Data Validasi = 0.53333%
Error final Data Testing = 0.49462%
MSE final Data Training = 0.0056803%
MSE final Data Validasi = 0.0089001%
MSE total Data Training = 0.0096597%
MSE total Data Validasi = 0.013547%
MSE total Data Testing = 0.0066021%
```

Gambar 4.16 Hidden Neuron 10

```
Error final Data Training = 0.40756%
Error final Data Validasi = 0.46825%
Error final Data Testing = 0.4331%
MSE final Data Training = 0.0041851%
MSE final Data Validasi = 0.0071013%
MSE total Data Training = 0.0056305%
MSE total Data Validasi = 0.0086279%
MSE total Data Testing = 0.005149%
```

Gambar 4.17 Hidden Neuron 30

```
Error final Data Training = 0.33055%
Error final Data Validasi = 0.37866%
Error final Data Testing = 0.35272%
MSE final Data Training = 0.0028793%
MSE final Data Validasi = 0.0050122%
MSE total Data Training = 0.0041991%
MSE total Data Validasi = 0.0063944%
MSE total Data Testing = 0.0035443%
```

Gambar 4.18 Hidden Neuron 50

```
Error final Data Training = 0.45382%
Error final Data Validasi = 0.50732%
Error final Data Testing = 0.47298%
MSE final Data Training = 0.0053755%
MSE final Data Validasi = 0.0087964%
MSE total Data Training = 0.01432%
MSE total Data Validasi = 0.017713%
MSE total Data Testing = 0.0064766%
```

Gambar 4.19 Hidden Neuron 70

Pada percobaan ini, ingin dilihat pengaruh dari jumlah *hidden neuron* terhadap nilai MSE pada data pengujian (utamanya). Pengujian dilakukan dengan 4 variasi nilai *hidden neuron* yang berbeda, yaitu [10; 30; 50; 70]. Proses pelatihan akan dilakukan sebanyak 1000 epoch pada setiap percobaan. Dapat diketahui dari variasi nilai *hidden neuron*, diketahui jika perubahan nilai *hidden neuron* dari model akan memberikan perubahan yang cukup signifikan terhadap nilai MSE dari model ANN yang telah dibangun. Nilai MSE yang diperoleh baik itu dari tahap 1 dan tahap 2 untuk semua variasi nilai *hidden neuron* bisa dibilang masih sangat kecil nilainya (bagus). Akan tetapi, terdapat kecenderungan perubahan nilai MSE data *testing* yang diperoleh untuk peningkatan jumlah *hidden neuron*.

Pada percobaan tahap 1, nilai MSE data pengujian terkecil yaitu sebesar 0.00678% yang diperoleh dengan menggunakan 50 *hidden neuron*. Adapun pada tahap 2, nilai MSE data pengujian terkecil yaitu sebesar 0.00354% yang diperoleh dengan menggunakan 50 *hidden neuron*. Hal ini menunjukkan bahwa, untuk dataset dan *hyperparameter* yang digunakan, model ANN dengan jumlah *hidden neuron* 50 akan memberikan performa yang terbaik dengan nilai MSE paling kecil.

Jumlah *hidden neuron* yang terlalu sedikit akan membuat model ANN kurang mampu untuk memahami dan mengekstraksi pola yang ada dalam data. Ini dapat mengakibatkan nilai MSE data pengujian akan semakin tinggi karena model tidak memiliki kemampuan untuk "berubah secara signifikan" terhadap perubahan masukan yang ada (kurang adaptif) sehingga hasil prediksi akan kalah akurat (karena tidak dapat menggambarkan data pelatihan dengan optimal akibat *underfitting*) dengan model yang menggunakan *hidden neuron* yang terlalu banyak.

Di lain sisi, jumlah *hidden neuron* yang berlebihan dapat menyebabkan munculnya *noise* dalam data. Dalam hal ini, model tidak akan memiliki kemampuan untuk menggeneralisasi data masukan yang ada dengan lebih baik (*overfitting*) sehingga nilai MSE yang diperoleh pun tidak pasti menurun seiring dengan peningkatan jumlah *hidden neuron*.

Itulah sebabnya, harus digunakan jumlah *hidden neuron* yang optimal berdasarkan beberapa pertimbangan, seperti tingkat kompleksitas dataset, ataupun *hyperparameter* yang digunakan selama proses pelatihan dilakukan.

4.3 Pengaruh Laju Pembelajaran Terhadap Mean Squared Error

- Jumlah hidden neuron = 10
- Jumlah iterasi = 1000
- Laju pembelajaran momentum = 0.5

4.3.1 Tahap 1

```
Error final Data Training = 0.3385%
Error final Data Validasi = 0.35811%
Error final Data Testing = 0.33981%
MSE final Data Training = 0.003127%
MSE final Data Validasi = 0.0048998%
MSE total Data Training = 0.0083816%
MSE total Data Validasi = 0.0098801%
MSE total Data Testing = 0.0035941%
```

Gambar 4.20 Laju Pembelajaran 0.1

```
Error final Data Training = 0.15326%
Error final Data Validasi = 0.17321%
Error final Data Testing = 0.16268%
MSE final Data Training = 0.00067639%
MSE final Data Validasi = 0.0011015%
MSE total Data Training = 0.0029413%
MSE total Data Validasi = 0.0023135%
MSE total Data Testing = 0.00074992%
```

Gambar 4.21 Laju Pembelajaran 0.3

```
Error final Data Training = 0.18099%
Error final Data Validasi = 0.19242%
Error final Data Testing = 0.17823%
MSE final Data Training = 0.0010243%
MSE final Data Validasi = 0.0017053%
MSE total Data Training = 0.048798%
MSE total Data Validasi = 0.061695%
MSE total Data Testing = 0.0010912%
```

Gambar 4.22 Laju Pembelajaran 0.5

```
Error final Data Training = 0.17973%
Error final Data Validasi = 0.19321%
Error final Data Testing = 0.17962%
MSE final Data Training = 0.00076262%
MSE final Data Validasi = 0.0010099%
MSE total Data Training = 0.0043337%
MSE total Data Validasi = 0.0023421%
MSE total Data Testing = 0.00070411%
```

Gambar 4.23 Laju Pembelajaran 0.7

4.3.2 Tahap 2

```
Error final Data Training = 0.17547%
Error final Data Validasi = 0.192%
Error final Data Testing = 0.17056%
MSE final Data Training = 0.00090099%
MSE final Data Validasi = 0.0012737%
MSE total Data Training = 0.00098599%
MSE total Data Validasi = 0.0013124%
MSE total Data Testing = 0.00093971%
```

Gambar 4.24 Laju Pembelajaran 0.1

```
Error final Data Training = 0.19806%
Error final Data Validasi = 0.19874%
Error final Data Testing = 0.17483%
MSE final Data Training = 0.00098792%
MSE final Data Validasi = 0.0012622%
MSE total Data Training = 0.0011066%
MSE total Data Validasi = 0.0012997%
MSE total Data Testing = 0.00093369%
```

Gambar 4.25 Laju Pembelajaran 0.3

```
Error final Data Training = 0.17312%
Error final Data Validasi = 0.17537%
Error final Data Testing = 0.16348%
MSE final Data Training = 0.00070124%
MSE final Data Validasi = 0.00092221%
MSE total Data Training = 0.0012373%
MSE total Data Validasi = 0.0012511%
MSE total Data Testing = 0.00069728%
```

Gambar 4.26 Laju Pembelajaran 0.5

```
Error final Data Training = 0.19985%
Error final Data Validasi = 0.19191%
Error final Data Testing = 0.18121%
MSE final Data Training = 0.00086208%
MSE final Data Validasi = 0.0010201%
MSE total Data Training = 0.0013216%
MSE total Data Validasi = 0.0012196%
MSE total Data Testing = 0.0007597%
```

Gambar 4.27 Laju Pembelajaran 0.7

Pada percobaan ini, ingin dilihat pengaruh dari nilai laju pembelajaran terhadap nilai MSE pada data pengujian (utamanya). Pengujian dilakukan dengan 4 variasi nilai laju pembelajaran yang berbeda, yaitu [0.1; 0.3; 0.5; 0.7]. Proses pelatihan akan dilakukan sebanyak 1000 epoch pada setiap percobaan. Dapat diketahui dari variasi nilai laju pembelajaran, diketahui jika perubahan nilai laju pembelajaran dari model akan memberikan perubahan yang cukup signifikan terhadap nilai MSE dari model ANN yang telah dibangun. Nilai MSE yang diperoleh baik itu dari tahap 1 dan tahap 2 untuk semua variasi nilai laju pembelajaran bisa dibilang masih sangat kecil nilainya (bagus). Akan tetapi, terdapat kecenderungan perubahan nilai MSE data *testing* yang diperoleh untuk peningkatan nilai laju pembelajaran.

Pada percobaan tahap 1, nilai MSE data pengujian terkecil yaitu sebesar 0.000704% yang diperoleh dengan menggunakan laju pembelajaran sebesar 0.7. Adapun pada tahap 2, nilai MSE data pengujian terkecil yaitu sebesar 0.000697% yang diperoleh dengan menggunakan nilai laju pembelajaran sebesar 0.5. Hal ini menunjukkan bahwa, untuk dataset dan *hyperparameter* yang digunakan, model ANN dengan nilai laju pembelajaran antara 0.5 – 0. akan memberikan performa yang terbaik dengan nilai MSE paling kecil.

Nilai laju pembelajaran yang terlalu kecil akan membuat model ANN bergerak menuju minimum lokal dengan langkah yang sangat kecil saat proses pelatihan. Hal ini menyebabkan pelatihan membutuhkan waktu yang lambat untuk mencapai konvergensi.

Di lain sisi, nilai laju pembelajaran yang terlalu tinggi akan menyebabkan model dapat melompati minimum lokal yang diinginkan dan tidak akan pernah mencapainya. Ini menyebabkan pelatihan akan mengalami divergensi dimana nilai MSE akan meningkat.

Itulah sebabnya, harus digunakan nilai laju pembelajaran yang optimal berdasarkan beberapa pertimbangan yang ada, seperti tingkat kompleksitas dataset, ataupun *hyperparameter* yang digunakan selama proses pelatihan dilakukan. Adapun pada tahap 1, nilai laju pembelajaran yang optimal 0.7 dan di tahap 2 0.5. Perbedaan nilai optimal dari laju pembelajaran ini bisa disebabkan karena perbedaan prosedur dari proses *training*, validasi, dan *testing* pada tahap 1 dan 2, dimana pada tahap 2 nilai keluaran estimasi model ANN akan di-*feedback* sebagai masukan model.

4.4 Pengaruh Laju Pembelajaran Momentum Terhadap Mean Squared Error

- Jumlah hidden neuron = 10
- Jumlah iterasi = 1000

- Laju pembelajaran = 0.1

4.4.1 Tahap 1

```
Error final Data Training = 0.31262%
Error final Data Validasi = 0.34806%
Error final Data Testing = 0.32598%
MSE final Data Training = 0.0028671%
MSE final Data Validasi = 0.0048908%
MSE total Data Training = 0.029296%
MSE total Data Validasi = 0.036905%
MSE total Data Testing = 0.0034369%
```

Gambar 4.28 Laju Pembelajaran Momentum 0.1

```
Error final Data Training = 0.25211%
Error final Data Validasi = 0.28132%
Error final Data Testing = 0.26354%
MSE final Data Training = 0.001949%
MSE final Data Validasi = 0.0034498%
MSE total Data Training = 0.014077%
MSE total Data Validasi = 0.017122%
MSE total Data Testing = 0.0023665%
```

Gambar 4.29 Laju Pembelajaran Momentum 0.3

```
Error final Data Training = 0.14482%
Error final Data Validasi = 0.16758%
Error final Data Testing = 0.15531%
MSE final Data Training = 0.00083786%
MSE final Data Validasi = 0.0015038%
MSE total Data Training = 0.0054648%
MSE total Data Validasi = 0.0065047%
MSE total Data Testing = 0.00098407%
```

Gambar 4.30 Laju Pembelajaran Momentum 0.7

```
Error final Data Training = 0.17166%
Error final Data Validasi = 0.16922%
Error final Data Testing = 0.15584%
MSE final Data Training = 0.00087963%
MSE final Data Validasi = 0.001388%
MSE total Data Training = 0.04391%
MSE total Data Validasi = 0.060618%
MSE total Data Testing = 0.00086907%
```

Gambar 4.31 Laju Pembelajaran Momentum 0.9

4.4.2 Tahap 2

```
Error final Data Training = 0.17089%
Error final Data Validasi = 0.18441%
Error final Data Testing = 0.16788%
MSE final Data Training = 0.001051%
MSE final Data Validasi = 0.0015075%
MSE total Data Training = 0.001177%
MSE total Data Validasi = 0.0016431%
MSE total Data Testing = 0.0010777%
```

Gambar 4.32 Laju Pembelajaran 0.1

```
Error final Data Training = 0.17161%
Error final Data Validasi = 0.18451%
Error final Data Testing = 0.1682%
MSE final Data Training = 0.0010339%
MSE final Data Validasi = 0.0014684%
MSE total Data Training = 0.0011482%
MSE total Data Validasi = 0.0016065%
MSE total Data Testing = 0.0010506%
```

Gambar 4.33 Laju Pembelajaran 0.3

```
Error final Data Training = 0.17932%
Error final Data Validasi = 0.18685%
Error final Data Testing = 0.16993%
MSE final Data Training = 0.00097867%
MSE final Data Validasi = 0.0013411%
MSE total Data Training = 0.0010964%
MSE total Data Validasi = 0.001481%
MSE total Data Testing = 0.00094518%
```

Gambar 4.34 Laju Pembelajaran 0.7

```
Error final Data Training = 0.19976%
Error final Data Validasi = 0.2023%
Error final Data Testing = 0.17701%
MSE final Data Training = 0.0010341%
MSE final Data Validasi = 0.0013956%
MSE total Data Training = 0.0012331%
MSE total Data Validasi = 0.0014478%
MSE total Data Testing = 0.00087503%
```

Gambar 4.35 Laju Pembelajaran 0.9

Pada percobaan ini, ingin dilihat pengaruh dari nilai laju pembelajaran momentum terhadap nilai MSE pada data pengujian (utamanya). Pengujian dilakukan dengan 4 variasi nilai laju pembelajaran yang berbeda, yaitu [0.1; 0.3; 0.7; 0.9]. Proses pelatihan akan dilakukan sebanyak 1000 epoch pada setiap percobaan. Dapat diketahui dari variasi nilai laju pembelajaran momentum, diketahui jika perubahan nilai laju pembelajaran momentum dari model akan memberikan perubahan yang cukup signifikan terhadap nilai MSE dari model ANN yang telah dibangun. Nilai MSE yang diperoleh baik itu dari tahap 1 dan tahap 2 untuk semua variasi nilai laju pembelajaran momentum bisa dibilang masih sangat kecil nilainya (bagus). Akan tetapi, terdapat kecenderungan perubahan nilai MSE data *testing* yang diperoleh untuk peningkatan nilai laju pembelajaran momentum.

Pada percobaan tahap 1, nilai MSE data pengujian terkecil yaitu sebesar 0.000869% yang diperoleh dengan menggunakan laju pembelajaran momentum sebesar 0.9. Adapun pada tahap 2, nilai MSE data pengujian terkecil yaitu sebesar 0.000875% yang diperoleh dengan menggunakan nilai laju pembelajaran sebesar 0.9. Hal ini menunjukkan bahwa, untuk dataset dan *hyperparameter* yang digunakan, model ANN dengan nilai laju pembelajaran momentum sebsar 0.9 akan memberikan performa yang terbaik dengan nilai MSE paling kecil.

Nilai laju pembelajaran momentum yang terlalu kecil akan membuat pengaruh gradien sebelumnya pada pembaruan berikutnya akan menjadi sangat kecil sehingga model akan kesuliat mengatasi rintangan pada *error surface*.

Di lain sisi, nilai laju pembelajaran momentum yang terlalu tinggi akan menyebabkan gradien sebelumnya akan memberikan pengaruh besar pada pembaruan berikutnya sehingga model dapat melompati minimum lokal atau terjebak dalam osilasi sehingga tidak konvergen.

Itulah sebabnya, harus digunakan nilai laju pembelajaran yang optimal berdasarkan beberapa pertimbangan yang ada, seperti tingkat kompleksitas dataset, ataupun *hyperparameter* yang digunakan selama proses pelatihan dilakukan. Adapun pada tahap 1 dan 2, nilai laju pembelajara momentum yan optimal yaitu sebesar 0.9. Artinya, dengan nilai laju pembelajaran momentum sebesar 0.9, konvergensi dari model ANN akan cepat dan berdasarkan dataset yang ada (yang dipeoroleh dari data *random*), secara kebetulan dengan nilai laju pembelajara momentum sebesar 0.9 akan memberikan hasil yang optimal dengan nilai konvergensi yang cepat (tanpa terjebak dalam osilasi) dan hasil prediksi yang akurat.

4.5 Analisis Keseluruhan

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 nilai *mean squared error* yang sangat kecil, yakni berada di bawah 0,01%. 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 diperoleh dari *plant* orde dua yang dapat dibilang cukup sederhana. Hal ini menyebabkan cukup mudah bagi model ANN untuk mempelajari pola-pola yang ada dalam data. Tidak heran jika dataset yang diperoleh dari *plant* merupakan salah satu faktor penting penyebab performa model yang 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 performa dari model (nilai *mean squared error*). 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 laju pembelajaran dan laju pembelajaran momentum 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. Akan tetapi, dari percobaan yang dilakukan, ternyata nilai laju pembelajaran yang optimal berada di rentang 05 – 0.7 dan laju pembelajaran momentum sebesar 0.9. Ini artinya, secara garis besar, nilai alpha sebesar 0.2 – 0.4 dan miu sebesar 0.7 – 0.8 akan menghasilkan performa model yang memuaskan. Apakah itu yang terbaik? Jawaban dari nilai laju pembelajaran dan laju pembelajaran momentum yang terbaik tentu dapat diketahui berdasarkan percobaan yang dilakukan. Berdasarkan nilai parameter model, tentunya akan diketahui nilai alpha dan miu yang terbaik. Terlepas dari itu, nilai alpha 0.2 – 0.4 dan miu 0.7 – 0.8 tetap akan memberikan performa yang sangat baik dari model ANN yang dibangun (bagus untuk berbagai variasi jumlahh *hidden neuron*, jumlah *epoch*, dan dataset yang ada).

Pada proses pelatihan model, nilai laju pembelajaran dan laju pembelajaran momentum yang terlalu besar dapat menyebabkan terjadinya 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. Nilai alpha dan miu yang optimal pun dapat diketahui melalui berbagai metode, seperti eksperimen bertahap, validasi silang, *grid search*, dan lain sebagainya.

Tidak hanya alpha dan miu, jumlah *hidden neuron* memiliki efektivitas yang baik. Jumlah *hidden neuron* banyak tidak selalu memberikan hasil yang lebih baik. Sebaliknya, jumlah *hidden neuron* yang sedikit tidak selalu memberikan hasil yang lebih buruk. Jumlah *hidden neuron* yang optimal bagi model ANN dapat diketahui berdasarkan beberapa pertimbangan, seperti kompleksitas dataset, jumlah *epoch*, ataupun nilai *hyperparameter* yang digunakan.

Plant yang digunakan untuk meng-*generate* merupakan *plant* orde dua yang bisaa dibilang masih cukup sederhana. Tidak heran jika dataset yang diperoleh dar *plant* sudah bisa diproses dengan baik menggunakan model ANN yang sederhana. Jumlah *hidden* neuron yang sangat banyak tidaklah terlalu dibutuhkan dalam membangun model ANN untuk memproses dataset yang ada. Model ANN yang kompleks akan terkesan "overkill" jika hanya digunakan untuk memproses dataset *plant* orde dua karena semakin kompleks model ANN yang dibangun, maka semakin banyak proses komputasi yang perlu dilakukan.

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). Adapun pada percobaan yang dilakukan, peristiwa *overfitting* ini tidak pernah terjadi untuk variasi jumlah *hidden neuron* yang digunakan.

Sebaliknya, jumlah *hidden neuron* yang terlalu sedikit akan menyebabkan model ANN memiliki kemampuan yang kurang optimal dalam melakukan estimasi keluaran dari *plant*. Hal ini akan menyebabkan performa dari model ANN dengan jumlah *hidden neuron* yang terlalu sedikit akan kalah dengan performa dari model ANN dengan jumlah *hidden neuron* yang optimal.

Dari penjelasan-penjelasan tersebut dapat disimpulkan jika untuk memproses dataset yang diperoleh dari *plant* orde dua, diperlukan model ANN dengan jumlah *hidden neuron*, nilai laju pembelajaran, dan laju pembelajaran momentum yang optimal sehingga bisa dihasilkan model dengan performa yang optimal, dengan nilai MSE yang kecil dan waktu komputasi yang cepat.

BAB V
KESIMPULAN

Dari beberapa percobaan yang telah dilakukan, penulis dapat memberikan kesimpulan yang dapat direpresentasikan dalam beberapa poin poin penting, yaitu:

- Model ANN dapat digunakan untuk memodelkan suatu *plant* pada suatu sistem kendali. *Plant* ini merupakan gambaran dari pengendali dari suatu sistem kendali.
- Model ANN memiliki performa yang baik jika diterapkan sebagai pengendali dalam suatu sistem kendali.
- Nilai laju pembelajaran akan mempengaruhi seberapa cepat model ANN dalam mempelajari pola yang ada dari sekumpulan data. Perlu digunakan nilai laju pembelajaran yang optimal untuk menghasilkan model dengan performa yang optimal (MSE data *testing* kecil).
- Nilai laju pembelajaran momentum akan mempengaruhi seberapa cepat model ANN dalam mempelajari pola yang ada dari sekumpulan data dengan memberikan momentum kepada gradien yang telah dihitung dalam iterasi sebelumnya dan menggabungkannya dengan gradien saat ini. Perlu digunakan nilai laju pembelajaran momentum yang optimal untuk menghasilkan model dengan performa yang optimal (MSE data *testing* kecil)
- Jumlah *hidden neuron* akan mempengaruhi seberapa bagus kemampuan model ANN dalam memahami dan mengekstraksi pola yang ada di dalam sekumpulan data. Perlu digunakan jumlah *hidden neuron* yang optimal untuk menghasilkan model dengan performa yang optimal (MSE data *testing* kecil)
- Normalisasi merupakan tahapan yang penting untuk dilakukan sebelum memulai proses pembelajaran karena dengan normalisasi, akan lebih mudah bagi model untuk mempelajari "pola" yang ada pada sekumpulan data.
- Pada proses memodelkan ANN, perlu digunakan nilai laju pembelajaran, laju pembelajaran momentum, dan jumlah *hidden neuron* yang optimal sehingga akan dihasilkan performa model yang optimal dengan nilai MSE yang kecil.
- Penentuan nilai laju pembelajaran, laju pembelajaran momentum, dan jumlah *hidden neuron* yang optimal dapat dilakukan dengan berbagai metode, mulai dari eksperimen bertahap, validasi silang, *grid search*, dan lain sebagainya.

REFERENSI

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